☐ Home / Spark / Spark Use Case – Travel Data Analysis



08 MAY 2016

# Spark Use Case – Travel Data Analysis



In this blog, we will discuss on the analysis of travel dataset and gain insights from the dataset using Apache Spark.

The travel dataset is publically available and the contents are detailed under the heading, 'Travel Sector Dataset Description'.

Based on the data, we will find the top 20 destination people travel the most, top 20 locations from where people travel the most, top 20 cities that generate high airline revenues for travel, based on booked trip count.

## **Travel Sector Dataset Description**

Column 1: City pair (Combination of *from* and *to*): String

Column 2: From location: String

YES, I WANT TO BOOST MY CAREER & INCREASE MY					
SALARY!					
Your Name					

Your Email (required)

Your Contact
Number (required)

Your Message

TELL ME HOW

## LIKE WHAT YOU SEE? SUBSCRIBE TO OUR BLOG

We send only 1 email in a week

Enter your email...

Subscribe

5/17/2016

Column 3: To Location: String

Column 4: Product type: Integer (1=Air, 2=Car, 3 =Air+Car, 4 =Hotel, 5=Air+Hotel, 6=Hotel +Car, 7 =Air+Hotel+Car)

**SEARCH** 

fl Search Now

## ACADGILD

**CATEGORIES \( \rightarrow\)** 

This summer, give your child a skill for life. Check out our technology summer camps!

Tell Me More

Column 7: Children traveling: Integer

Column 8: Youth traveling: Integer

Column 9: Infant traveling: Integer

Column 10: Date of travel: String

Column 11: Time of travel: String

Column 12: Date of Return: String

Column 13: Time of Return: String

Column 14: Price of booking: Float

Column 15: Hotel name: String

You can download the dataset from the link below:

https://drive.google.com/open? id=0ByJLBTmJojjzZEg2bXpYa0dyd1k

#### **Problem Statement 1**

Top 20 destination people travel the most: Based on the given data, we can find the most popular destination that people travel frequently. There are many destinations out of which we will find only first 20, based on trips booked for particular destinations.

## **Source Code**

1 | val textFile = sc.textFile("hdfs://localhost:9000/TravelData.tx 2 | t")

- Android
- Android For Kids
- AngularJS
- Big Data and Hadoop
- Careers
- Cloud computing
- Database
- Digital Marketing
- Front End
- Full Stack
- Hadoop Administration
- IOS
- Java
- Kids
- Linux Administration
- NodeJS
- Others
- Python
- Quiz

val split = textFile.map(lines=>lines.split('\t')).map(x=>(x(2),
1)).reduceByKey(\_+\_).map(item => item.swap).sortByKey(fal
se).take(20)

## Description of the above code

**Line 1:** We are creating an RDD by loading a new dataset which is in HDFS.

Line 2: We have split each record by taking the delimiter as *tab* because the data is tab separated. We are creating the key-value pair, where key is the *destination* that is in 3<sup>rd</sup> column and the value is 1. Since we need to count the cities which are popular, we are using the reduceByKey method to count them. After counting the destinations, we are swapping the key-value pairs. The sortByKey method sorts the data with keys and *false* stands for descending order. Once the sorting is complete, we are considering the top 20 destinations.

## **Output**

(396,MIA), (290,SFO), (202,LAS), (162,LAX), (102,DFW), (64,DEN), (57,ORD), (54,PHL), (50,IAH), (45,JFK), (44,PHX), (40,FLL), (36,ATL), (31,BOS), (31,MCO), (27,SAN), (25,WAS), (24,CUN), (22,AUS), (22,LON)

You can see the same in the below screen shot.

scala> val textFile = sc.textFile("hdfs://localhost:9000/TravelData.txt")
textFile: org.apache.spark.rdd.R00[string] = RapPartitionsR00[i3] at textFile at <console>:21
scala> val split = textFile.map(lines>=lines.split('\t')).map(x>>(x(2),1)).reduceByKey(\_\*\_\_).map(line >> \ten.swap).sortByKey(false).take(20)
split: Array([int, String)] = Array((396,RIA), (296,SF0), (282,LAS), (162,LAS), (102,DFW), (64,DEW), (57,ORD), (54,PHL), (36,IAH), (45,JFK), (44,PKL), (36,FLL), (36,ATL), (31,BOS), (31,RCO), (27,SAN), (25,MAS), (24,CUM), (22,AUS), (22,LOM))

#### **Problem Statement 2**

Think you know it all about Spark? Take this simple quiz to find out!

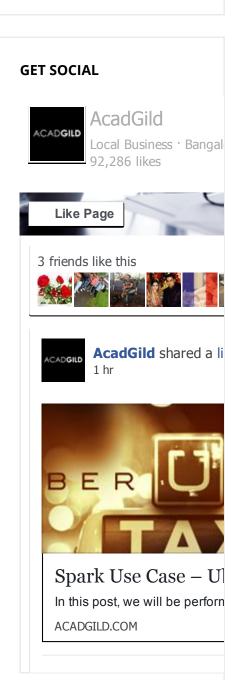


Yes, I'm Game

Top 20 locations from where people travel the most: We can find the places from where most of the trips are undertaken, based on the booked trip count.

## **Source Code**

R & Machine Learn
 Scala
 Spark
 Uncategorized



#### WHAT'S TRENDING

```
val textFile = sc.textFile("hdfs://localhost:9000/TravelData.tx
t")

val split = textFile.map(lines=>lines.split('\t')).map(x=>(x(1), 1)).reduceByKey(_+_).map(item => item.swap).sortByKey(false).take(20)
```

## Description of the above code

**Line 1:** We are creating an RDD by loading a new dataset which is in HDFS.

Line 2: We have split each record by taking the delimiter as *tab* since the data is tab separated. We are creating the key-value pair, where key is the *location from where people start*, that is in the 2<sup>nd</sup> column and the value is 1. Since we need to count the cities which are popular locations from where people undertake the trips, we are using the reduceByKey method to count them. After counting the locations, we are swapping the key-value pairs. We are using the sortByKey method which sorts the data with keys where *false* stands for descending order. Once the sorting is complete, we are taking the top 20 locations from where people undertake the trips.

## Output

(504,DFW), (293,MIA), (272,LAS), (167,BOM), (131,SFO), (101,ORD), (72,LAX), (55,DEN), (41,PHL), (37,IAH), (35,FLL), (33,PHX), (31,JFK), (24,WAS), (19,HOU), (19,ATL), (18,DXB), (17,SAN), (17,BOS), (17,BCN)

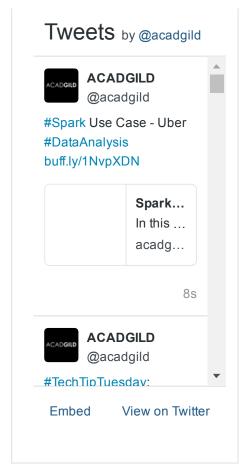
You can see the same in the below screen shot.

```
scala> val textFile = sc.textFile("hdfs://localhost:9000/TravelData.txt")
textFile: org.apache.spark.rdd.RD0[String] = MapPartitionsRD0[22] at textFile at <console>:21
scala>
scala> val split = textFile.nap(lines=>lines.split('\t')).nap(x=>(x(1),1)).reduceByKey(-+).nap(iten => iten.swap).sortByKey(false).take(20)
split: Array[[int, String)] = Array((504,DFM), (293,MIA), (272,LAS), (167,B0N), (131,SFD), (101,DRD), (72,LAX), (55,DEN), (41,PHL), (37,LAH), (35,LL), (33,PMX), (31,FFK), (24,MAS), (19,ATL), (18,DXB), (17,SAN), (17,BOS), (17,BCN))
```

#### **Problem Statement 3**

Top 20 cities that generate high airline revenues for travel, so that the site can concentrate on offering discount on booking, to those cities to attract more bookings.

## **Source Code**







Spark Use Case

– Uber Data

Analysis

□ May 16, 2016



Why Learning
MongoDB Will
Boost Your
Career

□ May 14, 2016



Job Responsibilities of Hadoop Professionals

□ May 13, 2016



Graphical Exploratory Data Analysis-II

□ May 13, 2016

```
val textFile = sc.textFile("hdfs://localhost:9000/TravelData.tx
t")

val fil = textFile.map(x=>x.split('\t')).filter(x=>{if((x(3).match es(("1")))) true else false })

val cnt = fil.map(x=>(x(2),1)).reduceByKey(_+_).map(item = > item.swap).sortByKey(false).take(20)
```

## Description of the above code

**Line 1:** We are creating an RDD by loading a new dataset which is in HDFS.

**Line 2:** We are splitting each record based on the delimiter tab as the data is tab separated. From this, we are filtering the records based on the mode of travel. Here, we need the count of people who travelled by flight which is denoted by **1** (1=Air, 2=Car, 3 =Air+Car, 4 =Hotel, 5=Air+Hotel, 6=Hotel +Car, 7 =Air+Hotel+Car).

Line 3: We are creating the key-value pairs for those people who travelled by air, where key is the *destination* which is in 3<sup>rd</sup> column and value is 1. Since we need to count the popular cities, we are counting them by using the reduceByKey method. After counting the destinations, we are swapping the key-value pairs. We are using the sortByKey method to sort the data with keys where *false* stands for descending order. Once sorting is completed, we are considering top 20 cities that generate high airline revenues for travel.

## **Output:**

(84,MIA), (68,SFO), (54,LAS), (42,LAX), (24,IAH), (23,DFW), (18,PHX), (17,BOS), (15,ORD), (13,NYC), (9,DCA), (8,WAS), (8,AUS), (7,DEN), (7,MEM), (7,JFK), (6,SYD), (6,PHL), (6,ATL), (5,RIC)

You can see the same in the below screen shot.

```
scala> val textFile = sc.textFile("hdfs://localhost:9000/fravelData.txt")
textFile: org.apache.spark.rdd.RDD[String] = MapPartLtionsRDD[33] at textFile at <console>:21
scala> val fil = textFile.nap(x>x.split('\tau').filter(x>=\fi(X(3).natches(('1')))) true else false })
fil: org.apache.spark.rdd.RDD[Array[String]] = MapPartLtionsRDD[35] at filter at <console>:23
scala> val cnt = fil.nap(x>=(X:0,1)).reduce8yKey(___).nap(Iten => Iten.swap).sortByMey(false).take(20)
cnt: Array[(int, String)] = Array((84,MIA), (68,SFD), (54,LAS), (42,LAM), (24,LAM), (23,DFM), (18,PMX), (17,BDS), (15,ORD), (13,NYC), (9,DCA), (8,MS), (8,AUS), (7,DEM), (7,DEM),
```

We hope this blog was useful. Keep visiting our site

#### **ARCHIVES**

- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- November 2014
- October 2014
- September 2014
- August 2014

www.acadgild.com for more updates on BigData and other technologies.



Learn SPARK from our Expert Mentors in

just 12 weeks and Boost your Career

**Enroll Today** 

#### Share this:













#### Related

Spark Use Case -Youtube Data Analysis

April 5, 2016 In "Spark"

Map reduce Use Analysis

November 6, 2015 In "Spark" In "AcadGild"

Spark Use Case – April 16, 2016



## KIRAN KRISHNA

Kiran Krishna Innamuri is a Passionate Big Data enthusiast with 2 + years of experience in Hadoop and Spark Development. He is a passionate Java and scala programmer. AcadGild was founded with the vision of "Learn. Do. Earn". We provide skill development courses based on current industry needs. But what sets us apart is earning opportunities we provide after successful completion of course. We also provide live mentoring and 24x7 support. Our mentors are industry thought leaders in their respective fields. We provide courses for Android Programming, Big Data, Front End, Full Stack, AngularJS, NodeJS and Android Programming for children.

□ PREVIOUS ARTICLE

NEXT ARTICLE

**TroubleShoot Hive** and Pig Errors

**Skewed Join in Pig** 

## **RELATED POSTS**







Spark Use Case -**Uber Data Analysis** 

May 16, 2016

SparkSQL with MySQL

May 12, 2016

**Integrating** 

**Analyzing New York Crime Data Using SparkSQL** 

April 28, 2016

LEAVE A REPLY	
COMMENTS *	
NAME *	EMAIL *
WEBSITE	
SUBMIT	
	OW LID COMMENTS BY EMAIL
NOTIFY ME OF FOL	OW-OF COMMENTS BY EMAIL.
NOTIFY ME OF FOL	

**CATEGORIES TAGS**  **LIKE WHAT YOU SEE?** 

5/17	/2016	Spark Use Case	– Travel Data Analys
•	AcadGild	ANDORID PROFILING TOOLS	SUBSCRIBE BLOG
	Android For Kids	ANDROID APP FOR SPEECH TO TEXT	We send or week
	AngularJS	ANDROID APP FOR TEXT TO SPEECH	Enter you
•	Big Data and Hadoop	ANDROID DEVELOPMENT	Suk
•	Careers		
	Cloud computing	ANDROID MEMORY ANALYZER	
•	Database	ANDROID MEMORY MANAGEMENT	
•	Digital Marketing	BANGALORE SUMMER	
•	Front End	CAMP	
٠	Full Stack	BEST SUMMER CAMPS 2016	
•	Hadoop Administration	BIG DATA DEVELOPEMENT	
	IOS	COMMISSIONING AND	
	Java	DECOMMISSIONING OF DATANODE IN HADOOP	
•	Kids	DEPENDENCY	
	Linux Administration	INJECTION	
	NodeJS	DIFFERENCE BETWEEN ANDROID VS IOS	
	Others	FEATURES OF DDMS	
	Python	FILE FORMATS	
	Quiz	FILE FORMATS IN HADOOP	
•	R & Machine Learning	HADOOP	
•	Scala	HADOOP ADMINISTRATION	
	Spark	Прес	I

only 1 email in a

ur email...

bscribe

HDFS

Uncategorized

HIVE WITH MYSQL

INTRODUCTION TO SPARK

JAVASCRIPT MVC FRAMEWORK

JOB OPPORTUNITIES IN HADOOP

JOB TRENDS IN BIG DATA BLOG

JOB TRENDS IN HADOOP

LINUX

LINUX BASIC

LINUX BASIC COMMANDS

LINUX COMMANDS

MYSQL

MYSQL-CONNECTOR-JAVA-5.1.2.JAR

MYSQL-CONNECTOR-JAVA.JAR

MYSQL WITH HIVE

MYSQL WITH SQOOP

**PYTHON** 

**RACK AWARENESS** 

**RECYCLE BIN** 

RESILIENT
DISTRIBUTED
DATASET (RDD)

SPARK SQOOP

SQOOP WITH MYSQL

STYLING A RESPONSIVE WEB PAGE

SUMMER CAMP

SUMMER CAMP 2016

TOP 10 RECORDS IN MAPREDUCE

TRASH CONFIGURATION

© Copyright 2016. ACADGILD.