import os

import pandas as pd

import numpy as np

from pyspark import SparkConf, SparkContext

from pyspark.sql import SparkSession, SQLContext

from pyspark.sql.types import \*

import pyspark.sql.functions as fn

from pyspark.sql.functions import col, udf

**Reading CSV for each year**

****

year2005 = (spark.read.csv(path='/FileStore/tables/2005\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2006 = (spark.read.csv(path='/FileStore/tables/2006\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2007 = (spark.read.csv(path='/FileStore/tables/2007\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2008 = (spark.read.csv(path='/FileStore/tables/2008\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2009 = (spark.read.csv(path='/FileStore/tables/2009\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2010 = (spark.read.csv(path='/FileStore/tables/2010\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2011 = (spark.read.csv(path='/FileStore/tables/2011\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

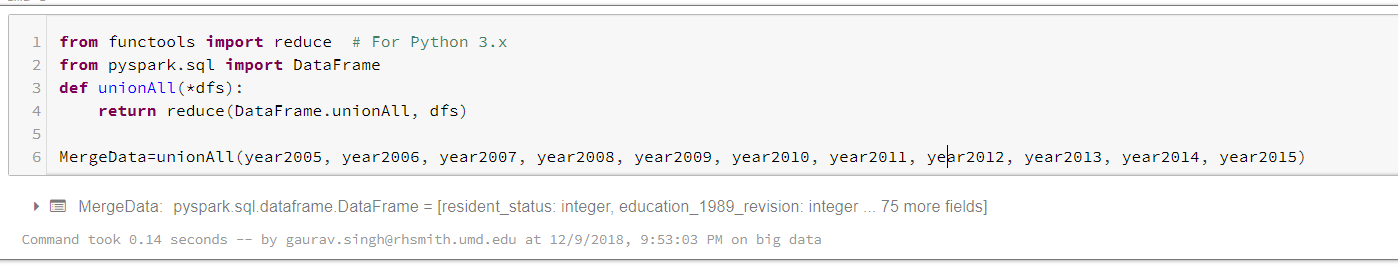
year2012 = (spark.read.csv(path='/FileStore/tables/2012\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2013 = (spark.read.csv(path='/FileStore/tables/2013\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2014 = (spark.read.csv(path='/FileStore/tables/2014\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

year2015 = (spark.read.csv(path='/FileStore/tables/2015\_data.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

**Merging data for all 11 years**



from functools import reduce # For Python 3.x

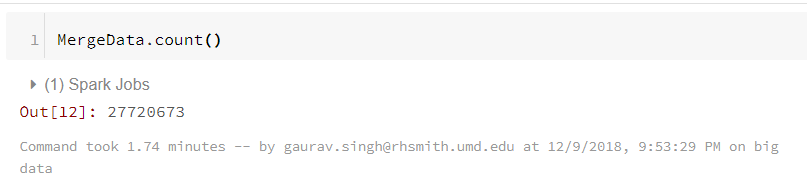
from pyspark.sql import DataFrame

def unionAll(\*dfs):

return reduce(DataFrame.unionAll, dfs)

MergeData=unionAll(year2005, year2006, year2007, year2008, year2009, year2010, year2011, year2012, year2013, year2014, year2015)

Count() for Merged Data

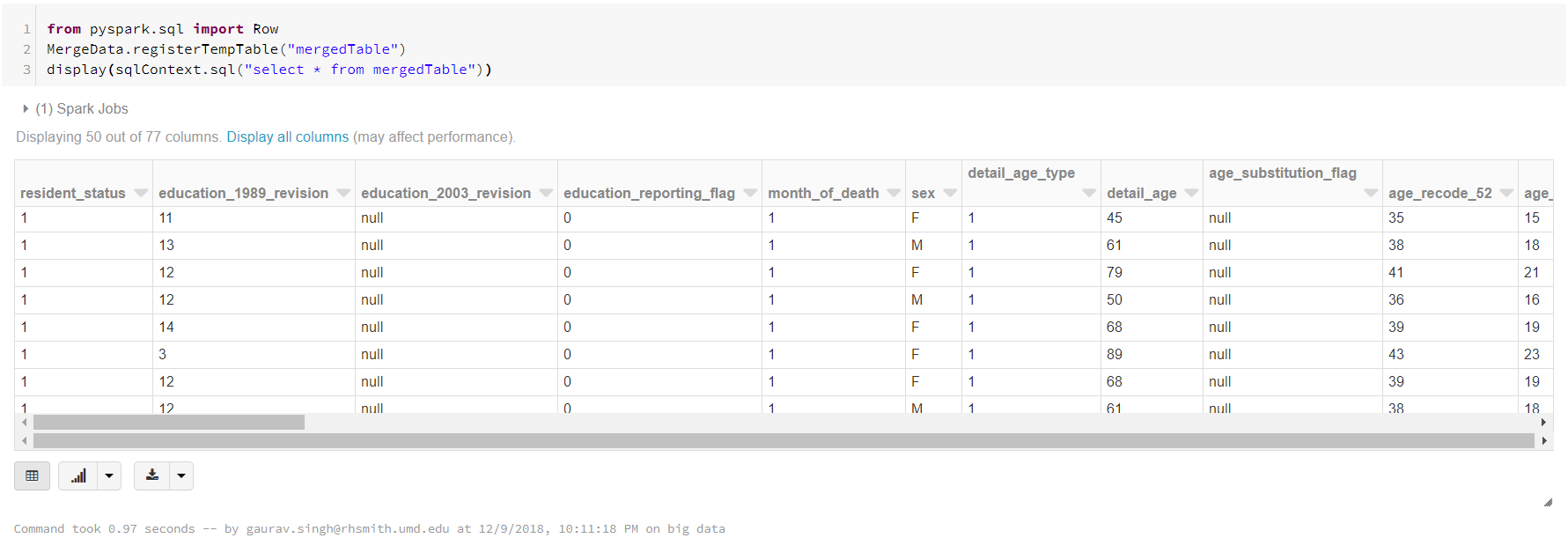


Creating SQL Table for MergeData

from pyspark.sql import Row

MergeData.registerTempTable("mergedTable")

display(sqlContext.sql("select \* from mergedTable"))



Reading ICD10 data from csv

icd10= (spark.read.csv(path='/FileStore/tables/ICD10.csv',header=True,inferSchema=True, ignoreLeadingWhiteSpace=True, ignoreTrailingWhiteSpace=True).cache())

Creating ICD10 SQL table

icd10.registerTempTable("icd10")

display(sqlContext.sql("select \* from icd10"))

SQL query

1. Male vs female deaths by resident status

Resident Status United States

1 ... RESIDENTS State and County of Occurrence and Residence are the same.

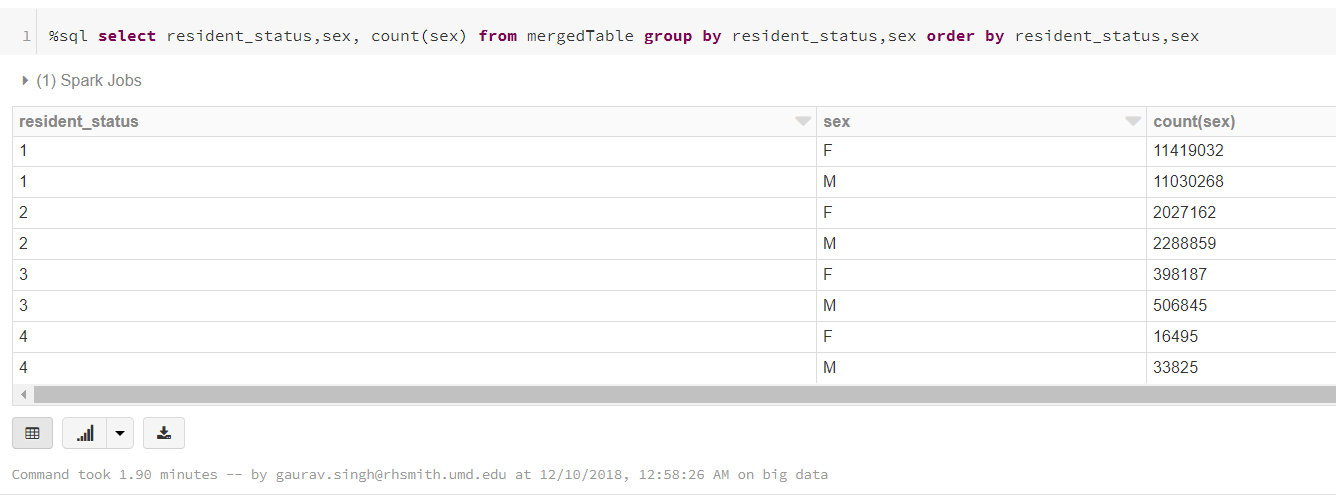
2 ... INTRASTATE NONRESIDENTS State of Occurrence and Residence are the same, but County is different.

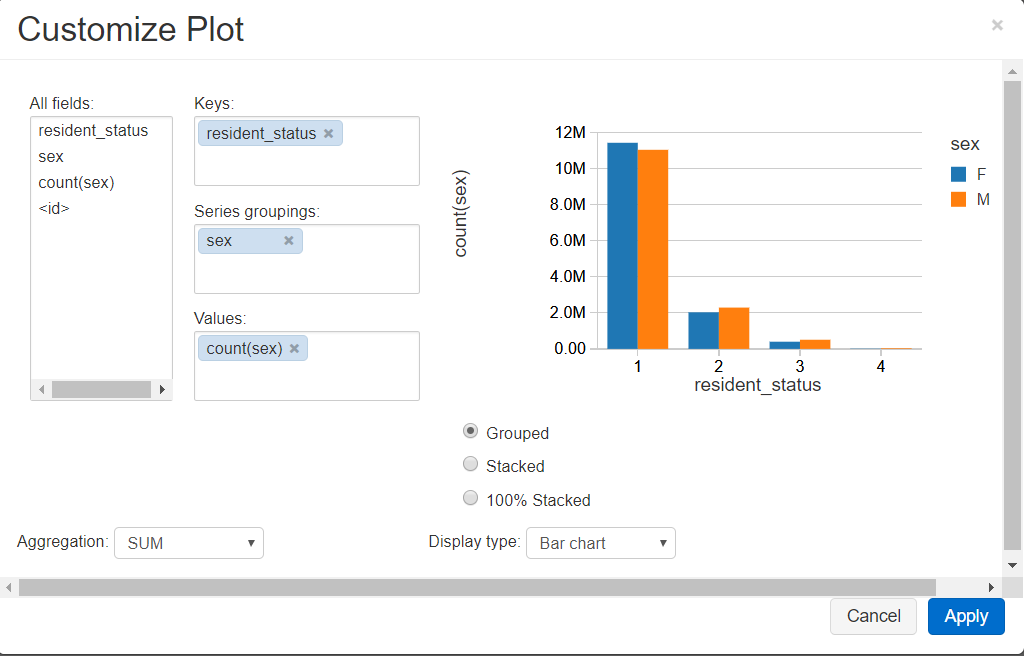
3 ... INTERSTATE NONRESIDENTS State of Occurrence and Residence are different, but both are in the U.S.

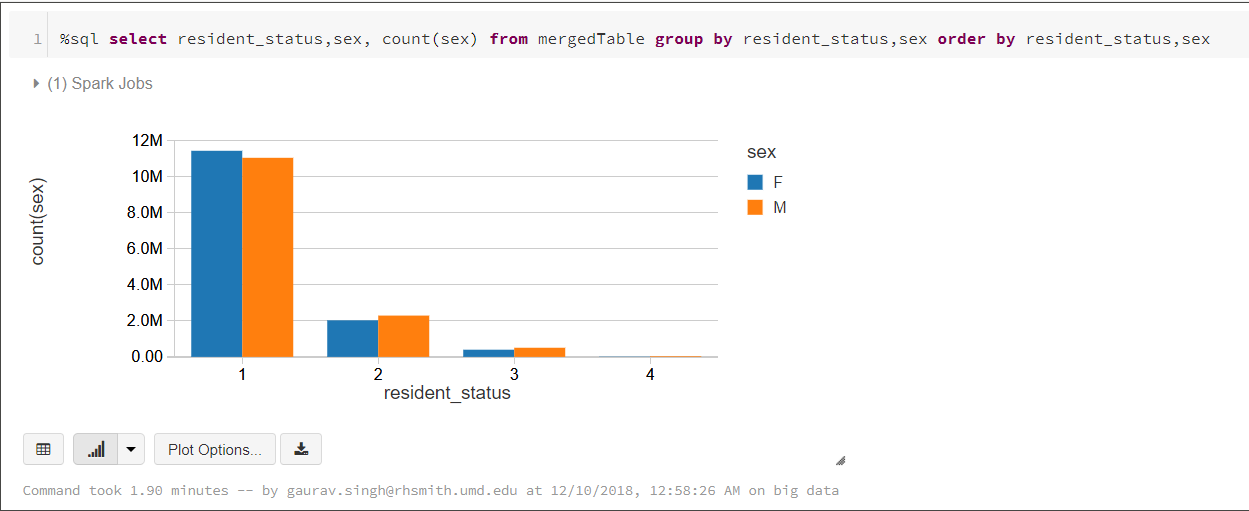
4 ... FOREIGN RESIDENTS State of Occurrence is one of the 50 States or the District of Columbia, but Place of Residence is outside of the U.S.

SQL code:

%sql select resident\_status,sex, count(sex) from mergedTable group by resident\_status,sex order by resident\_status,sex



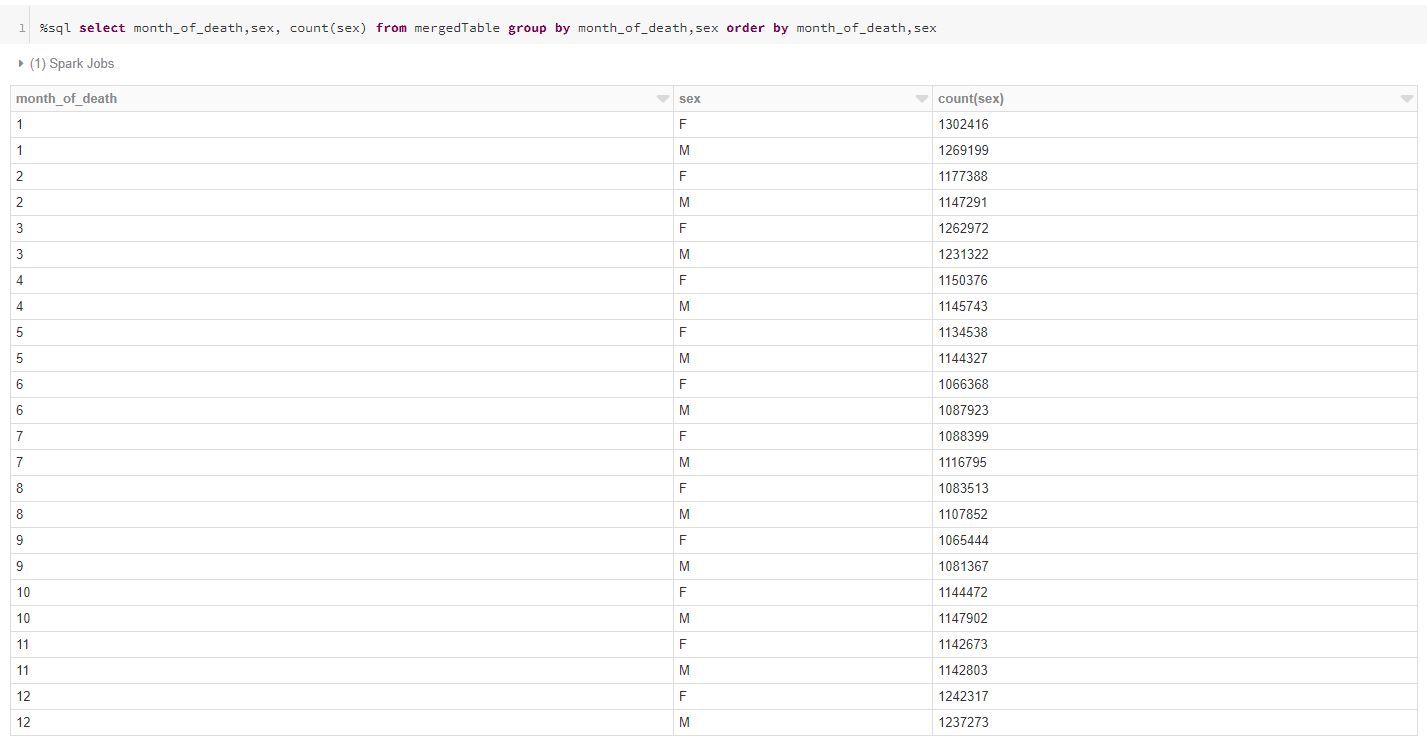


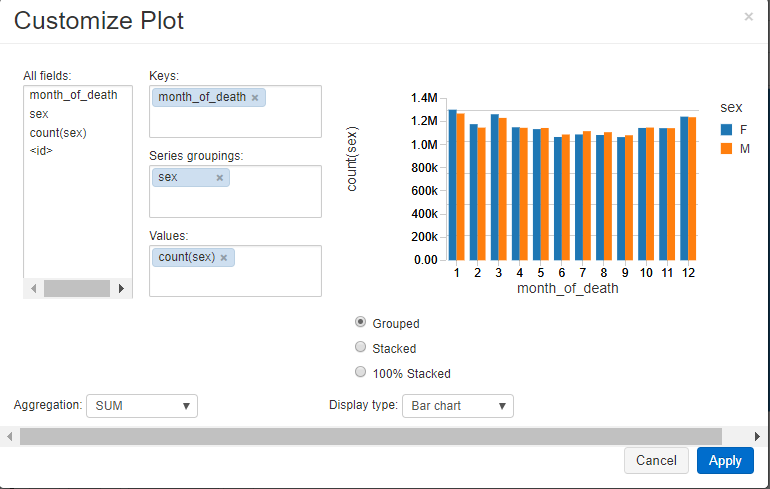


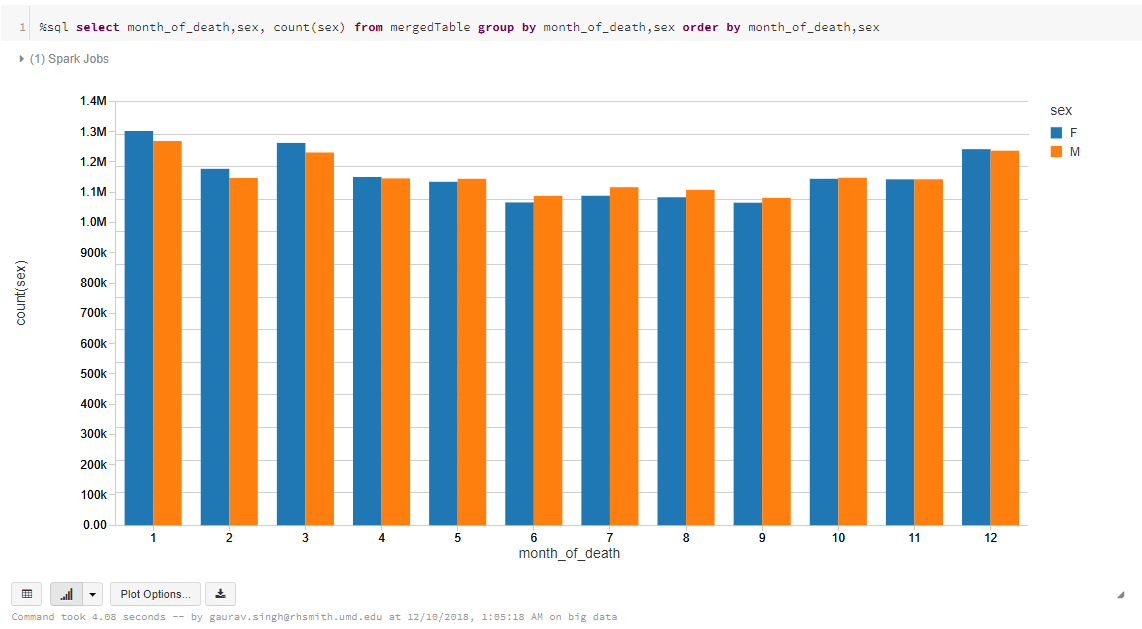
ii) Male vs Female death by month of the year

SQL code:

%sql select month\_of\_death,sex, count(sex) from mergedTable group by month\_of\_death,sex order by month\_of\_death,sex

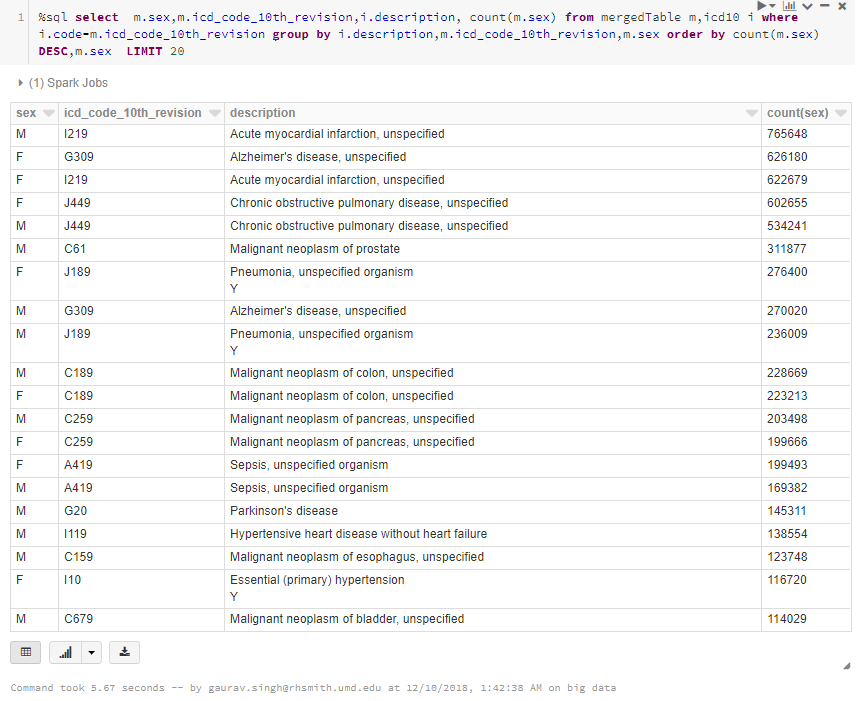


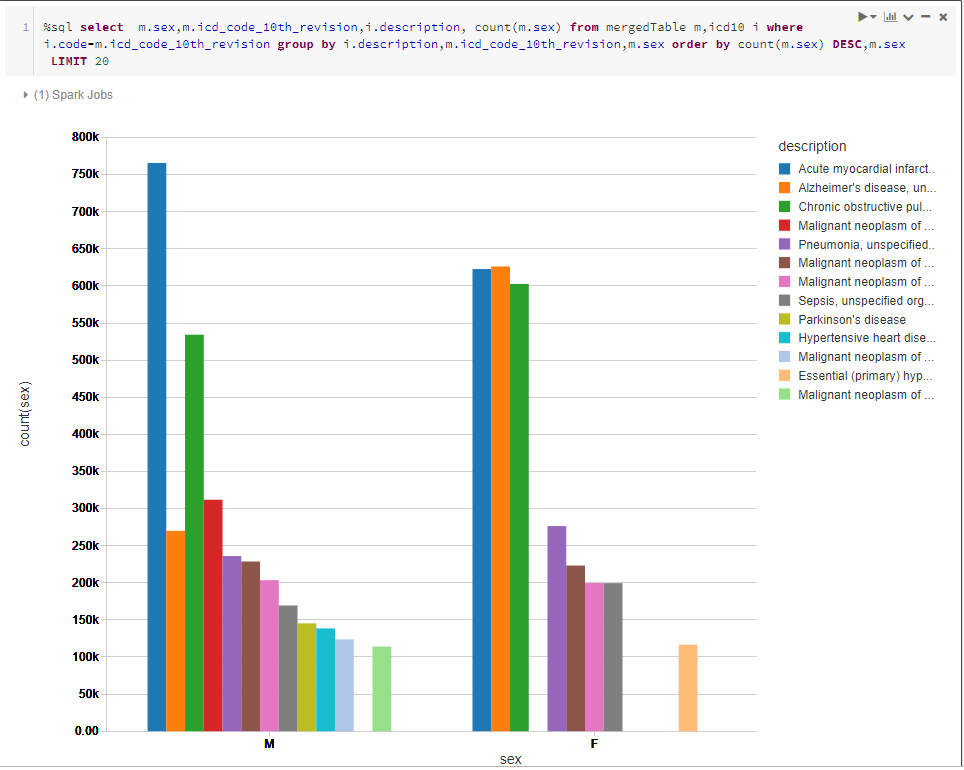




3) Top 20 diseases causing deaths for either men or women

%sql select m.sex,m.icd\_code\_10th\_revision,i.description, count(m.sex) from mergedTable m,icd10 i where i.code=m.icd\_code\_10th\_revision group by i.description,m.icd\_code\_10th\_revision,m.sex order by count(m.sex) DESC,m.sex LIMIT 20



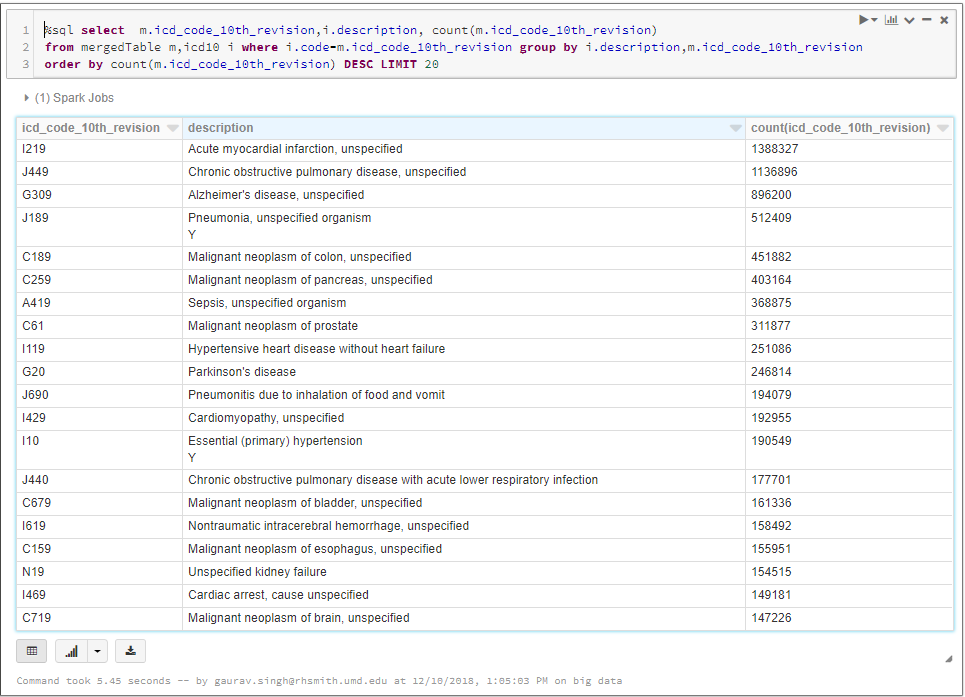


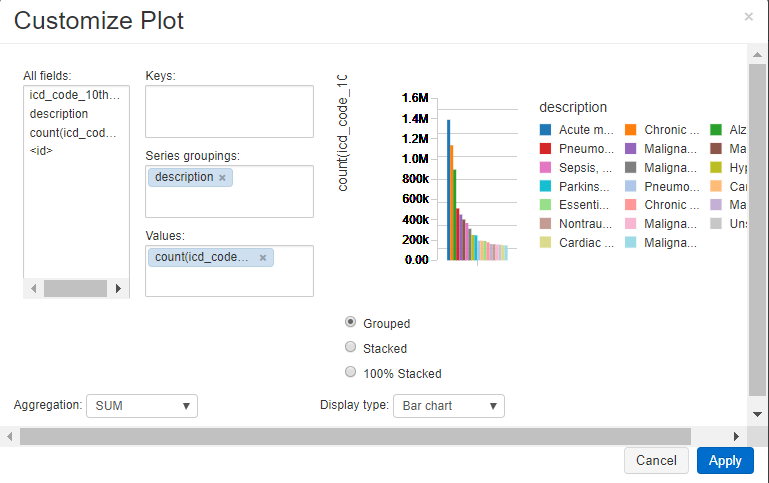
4) Top 20 diseases causing deaths

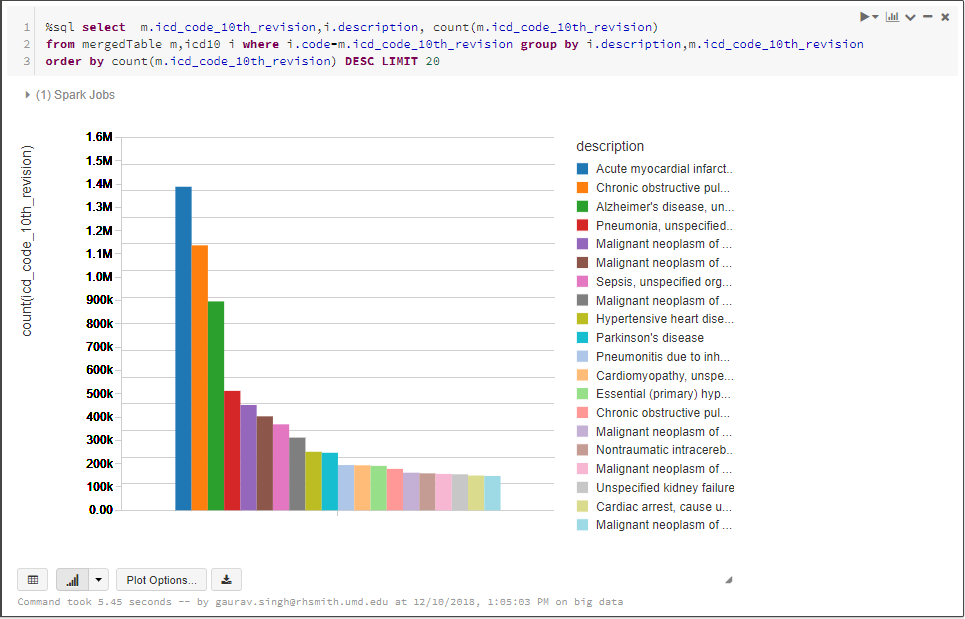
%sql select m.icd\_code\_10th\_revision,i.description, count(m.icd\_code\_10th\_revision)

from mergedTable m,icd10 i where i.code=m.icd\_code\_10th\_revision group by i.description,m.icd\_code\_10th\_revision

order by count(m.icd\_code\_10th\_revision) DESC LIMIT 20

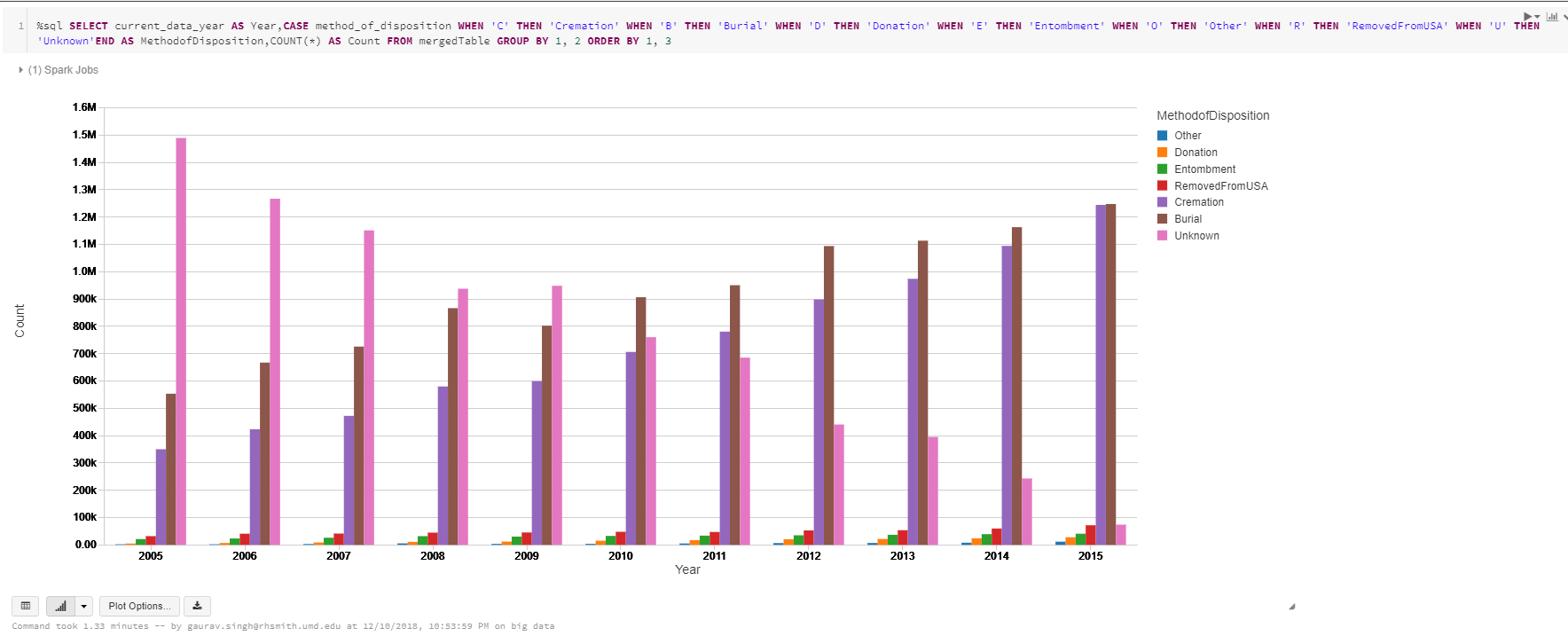


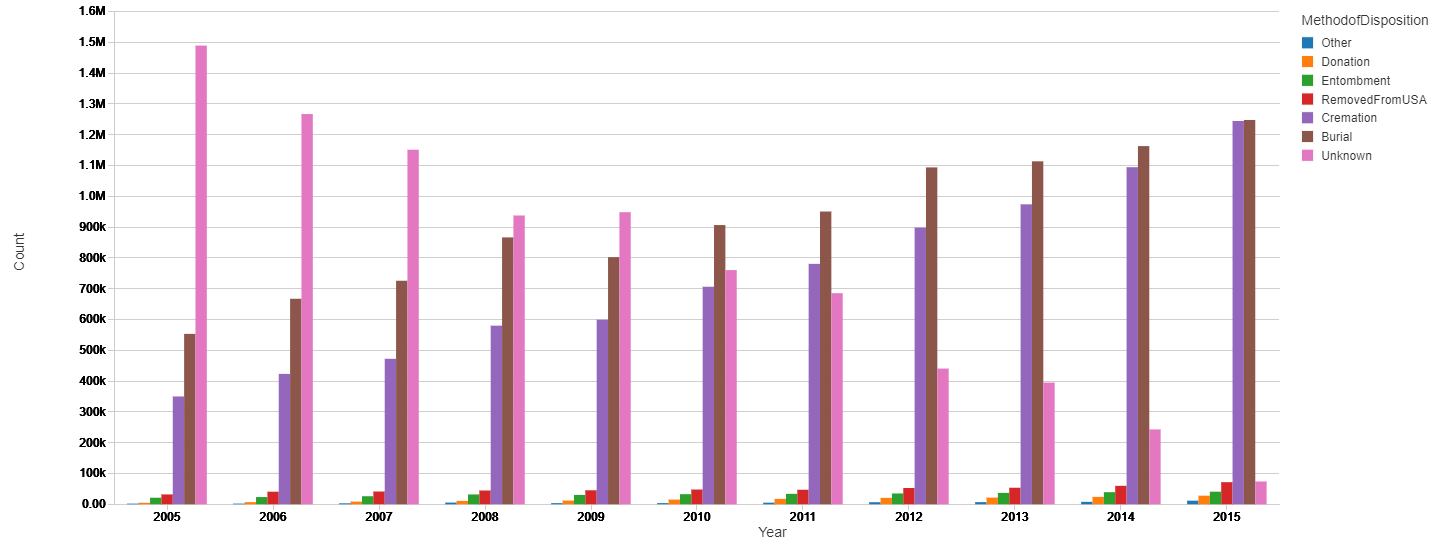




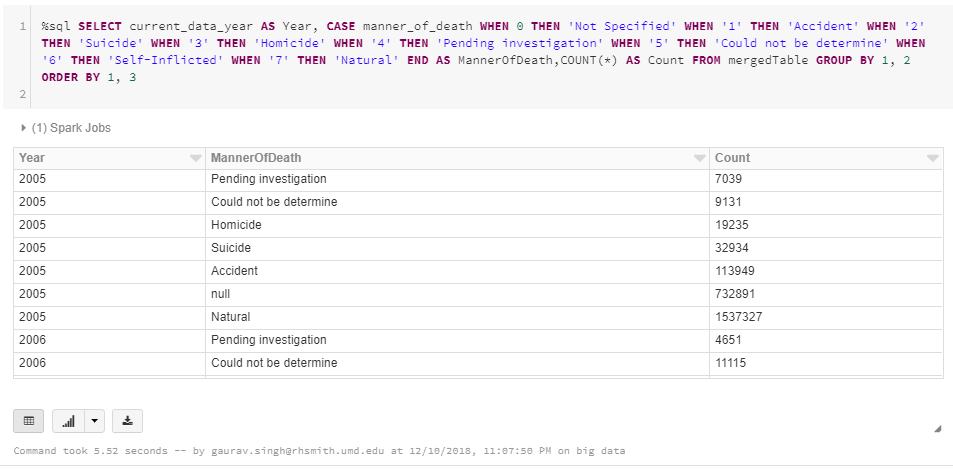
4) Method of disposition trend over 2005-2015

%sql SELECT current\_data\_year AS Year,CASE method\_of\_disposition WHEN 'C' THEN 'Cremation' WHEN 'B' THEN 'Burial' WHEN 'D' THEN 'Donation' WHEN 'E' THEN 'Entombment' WHEN 'O' THEN 'Other' WHEN 'R' THEN 'RemovedFromUSA' WHEN 'U' THEN 'Unknown'END AS MethodofDisposition,COUNT(\*) AS Count FROM mergedTable GROUP BY 1, 2 ORDER BY 1, 3

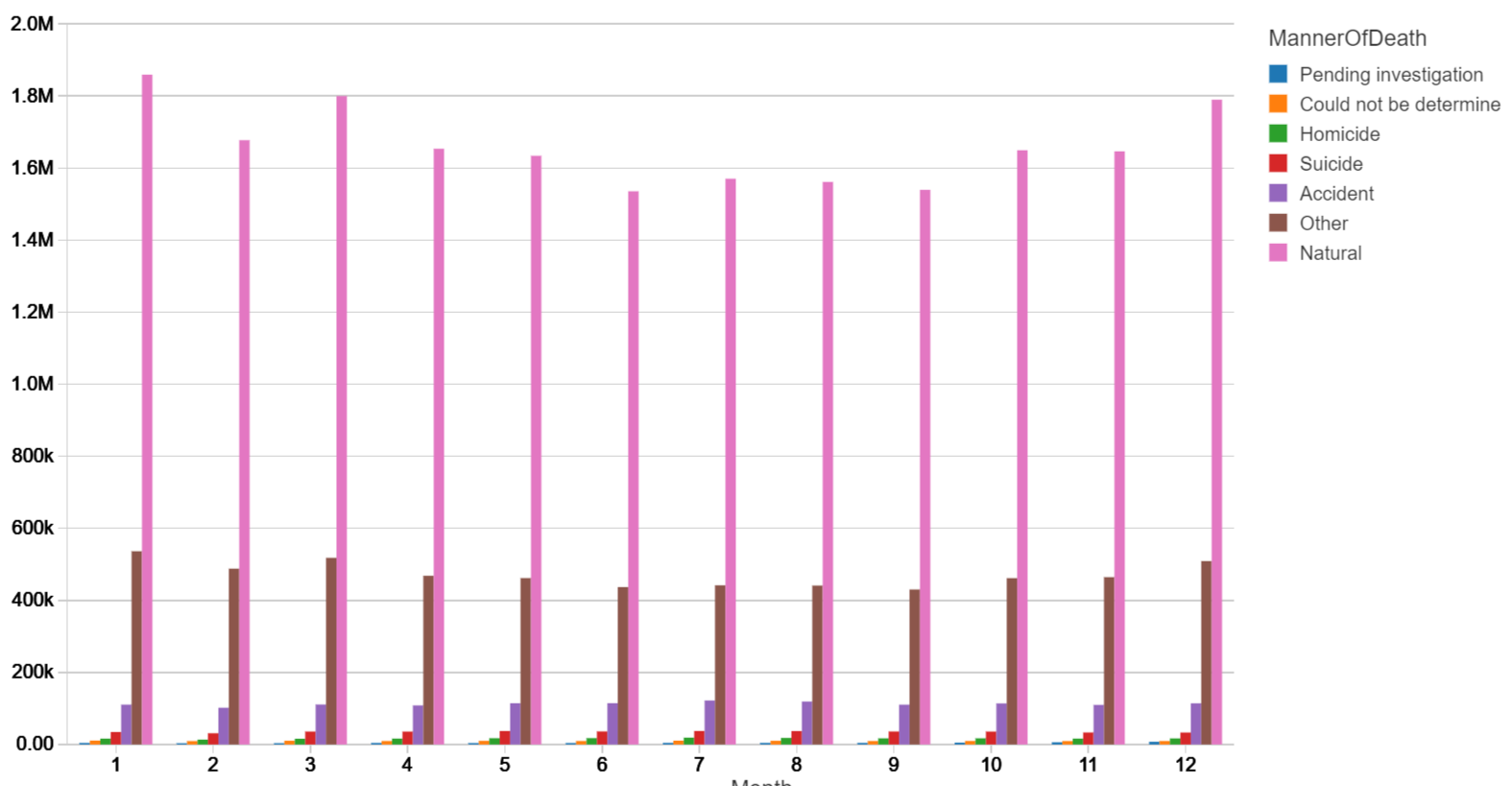




5) Trend of Manner of death from 2005 to 2015



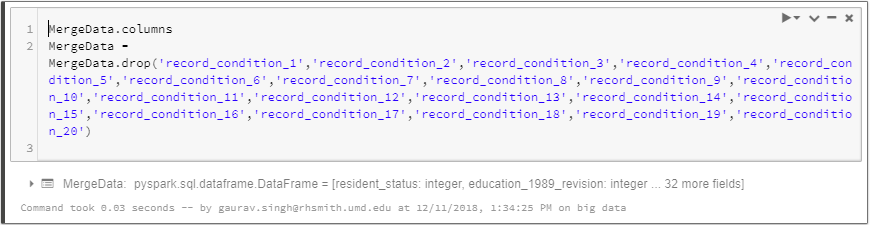
6) Based on month of the year

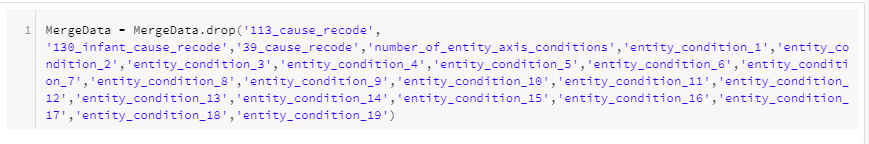


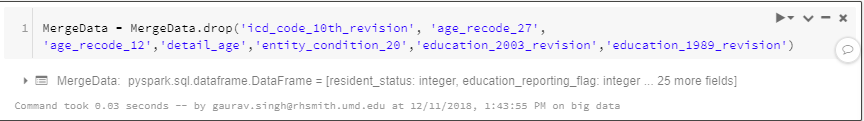
**PREDICTION MODEL:**

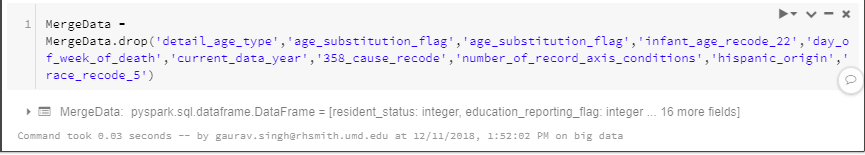
****

Removing unrequired columns from MergeData

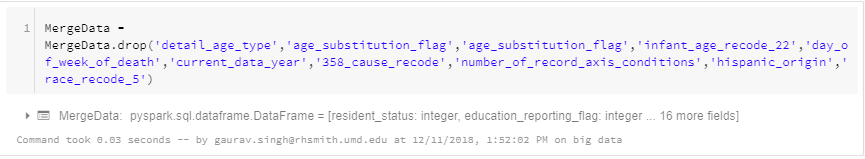




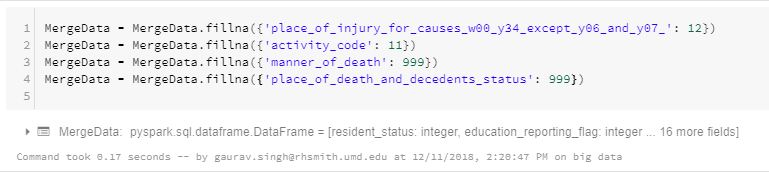




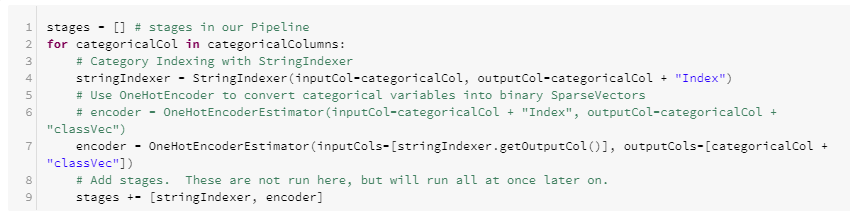
Filter method\_of\_disposition column by burial and cremation (‘B’ and ‘C’)

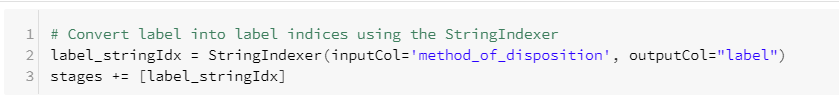


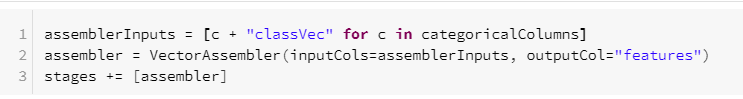
Null imputations- replacing null values

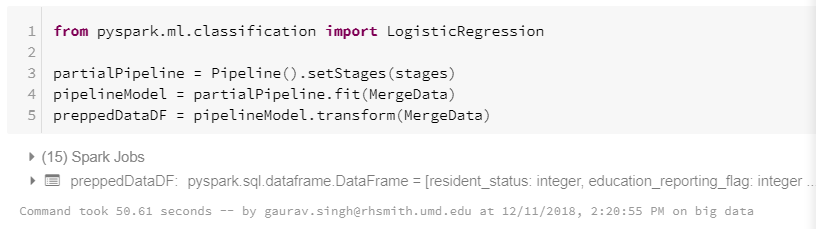
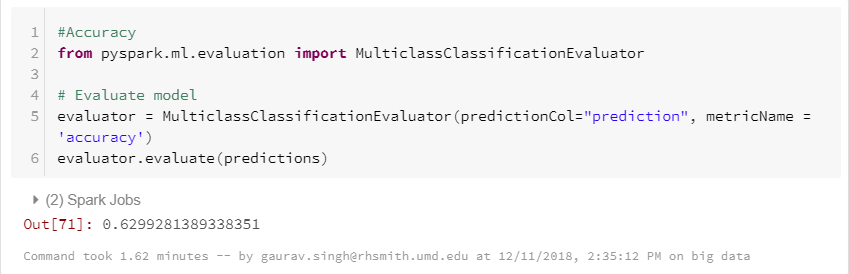


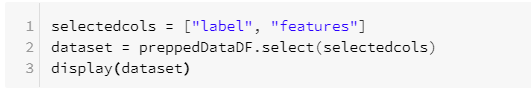
String indexing, creating dummy variables and marking the independent variables as vectors

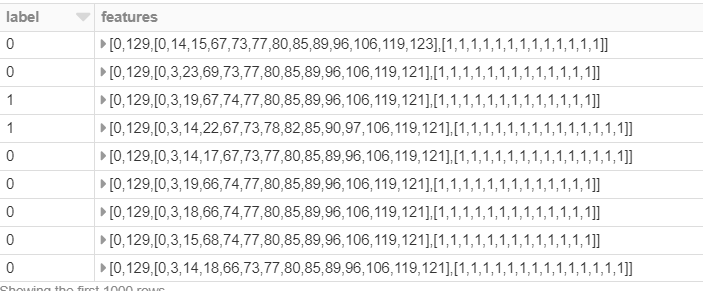












Splitting the dataset into test set and training set









