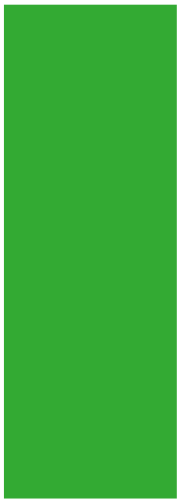
SF_Restaurants_Food_Safety

```
val inspections_raw = sc.textFile("/SFResturantdata/inspections_plus.tx
```

MapPartitionsRDD[1] at textFile at <console>:45 650 milliseconds

Inspection scores Distribution



```
val Inspection_scores = inspections_raw.map(line => line.split("\t").ma
Inspection_scores.foreach(println)
```

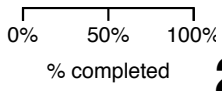
1 second 282 milliseconds

Risk category Distribution

```
val risks_raw = sc.textFile("/SFResturantdata/violations_plus.txt")
val risk_category = risks_raw.map(line => line.split("\t").map(_.trim))

//risk_category.take(10).foreach(println)
risk_category.foreach(println)
```

753 milliseconds



SparkUI (http://192.168.99.1:4041)

20 businesses which got lowest scores

"business_id","name","address","city","postal_code","latitude","longitude", score

```
val lowest_score_business = inspections_raw.map(line => line.split("\t"
val bus20rdd = sc.parallelize(lowest_score_business.take(20))
//bus20rdd.foreach(println)

//val bussiness = sc.textFile("/Users/sbapodara/Trainings/UCSC/Apache_S
val bussiness = sc.textFile("/SFResturantdata/businesses_plus.txt").map

val bdetail = bus20rdd.join(bussiness)
val bdetail_clean = bdetail.map(r =>  ( r._2._2._1,r._2._2._2,r._2._2._

val bdetail_clean2 =  sc.parallelize(bdetail_clean.take(20), 1)
bdetail_clean2.foreach(println)
```

1 second 991 milliseconds

20 Highest scoring Business

```
val highest_score_business = inspections_raw.map(line => line.split("\t
val highbusdistinct = highest_score_business.distinct().sortBy( _._2.to
val highbus20rdd = sc.parallelize(highbusdistinct.take(20))

val bdetail = highbus20rdd.join(bussiness)
//bdetail.take(20).foreach(println)

val bdetail_clean = bdetail.map(r =>  ( r._2._2._1,r._2._2._2,r._2._2._

val bdetail_clean2 =  sc.parallelize(bdetail_clean.take(20), 1)
bdetail_clean2.foreach(println)
```

1 second 169 milliseconds

violations of Business with 100 score

```
val Inspection_scores_100 = inspections_raw.map(line => line.split("\t"
val a = risks_raw.map(line => line.split("\t").map(_.trim)).map(s=> (s(
val b = Inspection_scores_100.join(a).distinct().map(r => (r._1, r._2._

b.take(20).foreach(println)
```

815 milliseconds

average inspection scores by zipcode

```
val distinct_Inspection_scores = inspections_raw.map(line => line.split
val zip_score = distinct_Inspection_scores.join(bussiness).map(r => ( r
val zip_score_values = zip_score.mapValues(x => (x.toInt, 1)).reduceByK
val average_score = zip_score_values.map(r => (r._1, (r._2._1/r._2._2))
average_score.foreach(println)
```

1 second 240 milliseconds

proportion of all businesses in each neighborhood (zip code) that have incurred at least one of the following violations

"Moderate risk vermin infestation" "Sewage or wastewater contamination" "Improper food labeling or menu misrepresentation" "Contaminated or adulterated food" "Reservice of previously served foods"

```
val business = sc.textFile("/SFResturantdata/businesses_plus.txt").map(
val business_zipcount = business.map(s => ( s._2.trim, 1)).reduceByKey(
val violations = sc.textFile("/SFResturantdata/violations_plus.txt").fi
val zip_violation = violations.join(business).map(x => (x._2._2, 1)).re
val business_violations = business_zipcount.leftOuterJoin(zip_violation
val v = business_violations.map(x => (x._1,(( x._2._2.getOrElse(0).toFl
v.foreach(println)
```

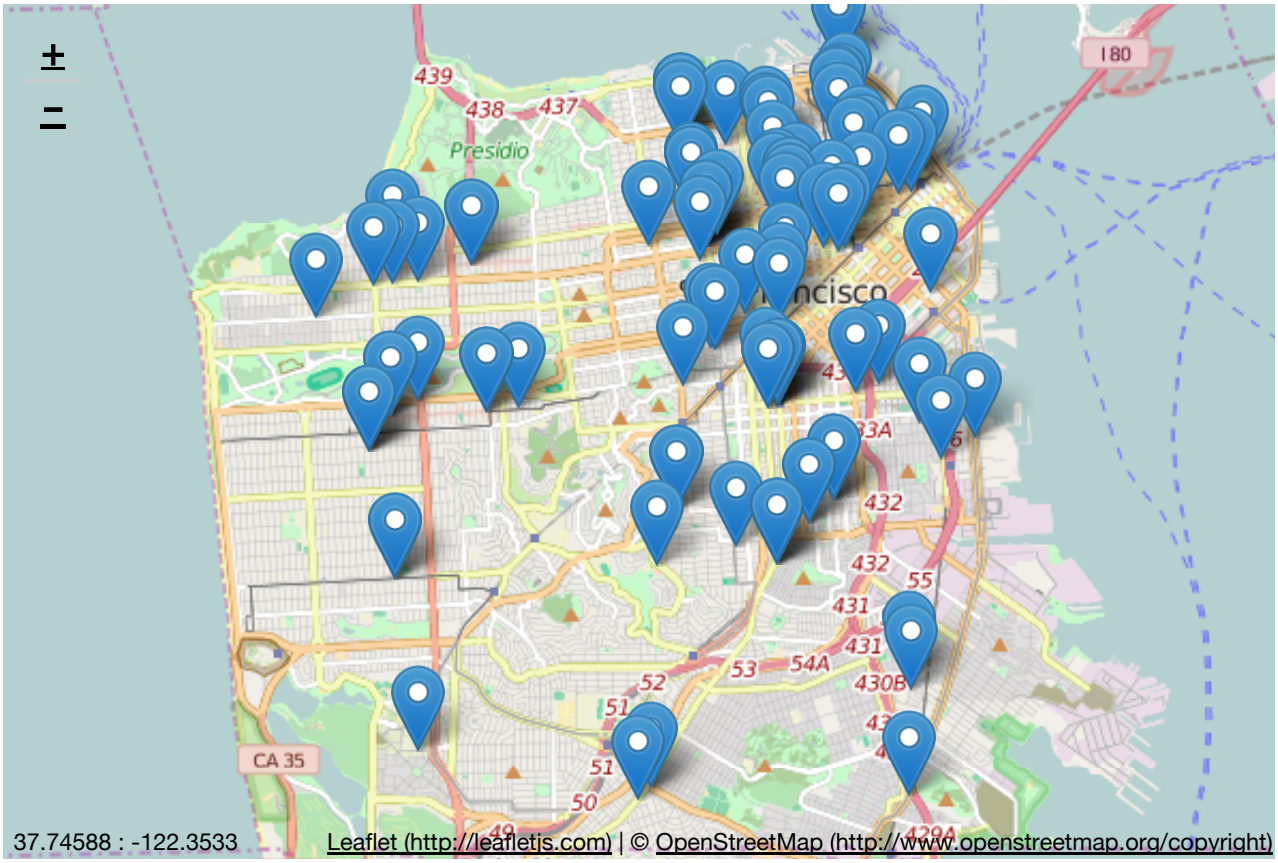
768 milliseconds

Are SF restaurants clean? Justify...

SF resturants are not clean. As it can be seen from above result that more than half of the zipcodes in san francisco includes resturants which have high risk violations, The resturant are not clean and this not safe.

Mapping 100 businesses with serious violations

```
import org.apache.spark.SparkContext._
val Sviolations = sc.textFile("/SFResturantdata/violations_plus.txt").f
val vbusiness = sc.textFile("/SFResturantdata/businesses_plus.txt").map
val bizviolations = Sviolations.join(vbusiness).map(x => (x._2._2._1, x
val a = bizviolations.collect()
val points = Seq((37.795991,-122.421817), (37.784884,-122.409425), (37.
val w = widgets.GeoPointsChart(a, maxPoints=100)
```



974 milliseconds

Build: | **buildTime**-*Sat Jan 09 20:28:53 UTC 2016* | **formattedShaVersion**-*0.6.2-7c7b07797474ce69a7edeee78cd1c1df09bd5730* | **sbtVersion**-*0.13.8* | **scalaVersion**-*2.11.7* | **sparkNotebookVersion**-*0.6.2* | **hadoopVersion**-*2.7.1* | **jets3tVersion**-*0.7.1* | **jlineDef**-*(jline,2.12)* | **sparkVersion**-*1.6.0* | **withHive**-*true* | **withParquet**-*true* |.