Spark Use Case – Uber Data Analysis.

The Uber dataset consists of 4 columns. They are dispatching_base_number, date, active_vehicles and trips.

Problem Statement:

Find the days on which each basement has more trips.

Source Code:

```
val dataset = sc.textFile("file:/home/cloudera/uber_data")

val header = dataset.first()

val format = new java.text.SimpleDateFormat("MM/dd/yyyy")

var days =Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")

val eliminate = dataset.filter(line => line != header)

val split = eliminate.map(line => line.split(",")).map { x =>
(x(0),format.parse(x(1)),x(3)) }

val combine = split.map(x => (x._1+" "+days(x._2.getDay),x._3.toInt))

val arrange = combine.reduceByKey(_+_).map(item =>
item.swap).sortByKey(false).collect.foreach(println)
```

Here's the explanation of the above code:

- In **line 1,** we are loading the dataset in our local system, using the textFile method.

- In **line 2,** we are creating a variable **header,** which holds the first line of the dataset (In this dataset the first line is header line).
- In **line 3,** we are declaring the date format using SimpleDateFormat in Java. Here the date is in the format of MM/dd/YYYY.
- In **line 4,** we are declaring an array, which will hold the days of the week from Sunday to Saturday.
- In **line 5,** we are filtering the header line from the dataset using the filter RDD.

- In **line 6,** we are splitting the dataset using the delimiter as ***coma and*** taking out the three columns; *dispatching_base_number,* which is in the 1st column, date which is in the second column and the *number of trips,* which is in the fourth column. While extracting the columns, we are parsing the date, which is in string format to date format.
- After this step, we will get the records as *B02512, Thu Jan 01 00:00:00 IST 2015, 1132.*
- In **line 7,** we are adding the two columns, *dispatching_base_number* and *formatted date.* To get the *day* from the formatted date, we need to use the **getDay** method of java.util.Date package. Here, we will get the day number of the week and pass the day number into the array consisting of the names of the days. Finally, we will get the combination of *dispatching_base_number* and *day* of the week and the number of weeks. These are like keys and values.
- In **line 8,** we are using the reduceByKey RDD to combine all the values for each unique key, where key is the combination of *dispatching_base_number* and *day* of the week. After this, we are swapping the keys and values and

then perform sortByKey action on the RDD, which will sort the records by values in the descending order. Finally, we are printing the result using the **collect** action.

Output:

(356789,B02764 Sat)
(326968,B02764 Fri)
(304200,B02764 Thu)
(249896,B02764 Sun)
(241137,B02764 Wed)
(221343,B02764 Tue)
(214116,B02764 Mon)
(127902,B02617 Sat)
(125067,B02617 Fri)
(120283,B02682 Sat)
(118254,B02617 Thu)
(114662,B02682 Fri)
(106643,B02682 Thu)
(94887,B02617 Wed)
(94588,B02598 Sat)
(93126,B02598 Fri)
(91722,B02617 Sun)
(90333,B02598 Thu)
(86602,B02617 Tue)
(86252,B02682 Wed)
(82825,B02682 Sun)
(80591,B02617 Mon)
(76905,B02682 Tue)
(74939,B02682 Mon)
(71956,B02598 Wed)
(66477,B02598 Sun)
(63429,B02598 Tue)
(60882,B02598 Mon)

(36737,B02765 Sat)
(34934,B02765 Fri)
(30408,B02765 Thu)
(24340,B02765 Wed)
(22741,B02765 Tue)
(22536,B02765 Sun)
(21974,B02765 Mon)
(16435,B02512 Fri)
(15809,B02512 Thu)
(15026,B02512 Sat)
(12691,B02512 Wed)
(12041,B02512 Tue)
(11297,B02512 Mon)
(10487,B02512 Sun)