Problem Statement 1

Find out the top 5 most visited destinations.

Solution-

Below is the code used to find the result-

- val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
- val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
- val mapping = removeHeader.map(x => x.split(",")).map(x => (x(18),1)).filter(x => x._1!=null).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(5)

We know that the input file contains header so we need to remove the header also. So first we are creating a RDD which will read the file and then we are creating another RDD removeHeader which will remove the header from the file. Below screenshot shows the same-

```
scala> val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-21/DelayedFlights.csv MapPartitionsRDD[36] at textFile at <console>:55

scala> val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
removeHeader: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at mapPartitionsWithIndex at <console>:57

scala> removeHeader.take(5).foreach(println)
0,2008,1,3,4,2003.0,1955,2211.0,2225,WN,335,N712SW,128.0,150.0,116.0,-14.0,8.0,IAD,TPA,810,4.0,8.0,0,N,0,,,,,
1,2008,1,3,4,754.0,735,1002.0,1000,WN,3231,N772SW,128.0,145.0,113.0,2.0,19.0,IAD,TPA,810,5.0,10.0,0,N,0,,,,,
2,2008,1,3,4,628.0,620,804.0,750,WN,448,N428WN,96.0,90.0,76.0,14.0,8.0,IND,BWI,515,3.0,17.0,0,N,0,,,,,
4,2008,1,3,4,1829.0,1755,1959.0,1925,WN,3920,N464WN,90.0,90.0,77.0,34.0,34.0,IND,BWI,515,3.0,10.0,0,N,0,2.0,0.0,0.0,0.0,0.0,32.0
5,2008,1,3,4,1940.0,1915,2121.0,2110,WN,378,N726SW,101.0,115.0,87.0,11.0,25.0,IND,JAX,688,4.0,10.0,0,N,0,,,,,

scala>
```

Now we are splitting the file with delimiter as “,” then we are extracting the destination column and mapping it with 1. After that we are counting all occurrences of each destination and sorting it and selecting the top 5 as shown in below screenshot

```
scala> val mapping = removeHeader.map(x => x.split(",")).map(x => (x(18),1)).filter(x => x._1!=null).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x =>
 (x._2,x._1)).take(5)
mapping: Array[(String, Int)] = Array((ORD,108984), (ATL,106898), (DFW,70657), (DEN,63003), (LAX,59969))

scala> mapping.foreach(println)
(ORD,108984)
(ATL,106898)
(DFW,70657)
(DEN,63003)
(LAX,59969)

scala>
```

Problem Statement 2

Which month has seen the most number of cancellations due to bad weather?

Solution-

Below is the code used to find the result-

- val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
- val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
- val canceled = removeHeader.map(x => x.split(",")).filter(x => ((x(22).equals("1"))&&(x(23).equals("B")))).map(x => (x(2),1)).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(1)

  We know that the input file contains header so we need to remove the header also. So first we are creating a RDD which will read the file and then we are creating another RDD removeHeader which will remove the header from the file. Below screenshot shows the same-

```
scala> val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-21/DelayedFlights.csv MapPartitionsRDD[36] at textFile at <console>:55

scala> val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
removeHeader: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at mapPartitionsWithIndex at <console>:57

scala> removeHeader.take(5).foreach(println)
0,2008,1,3,4,2003.0,1955,2211.0,2225,WN,335,N712SW,128.0,150.0,116.0,-14.0,8.0,IAD,TPA,810,4.0,8.0,0,N,0,,,,,
1,2008,1,3,4,754.0,735,1002.0,1000,WN,3231,N772SW,128.0,145.0,113.0,2.0,19.0,IAD,TPA,810,5.0,10.0,0,N,0,,,,,
2,2008,1,3,4,628.0,620,804.0,750,WN,448,N428WN,96.0,90.0,76.0,14.0,8.0,IND,BWI,515,3.0,17.0,0,N,0,,,,,
4,2008,1,3,4,1829.0,1755,1959.0,1925,WN,3920,N464WN,90.0,90.0,77.0,34.0,34.0,IND,BWI,515,3.0,10.0,0,N,0,2.0,0.0,0.0,0.0,32.0
5,2008,1,3,4,1940.0,1915,2121.0,2110,WN,378,N726SW,101.0,115.0,87.0,11.0,25.0,IND,JAX,688,4.0,10.0,0,N,0,,,,,

scala>
```

Now here we are splitting the file based on delimiter “,” and filtering the CANCEL column if its value is 1. Then we are mapping it with 1 and counting its occurrences.

Then we are sorting it and taking the top most value to get the required result-

```
scala> val canceled = removeHeader.map(x => x.split(",")).filter(x => ((x(22).equals("1"))&&(x(23).equals("B")))).map(x => (x(2),1)).reduceByKey(_+_).map(x => (x._2,
x._1)).sortByKey(false).map(x => (x._2,x._1)).take(1)
canceled: Array[(String, Int)] = Array((12,250))

scala> canceled.foreach(println)
(12,250)

scala>
```

Problem Statement 3

Top ten origins with the highest AVG departure delay

Solution-

Below is the code used to find the result-

- val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
- val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
- val avg = removeHeader.map(x => x.split(",")).map(x => (x(17),x(16).toDouble)).mapValues((_, 1)).reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2)).mapValues{ case (sum, count) => (1.0 * sum)/count}.map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(10)

  We know that the input file contains header so we need to remove the header also. So first we are creating a RDD which will read the file and then we are creating another RDD removeHeader which will remove the header from the file. Below screenshot shows the same-

```
scala> val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-21/DelayedFlights.csv MapPartitionsRDD[36] at textFile at <console>:55

scala> val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
removeHeader: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at mapPartitionsWithIndex at <console>:57

scala> removeHeader.take(5).foreach(println)
0,2008,1,3,4,2003.0,1955,2211.0,2225,WN,335,N712SW,128.0,150.0,116.0,-14.0,8.0,IAD,TPA,810,4.0,8.0,0,N,0,,,,,
1,2008,1,3,4,754.0,735,1002.0,1000,WN,3231,N772SW,128.0,145.0,113.0,2.0,19.0,IAD,TPA,810,5.0,10.0,0,N,0,,,,,
2,2008,1,3,4,628.0,620,804.0,750,WN,448,N428WN,96.0,90.0,76.0,14.0,8.0,IND,BWI,515,3.0,17.0,0,N,0,,,,,
4,2008,1,3,4,1829.0,1755,1959.0,1925,WN,3920,N464WN,90.0,90.0,77.0,34.0,34.0,IND,BWI,515,3.0,10.0,0,N,0,2.0,0.0,0.0,0.0,32.0
5,2008,1,3,4,1940.0,1915,2121.0,2110,WN,378,N726SW,101.0,115.0,87.0,11.0,25.0,IND,JAX,688,4.0,10.0,0,N,0,,,,,

scala>
```

Now we are splitting the file with delimiter as “,” and extracting column Origin and departure delay and mapping it with 1. Now using reducebyKey we are adding their number of occurrences and after that we are calculating average for delay.

Finally we are selecting top 10 records to get the required result. Below screenshot shows the same-

```
scala> val avg = removeHeader.map(x => x.split(",")).map(x => (x(17),x(16).toDouble)).mapValues((_, 1)).reduceByKey((x, y) => (x._1 + y._1, x._2 + y._2)).mapValues{
case (sum, count) => (1.0 * sum)/count}.map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(10)
avg: Array[(String, Double)] = Array((CMX,116.1470588235294), (PLN,93.76190476190476), (SPI,83.84873949579831), (ALO,82.2258064516129), (MQT,79.55665024630542), (ACY
,79.3103448275862), (MOT,78.66165413533835), (HHH,76.53005464480874), (EGE,74.12891986062718), (BGM,73.15533980582525))

scala> avg.foreach(println)
(CMX,116.1470588235294)
(PLN,93.76190476190476)
(SPI,83.84873949579831)
(ALO,82.2258064516129)
(MQT,79.55665024630542)
(ACY,79.3103448275862)
(MOT,78.66165413533835)
(HHH,76.53005464480874)
(EGE,74.12891986062718)
(BGM,73.15533980582525)
```

Problem Statement 4

Which route (origin & destination) has seen the maximum diversion?

Solution-

Below is the code used to find the result-

- val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
- val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
- val diversion = removeHeader.map(x => x.split(",")).filter(x => ((x(24).equals("1")))).map(x => ((x(17)+","+x(18)),1)).reduceByKey(_+_).map(x => (x._2,x._1)).sortByKey(false).map(x => (x._2,x._1)).take(10).foreach(println)

  We know that the input file contains header so we need to remove the header also. So first we are creating a RDD which will read the file and then we are creating another RDD removeHeader which will remove the header from the file. Below screenshot shows the same-

```
scala> val delayed_flights = sc.textFile("/home/acadgild/Assignment-21/DelayedFlights.csv")
delayed_flights: org.apache.spark.rdd.RDD[String] = /home/acadgild/Assignment-21/DelayedFlights.csv MapPartitionsRDD[36] at textFile at <console>:55

scala> val removeHeader = delayed_flights.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
removeHeader: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at mapPartitionsWithIndex at <console>:57

scala> removeHeader.take(5).foreach(println)
0,2008,1,3,4,2003.0,1955,2211.0,2225,WN,335,N712SW,128.0,150.0,116.0,-14.0,8.0,IAD,TPA,810,4.0,8.0,0,N,0,,,,,
1,2008,1,3,4,754.0,735,1002.0,1000,WN,3231,N772SW,128.0,145.0,113.0,2.0,19.0,IAD,TPA,810,5.0,10.0,0,N,0,,,,,
2,2008,1,3,4,628.0,620,804.0,750,WN,448,N428WN,96.0,90.0,76.0,14.0,8.0,IND,BWI,515,3.0,17.0,0,N,0,,,,,
4,2008,1,3,4,1829.0,1755,1959.0,1925,WN,3920,N464WN,90.0,90.0,77.0,34.0,34.0,IND,BWI,515,3.0,10.0,0,N,0,2.0,0.0,0.0,0.0,32.0
5,2008,1,3,4,1940.0,1915,2121.0,2110,WN,378,N726SW,101.0,115.0,87.0,11.0,25.0,IND,JAX,688,4.0,10.0,0,N,0,,,,,

scala>
```

Now we are splitting the file with delimiter as “,” and filtering the Diverted column whose value is 1. Then we are selecting Source and destination as key value and taking a count of occurrences for diversion with source and destination as key value pair.

Finally we are taking the top 10 of the list to get the required output as shown below-

```
scala> val diversion = removeHeader.map(x => x.split(",")).filter(x => ((x(24).equals("1")))).map(x => ((x(17)+","+x(18)),1)).reduceByKey(_+_).map(x => (x._2,x._1)).
sortByKey(false).map(x => (x._2,x._1)).take(10).foreach(println)
(ORD,LGA,39)
(DAL,HOU,35)
(DFW,LGA,33)
(ATL,LGA,32)
(SLC,SUN,31)
(ORD,SNA,31)
(MIA,LGA,31)
(BUR,JFK,29)
(HRL,HOU,28)
(BUR,DFW,25)
diversion: Unit = ()

scala>
```