```
##
from __future__ import division
#
import json
import re
import string
#
from datetime import datetime, timedelta, time
import time
import datetime
#
import numpy as np
from pyspark.mllib.stat import Statistics
#
import sys


###
#import json
#
#from datetime import datetime
#import time
from datetime import datetime, timedelta, time
import time
import datetime
#
import os
import sys


#
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext



#"""
###
conf = SparkConf()
conf.setAppName("PAQA01")
#
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
# Reduce log verbosity
#sc.setLogLevel("WARN")
#ValueError: Cannot run multiple SparkContexts at once; existing SparkContext(app=Zeppelin-DASI, master=yarn-
client) created by __init__ at /tmp/zeppelin_pyspark.py:204
#"""


###
#tm_start = datetime.now()
tm_start = datetime.datetime.now()








#### 1.set email
parms!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

# PARMS: email for when your app runs ok
EmailTo1 = 'mss@biomedupdater.com'
EmailSubject1 = 'CodeQA: OK ran VpnRecordCount'     +' at '+str(tm_start)
EmailBody1 = EmailSubject1
#'0Sat'
EmailDays1 = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']
#'n08'
EmailHours1 = ['00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15',
'16', '17', '18', '19', '20', '21', '22', '23']

# PARMS: email for when your app does not run!
EmailTo0 = 'mss@biomedupdater.com'
EmailSubject0 = 'CodeQA: FATAL exception caught for VpnRecordCount '     +' at '+str(tm_start)
```

```
EmailBody0 = EmailSubject0
#'0Sat'
EmailDays0 = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']
#'n08'
EmailHours0 = ['00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15',
'16', '17', '18', '19', '20', '21', '22', '23']




#### 2.paste your app
here!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# ASSUMEs the name of the subroutine is 'WorkerSub'
def WorkerSubVpn ():
          #UnboundLocalError: local variable 'tm_start' referenced before assignment
          tm_start = datetime.datetime.now()



          ### parms

          # hours into past to use for creating moving standard
          # includes current hour
          nPastHours = 5
          #nPastHours = 13
          #nPastHours = 49


          # PARMS: email for when WorkerSub wants to send WARN
          WorkerSubEmailTo0 = 'mss@biomedupdater.com'
          WorkerSubEmailSubject0 = 'PAQA DATA: WARN;'    +' at '+str(tm_start)
          WorkerSubEmailBody0 = WorkerSubEmailSubject0
          WorkerSubEmailDays0 = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']
          WorkerSubEmailHours0 = ['00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12',
'13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23']

          # PARMS: email for when WorkerSub wants to send OK
          WorkerSubEmailTo1 = 'mss@biomedupdater.com'
          WorkerSubEmailSubject1 = 'PAQA DATA: OK;'    +' at '+str(tm_start)
          WorkerSubEmailBody1 = WorkerSubEmailSubject1
          #'0Sat'
          WorkerSubEmailDays1 = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', '0Sat', '0Sun']
          #'n08'
          WorkerSubEmailHours1 = ['n00', 'n01', 'n02', 'n03', 'n04', '05', 'n06', 'n07', 'n08', 'n09', 'n10',
'n11', 'n12', 'n13', 'n14', 'n15', 'n16', 'n17', 'n18', 'n19', 'n20', 'n21', 'n22', 'n23']
          #











          # to execute on dir with date&time stamp=?
          #datePath = '2017/03/30/12'

          ####
          ##
          currentDT = datetime.datetime.now()
          print currentDT
          #2017-04-24 08:48:49.932382
          print str(currentDT)
          ##
          currentDTParts = re.split(r" +", str(currentDT) )
          print currentDTParts
          #['2017-04-24', '10:42:27.385279']
          #
          currentDParts = re.split(r"\-+", str(currentDTParts[0]) )
          print currentDParts
          #['2017', '04', '24']
          #
          currentTParts = re.split(r":+", str(currentDTParts[1]) )
          print currentTParts
          #['10', '43', '14.189765']
          ## construct path
          HourPrior = int(currentTParts[0])-1
          if (HourPrior<0) :
              HourPrior=0
          HourPrior=str(HourPrior)
          print HourPrior
          ##
          PathDateHour = [ currentDParts[0], currentDParts[1], currentDParts[2], HourPrior]
          print PathDateHour
          #['2017', '04', '24', '10']
          ##
          ParmPathDateHour = '/'.join(PathDateHour)
          print ParmPathDateHour
          #2017/04/24/9
          datePath = ParmPathDateHour



          ##
          tm_start = datetime.datetime.now() ; print tm_start
          #
          CumMsg = str(tm_start)

          #
          #print 'datePath:', datePath
          TmpMsg = str('datePath: '+ datePath)
          print TmpMsg
          #
          CumMsg = CumMsg+"\n"+  str(TmpMsg)



          ### create datetime obj given datePath = '2017/03/30/12'
          # ASSUMEs:   datePath = '2017/03/30/12'
          datePathParts = re.split(r"/", datePath)
          #['2017', '03', '30', '12']
          #print datePathParts[3]
          RequestedYear = datePathParts[0]
          RequestedMonth = datePathParts[1]
          RequestedDay = datePathParts[2]
          RequestedHour = datePathParts[3]
          #
          datePathDate = datetime.datetime.strptime(RequestedYear+RequestedMonth+RequestedDay, "%Y%m%d").date()
          #print datePathDate
          #
          tm = datetime.time(int(RequestedHour), 0, 0)
          #
          dtRequested = datetime.datetime.combine(datePathDate, tm)
          #print 'dt:', dt
          #dt: 2017-03-30 12:00:00




          def GetPathsNHoursPrior_MatchedWeekdayAndHour(dtRequested, nPastHours):
              # MSSH0: It is more relevant to the current hour data, to pick paths with same hour and same day
of week!
              datePaths = []
              #
              for x in range(0, nPastHours):
                  TmpDate = dtRequested - timedelta(weeks=x)
                  #2017-03-30 12:00:00
                  StartDateForStandard = re.sub(r" +", '-', str(TmpDate) )
                  StartDateForStandard = re.sub(r":+", '-', str(StartDateForStandard) )
                  StartDateForStandardParts = re.split(r'-', StartDateForStandard)
                  #['2017', '03', '30', '12', '00', '00']
                  TmpPth = StartDateForStandardParts[0:4]
                  TmpPthStr = '/'.join(TmpPth)
                  datePaths.append(TmpPthStr)
              #
```

```
    return datePaths
# ASSUMEs:






def GetPathsNHoursPrior(dtRequested, nPastHours):
    # create array of paths for constructing the standard
    datePaths = []
    #
    for x in range(0, nPastHours):
        TmpDate = dtRequested - timedelta(hours=x)
        #2017-03-30 12:00:00
        StartDateForStandard = re.sub(r" +", '-', str(TmpDate) )
        StartDateForStandard = re.sub(r":+", '-', str(StartDateForStandard) )
        StartDateForStandardParts = re.split(r'-', StartDateForStandard)
        #['2017', '03', '30', '12', '00', '00']
        TmpPth = StartDateForStandardParts[0:4]
        TmpPthStr = '/'.join(TmpPth)
        datePaths.append(TmpPthStr)
    #
    return datePaths
# ASSUMEs:
"""
drwxrwxrwx   - flume hdfs          0 2017-03-29 01:00 hdfs:///input/vpnlogs/2017/03/29/00
drwxrwxrwx   - flume hdfs          0 2017-03-29 02:01 hdfs:///input/vpnlogs/2017/03/29/01
drwxrwxrwx   - flume hdfs          0 2017-03-29 03:01 hdfs:///input/vpnlogs/2017/03/29/02
drwxrwxrwx   - flume hdfs          0 2017-03-29 04:01 hdfs:///input/vpnlogs/2017/03/29/03
drwxrwxrwx   - flume hdfs          0 2017-03-29 05:01 hdfs:///input/vpnlogs/2017/03/29/04
drwxrwxrwx   - flume hdfs          0 2017-03-29 06:00 hdfs:///input/vpnlogs/2017/03/29/05
drwxrwxrwx   - flume hdfs          0 2017-03-29 07:00 hdfs:///input/vpnlogs/2017/03/29/06
drwxrwxrwx   - flume hdfs          0 2017-03-29 08:00 hdfs:///input/vpnlogs/2017/03/29/07
drwxrwxrwx   - flume hdfs          0 2017-03-29 09:01 hdfs:///input/vpnlogs/2017/03/29/08
drwxrwxrwx   - flume hdfs          0 2017-03-29 10:01 hdfs:///input/vpnlogs/2017/03/29/09
drwxrwxrwx   - flume hdfs          0 2017-03-29 11:01 hdfs:///input/vpnlogs/2017/03/29/10
drwxrwxrwx   - flume hdfs          0 2017-03-29 12:01 hdfs:///input/vpnlogs/2017/03/29/11
drwxrwxrwx   - flume hdfs          0 2017-03-29 13:01 hdfs:///input/vpnlogs/2017/03/29/12
drwxrwxrwx   - flume hdfs          0 2017-03-29 14:01 hdfs:///input/vpnlogs/2017/03/29/13
drwxrwxrwx   - flume hdfs          0 2017-03-29 15:01 hdfs:///input/vpnlogs/2017/03/29/14
drwxrwxrwx   - flume hdfs          0 2017-03-29 16:01 hdfs:///input/vpnlogs/2017/03/29/15
drwxrwxrwx   - flume hdfs          0 2017-03-29 17:01 hdfs:///input/vpnlogs/2017/03/29/16
drwxrwxrwx   - flume hdfs          0 2017-03-29 18:00 hdfs:///input/vpnlogs/2017/03/29/17
drwxrwxrwx   - flume hdfs          0 2017-03-29 19:01 hdfs:///input/vpnlogs/2017/03/29/18
drwxrwxrwx   - flume hdfs          0 2017-03-29 20:00 hdfs:///input/vpnlogs/2017/03/29/19
drwxrwxrwx   - flume hdfs          0 2017-03-29 21:01 hdfs:///input/vpnlogs/2017/03/29/20
drwxrwxrwx   - flume hdfs          0 2017-03-29 22:01 hdfs:///input/vpnlogs/2017/03/29/21
drwxrwxrwx   - flume hdfs          0 2017-03-29 23:01 hdfs:///input/vpnlogs/2017/03/29/22
drwxrwxrwx   - flume hdfs          0 2017-03-30 00:01 hdfs:///input/vpnlogs/2017/03/29/23
"""
###






###
#datePaths = GetPathsNHoursPrior(dtRequested, nPastHours)
datePaths = GetPathsNHoursPrior_MatchedWeekdayAndHour(dtRequested, nPastHours)

#
print datePaths
#
CumMsg = CumMsg+"\n"+  str(datePaths)







###
countDatePaths = []
#
for dp in datePaths:
    #
    #tm_start = datetime.datetime.now() ; print datetime.datetime.now()
    #
    vpnLogs = sc.textFile('/input/vpnlogs/'+dp+'/*')
    #
    totCount = vpnLogs.count()
    #
    countDatePaths.append(totCount)
#
print countDatePaths
#[66111, 62079, 73933, 96993]
#
CumMsg = CumMsg+"\n"+  str(countDatePaths)
```

```
###
a = np.array([countDatePaths])
b = a.transpose()
#
mtx = sc.parallelize(
    b
)  # an RDD of Vectors
#
summary = Statistics.colStats(mtx)
#
vrnc = summary.variance()[0]
stdrrr = pow(vrnc, 0.5)
#
mn = summary.mean()[0]
# coefficient of variation
cov = stdrrr / mn
#
min = summary.min()[0]
#
print vrnc
print 'sd: ', stdrrr
print 'mean: ', mn
#print cov
print 'COV: ',  cov
#
CumMsg = CumMsg+"\n"+  str('COV: '+ str(cov))
```

```
### 2.outlier measure
# how many SDs the current hour is, compared to mean of the leave-current-out (LCO) past N hours
#
#countDatePathsLCO = countDatePaths[:-1]
CurentHourCount = countDatePaths[0]
countDatePathsLCO = countDatePaths
countDatePathsLCO.pop(0)
#countDatePaths = [CurentHourCount, countDatePaths]
print countDatePathsLCO
#
a = np.array([countDatePathsLCO])
b = a.transpose()
# an RDD of Vectors
mtx = sc.parallelize(b)
#
summaryLCO = Statistics.colStats(mtx)
#
#
vrncLCO = summaryLCO.variance()[0]
stdrrrLCO = pow(vrncLCO, 0.5)
#
mnLCO = summaryLCO.mean()[0]
##
om = (CurentHourCount - mnLCO) / stdrrrLCO
#
print 'current hour:', CurentHourCount
print 'mean LCO:', mnLCO
print 'sd LCO:', stdrrrLCO
print 'Outlier measure:', om
#
CumMsg = CumMsg+"\n"+  str('SDs away: '+ str(om))
```

```
          ### ruleset
          # ini
          IfDataOk=[0, 0, 0]
          # six sigma
          #if (abs(om)<6) :
          #if (abs(om)<3) :
          # asymmetric importance: if cur count is below standard, it is more important, trigger easier.
          if (om>-3 and om<6) :
              IfDataOk[0] =1
          # COV
          if (abs(cov)<.75) :
              IfDataOk[1] =1
          # min record count
          #if (abs(min)>0) :
          if (CurentHourCount >0) :
              IfDataOk[2] =1
          #
          ### decision
          # ini
          IfWarn=1
          #
          #if (sum(IfDataOk) >= (len(IfDataOk)-1) ) :
          #if (sum(IfDataOk) = len(IfDataOk)) :
          if (IfDataOk ==[1, 1, 1]) :
              IfWarn=0
          #
          ###
          CumMsg = CumMsg+"\n"+  str('Data IfWarn: '+ str(IfWarn))

          ### email
          if (IfWarn==1) :
                   WorkerSubEmailSubject0 = WorkerSubEmailSubject0 +' . Exception=||'+str(IfDataOk)+'||'
                   WorkerSubEmailBody0 = WorkerSubEmailBody0 +' . Exception=||'+str(IfDataOk)+'||' +"\n"+ CumMsg
                   #
                   bashCommand = 'echo "'+ WorkerSubEmailBody0 +'" | mail -s "'+ WorkerSubEmailSubject0 +'" '+
WorkerSubEmailTo0
                   #
                   print os.system(bashCommand)
          else :
                   WorkerSubEmailSubject1 = WorkerSubEmailSubject1 +' . State=||'+str(IfDataOk)+'||'
                   #WorkerSubEmailBody1 = WorkerSubEmailBody1 +' . State=||'+str(IfDataOk)+'||'
                   WorkerSubEmailBody1 = WorkerSubEmailBody1 +' . State=||'+str(IfDataOk)+'||' +"\n"+ CumMsg
                   #
                   bashCommand = 'echo "'+ WorkerSubEmailBody1 +'" | mail -s "'+ WorkerSubEmailSubject1 +'" '+
WorkerSubEmailTo1
                   #
                   print os.system(bashCommand)
          #




          ###
          return CumMsg
#def WorkerSubVpn:
```

```
### catches all exceptions, not just system, hence can handle string exceptions etc.
try:
    ## call worker sub
    #WorkerSub()
    CumMsg = WorkerSubVpn()
    EmailBody1= EmailBody1 + str(CumMsg )
    #
    bashCommand = 'echo "'+ EmailBody1 +'" | mail -s "'+ EmailSubject1 +'" '+ EmailTo1
    #
    print os.system(bashCommand)
except:
    e = sys.exc_info()[0]
    # catches all exceptions, not just system, hence can handle string exceptions etc.
    #print 'WARN: exception caught:', e
    ## ALERT SysAdmin!
    #
    EmailSubject0 = EmailSubject0 +' . Exception=||'+str(e)+'||'
    EmailBody0 = EmailBody0 +' . Exception=||'+str(e)+'||'
    #
    bashCommand = 'echo "'+ EmailBody0 +'" | mail -s "'+ EmailSubject0 +'" '+ EmailTo0
    #
    print os.system(bashCommand)
#
```