*Some people, when confronted with a problem, think “I know, I’ll use regular expressions.” Now they have two problems*. – Jamie Zawinski

*Some programmers, when confronted with a problem, think “I know, I’ll use floating point arithmetic.” Now they have 1.999999999997 problems*. – @tomscott

*Some people, when confronted with a problem, think “I know, I’ll use multithreading”. Nothhw tpe yawrve o oblems*. – @d6

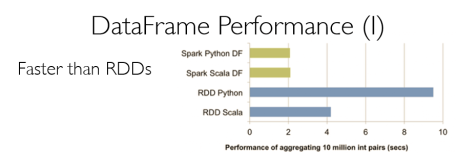
*Some people, when confronted with a problem, think “I know, I’ll use versioning.” Now they have 2.1.0 problems.* – @JaesCoyle

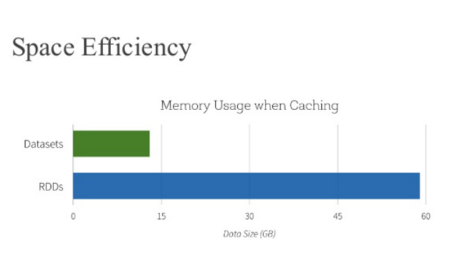
*Some people, when faced with a problem, think, “I know, I’ll use binary.” Now they have 10 problems*. – @nedbat

**Introduction**

The power of Spark, which operates on in-memory datasets, is the fact that it stores the data as collections using Resilient Distributed Datasets (RDDs), which are themselves distributed in partitions across clusters. RDDs, are a fast way of processing data, as the data is operated on parallel based on the map-reduce paradigm. RDDs can be be used when the operations are low level. RDDs, are typically used on unstructured data like logs or text. For structured and semi-structured data, Spark has a higher abstraction called Dataframes. Handling data through dataframes are extremely fast as they are Optimized using the Catalyst Optimization engine and the performance is orders of magnitude faster than RDDs. In addition Dataframes also use Tungsten which handle memory management and garbage collection more effectively.

The picture below shows the performance improvement achieved with Dataframes over RDDs



Benefits from Project Tungsten  


Npte: The above data and graph is taken from the course [Big Data Analysis with Apache Spark](https://courses.edx.org/courses/course-v1:BerkeleyX+CS110x+2T2016/course/) at edX, UC Berkeley  
This post is a continuation of my 2 earlier posts  
1. [Big Data-1: Move into the big league:Graduate from Python to Pyspark](https://gigadom.in/2018/10/08/big-data-1-move-into-the-big-leaguegraduate-from-python-to-pyspark/)  
2. [Big Data-2: Move into the big league:Graduate from R to SparkR](https://gigadom.in/2018/10/09/big-data-2-move-into-the-big-leaguegraduate-from-r-to-sparkr/)

In this post I perform equivalent operations on a small dataset using RDDs, Dataframes in Pyspark & SparkR and HiveQL. As in some of my earlier posts, I have used the tendulkar.csv file for this post. The dataset is small and allows me to do most everything from data cleaning, data transformation and grouping etc.  
You can clone fork the notebooks from github at [Big Data:Part 3](https://github.com/tvganesh/Big-Data---Part-3)

**1. RDD – Select all columns of tables**

from pyspark import SparkContext

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

rdd.map(lambda line: (line.split(","))).take(5)

Out[90]: [[‘Runs’, ‘Mins’, ‘BF’, ‘4s’, ‘6s’, ‘SR’, ‘Pos’, ‘Dismissal’, ‘Inns’, ‘Opposition’, ‘Ground’, ‘Start Date’], [’15’, ’28’, ’24’, ‘2’, ‘0’, ‘62.5’, ‘6’, ‘bowled’, ‘2’, ‘v Pakistan’, ‘Karachi’, ’15-Nov-89′], [‘DNB’, ‘-‘, ‘-‘, ‘-‘, ‘-‘, ‘-‘, ‘-‘, ‘-‘, ‘4’, ‘v Pakistan’, ‘Karachi’, ’15-Nov-89′], [’59’, ‘254’, ‘172’, ‘4’, ‘0’, ‘34.3’, ‘6’, ‘lbw’, ‘1’, ‘v Pakistan’, ‘Faisalabad’, ’23-Nov-89′], [‘8′, ’24’, ’16’, ‘1’, ‘0’, ’50’, ‘6’, ‘run out’, ‘3’, ‘v Pakistan’, ‘Faisalabad’, ’23-Nov-89′]]

**1b.RDD – Select columns 1 to 4**

from pyspark import SparkContext

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

rdd.map(lambda line: (line.split(",")[0:4])).take(5)

Out[91]:  
[[‘Runs’, ‘Mins’, ‘BF’, ‘4s’],  
[’15’, ’28’, ’24’, ‘2’],  
[‘DNB’, ‘-‘, ‘-‘, ‘-‘],  
[’59’, ‘254’, ‘172’, ‘4’],  
[‘8′, ’24’, ’16’, ‘1’]]

**1c. RDD – Select specific columns 0, 10**

from pyspark import SparkContext

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

df=rdd.map(lambda line: (line.split(",")))

df.map(lambda x: (x[10],x[0])).take(5)

Out[92]:  
[(‘Ground’, ‘Runs’),  
(‘Karachi’, ’15’),  
(‘Karachi’, ‘DNB’),  
(‘Faisalabad’, ’59’),  
(‘Faisalabad’, ‘8’)]

**2. Dataframe:Pyspark – Select all columns**

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1.show(5)

+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
|Runs|Mins| BF| 4s| 6s| SR|Pos|Dismissal|Inns|Opposition| Ground|Start Date|  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
| 15| 28| 24| 2| 0| 62.5| 6| bowled| 2|v Pakistan| Karachi| 15-Nov-89|  
| DNB| -| -| -| -| -| -| -| 4|v Pakistan| Karachi| 15-Nov-89|  
| 59| 254|172| 4| 0| 34.3| 6| lbw| 1|v Pakistan|Faisalabad| 23-Nov-89|  
| 8| 24| 16| 1| 0| 50| 6| run out| 3|v Pakistan|Faisalabad| 23-Nov-89|  
| 41| 124| 90| 5| 0|45.55| 7| bowled| 1|v Pakistan| Lahore| 1-Dec-89|  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
only showing top 5 rows

**2a. Dataframe:Pyspark- Select specific columns**

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1.select("Runs","BF","Mins").show(5)

+—-+—+—-+  
|Runs| BF|Mins|  
+—-+—+—-+  
| 15| 24| 28|  
| DNB| -| -|  
| 59|172| 254|  
| 8| 16| 24|  
| 41| 90| 124|  
+—-+—+—-+

**3. Dataframe:SparkR – Select all columns**

# Load the SparkR library

library(SparkR)

# Initiate a SparkR session

sparkR.session()

tendulkar1 <- read.df("/FileStore/tables/tendulkar.csv",

header = "true",

delimiter = ",",

source = "csv",

inferSchema = "true",

na.strings = "")

# Check the dimensions of the dataframe

df=SparkR::select(tendulkar1,"\*")

head(SparkR::collect(df))

Runs Mins BF 4s 6s SR Pos Dismissal Inns Opposition Ground Start Date

1 15 28 24 2 0 62.5 6 bowled 2 v Pakistan Karachi 15-Nov-89

2 DNB - - - - - - - 4 v Pakistan Karachi 15-Nov-89

3 59 254 172 4 0 34.3 6 lbw 1 v Pakistan Faisalabad 23-Nov-89

4 8 24 16 1 0 50 6 run out 3 v Pakistan Faisalabad 23-Nov-89

5 41 124 90 5 0 45.55 7 bowled 1 v Pakistan Lahore 1-Dec-89

6 35 74 51 5 0 68.62 6 lbw 1 v Pakistan Sialkot 9-Dec-89

**3a. Dataframe:SparkR- Select specific columns**

# Load the SparkR library

library(SparkR)

# Initiate a SparkR session

sparkR.session()

tendulkar1 <- read.df("/FileStore/tables/tendulkar.csv",

header = "true",

delimiter = ",",

source = "csv",

inferSchema = "true",

na.strings = "")

# Check the dimensions of the dataframe

df=SparkR::select(tendulkar1, "Runs", "BF","Mins")

head(SparkR::collect(df))

Runs BF Mins  
1 15 24 28  
2 DNB – –  
3 59 172 254  
4 8 16 24  
5 41 90 124  
6 35 51 74

**4. Hive QL – Select all columns**

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1.createOrReplaceTempView('tendulkar1\_table')

spark.sql('select \* from tendulkar1\_table limit 5').show(10, truncate = False)

+—-+—+—-++—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
|Runs|Mins|BF |4s |6s |SR |Pos|Dismissal|Inns|Opposition|Ground |Start Date|  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
|15 |28 |24 |2 |0 |62.5 |6 |bowled |2 |v Pakistan|Karachi |15-Nov-89 |  
|DNB |- |- |- |- |- |- |- |4 |v Pakistan|Karachi |15-Nov-89 |  
|59 |254 |172|4 |0 |34.3 |6 |lbw |1 |v Pakistan|Faisalabad|23-Nov-89 |  
|8 |24 |16 |1 |0 |50 |6 |run out |3 |v Pakistan|Faisalabad|23-Nov-89 |  
|41 |124 |90 |5 |0 |45.55|7 |bowled |1 |v Pakistan|Lahore |1-Dec-89 |  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+

**4a. Hive QL – Select specific columns**

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1.createOrReplaceTempView('tendulkar1\_table')

spark.sql('select Runs, BF,Mins from tendulkar1\_table limit 5').show(10, truncate = False)

+—-+—+—-+

|Runs|BF |Mins|  
+—-+—+—-+  
|15 |24 |28 |  
|DNB |- |- |  
|59 |172|254 |  
|8 |16 |24 |  
|41 |90 |124 |  
+—-+—+—-+

**5. RDD – Filter rows on specific condition**

from pyspark import SparkContext

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

df=(rdd.map(lambda line: line.split(",")[:])

.filter(lambda x: x !="DNB")

.filter(lambda x: x!= "TDNB")

.filter(lambda x: x!="absent")

.map(lambda x: [x[0].replace("\*","")] + x[1:]))

df.take(5)

Out[97]:  
[[‘Runs’,  
‘Mins’,  
‘BF’,  
‘4s’,  
‘6s’,  
‘SR’,  
‘Pos’,  
‘Dismissal’,  
‘Inns’,  
‘Opposition’,  
‘Ground’,  
‘Start Date’],  
[’15’,  
’28’,  
’24’,  
‘2’,  
‘0’,  
‘62.5’,  
‘6’,  
‘bowled’,  
‘2’,  
‘v Pakistan’,  
‘Karachi’,  
’15-Nov-89′],  
[‘DNB’,  
‘-‘,  
‘-‘,  
‘-‘,  
‘-‘,  
‘-‘,  
‘-‘,  
‘-‘,  
‘4’,  
‘v Pakistan’,  
‘Karachi’,  
’15-Nov-89′],  
[’59’,  
‘254’,  
‘172’,  
‘4’,  
‘0’,  
‘34.3’,  
‘6’,  
‘lbw’,  
‘1’,  
‘v Pakistan’,  
‘Faisalabad’,  
’23-Nov-89′],  
[‘8′,  
’24’,  
’16’,  
‘1’,  
‘0’,  
’50’,  
‘6’,  
‘run out’,  
‘3’,  
‘v Pakistan’,  
‘Faisalabad’,  
’23-Nov-89′]]

**5a. Dataframe:Pyspark – Filter rows on specific condition**

from pyspark.sql import SparkSession

from pyspark.sql.functions import regexp\_replace

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'DNB')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'TDNB')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'absent')

tendulkar1 = tendulkar1.withColumn('Runs', regexp\_replace('Runs', '[\*]', ''))

tendulkar1.show(5)

+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
|Runs|Mins| BF| 4s| 6s| SR|Pos|Dismissal|Inns|Opposition| Ground|Start Date|  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
| 15| 28| 24| 2| 0| 62.5| 6| bowled| 2|v Pakistan| Karachi| 15-Nov-89|  
| 59| 254|172| 4| 0| 34.3| 6| lbw| 1|v Pakistan|Faisalabad| 23-Nov-89|  
| 8| 24| 16| 1| 0| 50| 6| run out| 3|v Pakistan|Faisalabad| 23-Nov-89|  
| 41| 124| 90| 5| 0|45.55| 7| bowled| 1|v Pakistan| Lahore| 1-Dec-89|  
| 35| 74| 51| 5| 0|68.62| 6| lbw| 1|v Pakistan| Sialkot| 9-Dec-89|  
+—-+—-+—+—+—+—–+—+———+—-+———-+———-+———-+  
only showing top 5 rows

**5b. Dataframe:SparkR – Filter rows on specific condition**

sparkR.session()

tendulkar1 <- read.df("/FileStore/tables/tendulkar.csv",

header = "true",

delimiter = ",",

source = "csv",

inferSchema = "true",

na.strings = "")

print(dim(tendulkar1))

tendulkar1 <-SparkR::filter(tendulkar1,tendulkar1$Runs != "DNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "TDNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "absent")

print(dim(tendulkar1))

# Cast the string type Runs to double

withColumn(tendulkar1, "Runs", cast(tendulkar1$Runs, "double"))

head(SparkR::distinct(tendulkar1[,"Runs"]),20)

# Remove the "\* indicating not out

tendulkar1$Runs=SparkR::regexp\_replace(tendulkar1$Runs, "\\\*", "")

df=SparkR::select(tendulkar1,"\*")

head(SparkR::collect(df))

**5c Hive QL – Filter rows on specific condition**

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1.createOrReplaceTempView('tendulkar1\_table')

spark.sql('select Runs, BF,Mins from tendulkar1\_table where Runs NOT IN ("DNB","TDNB","absent")').show(10, truncate = False)

+—-+—+—-+  
|Runs|BF |Mins|  
+—-+—+—-+  
|15 |24 |28 |  
|59 |172|254 |  
|8 |16 |24 |  
|41 |90 |124 |  
|35 |51 |74 |  
|57 |134|193 |  
|0 |1 |1 |  
|24 |44 |50 |  
|88 |266|324 |  
|5 |13 |15 |  
+—-+—+—-+  
only showing top 10 rows

**6. RDD – Find rows where Runs > 50**

from pyspark import SparkContext

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

df=rdd.map(lambda line: (line.split(",")))

df=rdd.map(lambda line: line.split(",")[0:4]) \

.filter(lambda x: x[0] not in ["DNB", "TDNB", "absent"])

df1=df.map(lambda x: [x[0].replace("\*","")] + x[1:4])

header=df1.first()

df2=df1.filter(lambda x: x !=header)

df3=df2.map(lambda x: [float(x[0])] +x[1:4])

df3.filter(lambda x: x[0]>=50).take(10)

Out[101]:

[[59.0, '254', '172', '4'],

[57.0, '193', '134', '6'],

[88.0, '324', '266', '5'],

[68.0, '216', '136', '8'],

[119.0, '225', '189', '17'],

[148.0, '298', '213', '14'],

[114.0, '228', '161', '16'],

[111.0, '373', '270', '19'],

[73.0, '272', '208', '8'],

[50.0, '158', '118', '6']]

**6a. Dataframe:Pyspark – Find rows where Runs >50**

from pyspark.sql import SparkSession

from pyspark.sql.functions import regexp\_replace

from pyspark.sql.types import IntegerType

spark = SparkSession.builder.appName('Read CSV DF').getOrCreate()

tendulkar1 = spark.read.format('csv').option('header','true').load('/FileStore/tables/tendulkar.csv')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'DNB')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'TDNB')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'absent')

tendulkar1 = tendulkar1.withColumn("Runs", tendulkar1["Runs"].cast(IntegerType()))

tendulkar1.filter(tendulkar1['Runs']>=50).show(10)

+—-+—-+—+—+—+—–+—+———+—-+————–+————+———-+  
|Runs|Mins| BF| 4s| 6s| SR|Pos|Dismissal|Inns| Opposition| Ground|Start Date|  
+—-+—-+—+—+—+—–+—+———+—-+————–+————+———-+  
| 59| 254|172| 4| 0| 34.3| 6| lbw| 1| v Pakistan| Faisalabad| 23-Nov-89|  
| 57| 193|134| 6| 0|42.53| 6| caught| 3| v Pakistan| Sialkot| 9-Dec-89|  
| 88| 324|266| 5| 0|33.08| 6| caught| 1| v New Zealand| Napier| 9-Feb-90|  
| 68| 216|136| 8| 0| 50| 6| caught| 2| v England| Manchester| 9-Aug-90|  
| 114| 228|161| 16| 0| 70.8| 4| caught| 2| v Australia| Perth| 1-Feb-92|  
| 111| 373|270| 19| 0|41.11| 4| caught| 2|v South Africa|Johannesburg| 26-Nov-92|  
| 73| 272|208| 8| 1|35.09| 5| caught| 2|v South Africa| Cape Town| 2-Jan-93|  
| 50| 158|118| 6| 0|42.37| 4| caught| 1| v England| Kolkata| 29-Jan-93|  
| 165| 361|296| 24| 1|55.74| 4| caught| 1| v England| Chennai| 11-Feb-93|  
| 78| 285|213| 10| 0|36.61| 4| lbw| 2| v England| Mumbai| 19-Feb-93|  
+—-+—-+—+—+—+—–+—+———+—-+————–+————+———-+

**6b. Dataframe:SparkR – Find rows where Runs >50**

# Load the SparkR library

library(SparkR)

sparkR.session()

tendulkar1 <- read.df("/FileStore/tables/tendulkar.csv",

header = "true",

delimiter = ",",

source = "csv",

inferSchema = "true",

na.strings = "")

print(dim(tendulkar1))

tendulkar1 <-SparkR::filter(tendulkar1,tendulkar1$Runs != "DNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "TDNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "absent")

print(dim(tendulkar1))

# Cast the string type Runs to double

withColumn(tendulkar1, "Runs", cast(tendulkar1$Runs, "double"))

head(SparkR::distinct(tendulkar1[,"Runs"]),20)

# Remove the "\* indicating not out

tendulkar1$Runs=SparkR::regexp\_replace(tendulkar1$Runs, "\\\*", "")

df=SparkR::select(tendulkar1,"\*")

df=SparkR::filter(tendulkar1, tendulkar1$Runs > 50)

head(SparkR::collect(df))

Runs Mins BF 4s 6s SR Pos Dismissal Inns Opposition Ground

1 59 254 172 4 0 34.3 6 lbw 1 v Pakistan Faisalabad

2 57 193 134 6 0 42.53 6 caught 3 v Pakistan Sialkot

3 88 324 266 5 0 33.08 6 caught 1 v New Zealand Napier

4 68 216 136 8 0 50 6 caught 2 v England Manchester

5 119 225 189 17 0 62.96 6 not out 4 v England Manchester

6 148 298 213 14 0 69.48 6 not out 2 v Australia Sydney

Start Date

1 23-Nov-89

2 9-Dec-89

3 9-Feb-90

4 9-Aug-90

5 9-Aug-90

6 2-Jan-92

**7 RDD – groupByKey() and reduceByKey()**

from pyspark import SparkContext

from pyspark.mllib.stat import Statistics

rdd = sc.textFile( "/FileStore/tables/tendulkar.csv")

df=rdd.map(lambda line: (line.split(",")))

df=rdd.map(lambda line: line.split(",")[0:]) \

.filter(lambda x: x[0] not in ["DNB", "TDNB", "absent"])

df1=df.map(lambda x: [x[0].replace("\*","")] + x[1:])

header=df1.first()

df2=df1.filter(lambda x: x !=header)

df3=df2.map(lambda x: [float(x[0])] +x[1:])

df4 = df3.map(lambda x: (x[10],x[0]))

df5=df4.reduceByKey(lambda a,b: a+b,1)

df4.groupByKey().mapValues(lambda x: sum(x) / len(x)).take(10)

[(‘Georgetown’, 81.0),  
(‘Lahore’, 17.0),  
(‘Adelaide’, 32.6),  
(‘Colombo (SSC)’, 77.55555555555556),  
(‘Nagpur’, 64.66666666666667),  
(‘Auckland’, 5.0),  
(‘Bloemfontein’, 85.0),  
(‘Centurion’, 73.5),  
(‘Faisalabad’, 27.0),  
(‘Bridgetown’, 26.0)]

**7a Dataframe:Pyspark – Compute mean, min and max**

from pyspark.sql.functions import \*

tendulkar1= (sqlContext

.read.format("com.databricks.spark.csv")

.options(delimiter=',', header='true', inferschema='true')

.load("/FileStore/tables/tendulkar.csv"))

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'DNB')

tendulkar1= tendulkar1.where(tendulkar1['Runs'] != 'TDNB')

tendulkar1 = tendulkar1.withColumn('Runs', regexp\_replace('Runs', '[\*]', ''))

tendulkar1.select('Runs').rdd.distinct().collect()

from pyspark.sql import functions as F

df=tendulkar1[['Runs','BF','Ground']].groupby(tendulkar1['Ground']).agg(F.mean(tendulkar1['Runs']),F.min(tendulkar1['Runs']),F.max(tendulkar1['Runs']))

df.show()

————-+—————–+———+———+  
| Ground| avg(Runs)|min(Runs)|max(Runs)|  
+————-+—————–+———+———+  
| Bangalore| 54.3125| 0| 96|  
| Adelaide| 32.6| 0| 61|  
|Colombo (PSS)| 37.2| 14| 71|  
| Christchurch| 12.0| 0| 24|  
| Auckland| 5.0| 5| 5|  
| Chennai| 60.625| 0| 81|  
| Centurion| 73.5| 111| 36|  
| Brisbane|7.666666666666667| 0| 7|  
| Birmingham| 46.75| 1| 40|  
| Ahmedabad| 40.125| 100| 8|  
|Colombo (RPS)| 143.0| 143| 143|  
| Chittagong| 57.8| 101| 36|  
| Cape Town|69.85714285714286| 14| 9|  
| Bridgetown| 26.0| 0| 92|  
| Bulawayo| 55.0| 36| 74|  
| Delhi|39.94736842105263| 0| 76|  
| Chandigarh| 11.0| 11| 11|  
| Bloemfontein| 85.0| 15| 155|  
|Colombo (SSC)|77.55555555555556| 104| 8|  
| Cuttack| 2.0| 2| 2|  
+————-+—————–+———+———+  
only showing top 20 rows

**7b Dataframe:SparkR – Compute mean, min and max**

sparkR.session()

tendulkar1 <- read.df("/FileStore/tables/tendulkar.csv",

header = "true",

delimiter = ",",

source = "csv",

inferSchema = "true",

na.strings = "")

print(dim(tendulkar1))

tendulkar1 <-SparkR::filter(tendulkar1,tendulkar1$Runs != "DNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "TDNB")

print(dim(tendulkar1))

tendulkar1<-SparkR::filter(tendulkar1,tendulkar1$Runs != "absent")

print(dim(tendulkar1))

# Cast the string type Runs to double

withColumn(tendulkar1, "Runs", cast(tendulkar1$Runs, "double"))

head(SparkR::distinct(tendulkar1[,"Runs"]),20)

# Remove the "\* indicating not out

tendulkar1$Runs=SparkR::regexp\_replace(tendulkar1$Runs, "\\\*", "")

head(SparkR::distinct(tendulkar1[,"Runs"]),20)

df=SparkR::summarize(SparkR::groupBy(tendulkar1, tendulkar1$Ground), mean = mean(tendulkar1$Runs), minRuns=min(tendulkar1$Runs),maxRuns=max(tendulkar1$Runs))

head(df,20)

Ground mean minRuns maxRuns

1 Bangalore 54.312500 0 96

2 Adelaide 32.600000 0 61

3 Colombo (PSS) 37.200000 14 71

4 Christchurch 12.000000 0 24

5 Auckland 5.000000 5 5

6 Chennai 60.625000 0 81

7 Centurion 73.500000 111 36

8 Brisbane 7.666667 0 7

9 Birmingham 46.750000 1 40

10 Ahmedabad 40.125000 100 8

11 Colombo (RPS) 143.000000 143 143

12 Chittagong 57.800000 101 36

13 Cape Town 69.857143 14 9

14 Bridgetown 26.000000 0 92

15 Bulawayo 55.000000 36 74

16 Delhi 39.947368 0 76

17 Chandigarh 11.000000 11 11

18 Bloemfontein 85.000000 15 155

19 Colombo (SSC) 77.555556 104 8

20 Cuttack 2.000000 2 2