Scala and SparkSQL code used for Analytics and loading dataset.

**Scala for loading the dataset:**

import org.apache.commons.io.IOUtils

import java.net.URL

import java.nio.charset.Charset

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)

// So you don't need create them manually

// load bank data

val bankText = sc.textFile("/Users/Shreyams/desktop/cmpe\_273/zeppelin/incubator-zeppelin/datafile/locationOutput1.csv")

case class DeathCause(Year: Integer,ZIP\_Code: Integer, Causes\_of\_Death: String, Count: Integer, Location: String)

val deathCause = bankText.map(s => s.split(",")).filter(s => s(0) != "Year").map(s => DeathCause(s(0).toInt, s(1).toInt, s(2), s(3).toInt, s(4))).toDF()

deathCause.registerTempTable("deathCause")

**SparkSQL Queries:-**

%sql

select Year, count(1) value

from deathCause

where Year < 2013

group by Year

order by Year

%sql

select Location, Count(1) value

from deathCause

group by Location

having Count(1) < ${maxCount(1)=500}

order by Location

%sql

select Causes\_of\_Death, Count(1) value

from deathCause

group by Causes\_of\_Death

order by Causes\_of\_Death

%sql

select ZIP\_Code, Count(1) value

from deathCause

group by ZIP\_Code

having Count(1) < 50

order by ZIP\_Code

%sql

select Year, count(1) value

from deathCause

where Year < 2014

group by Year

order by Year

%sql

select Location, count(1) value

from deathCause

where Causes\_of\_Death = 'SUI'

group by Location

order by Location

%sql

select Location, count(1) value

from deathCause

where Causes\_of\_Death = 'HOM'

group by Location

order by Location

%sql

select Year, count(1) value

from deathCause

where Causes\_of\_Death = 'CAN' and Year = 1999

group by Year

order by Year