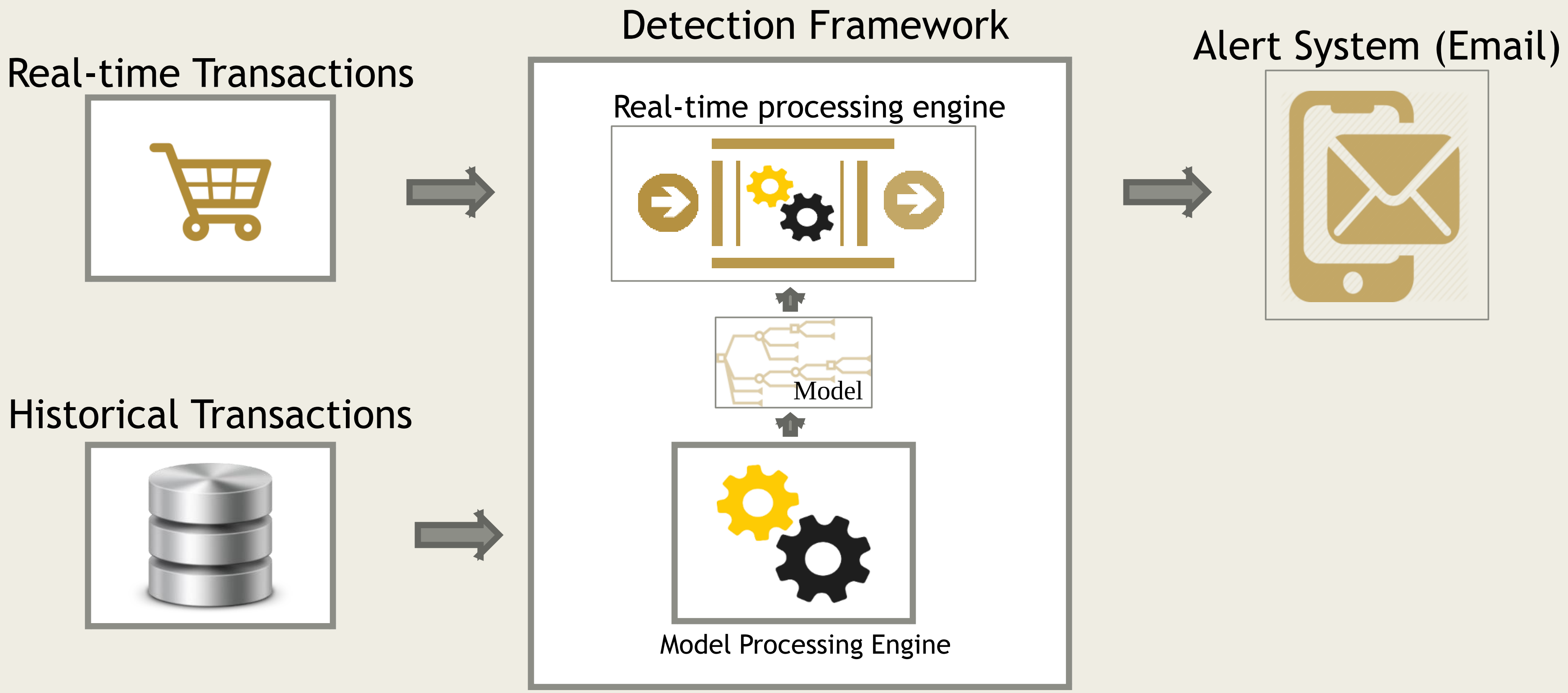
TERM PROJECT

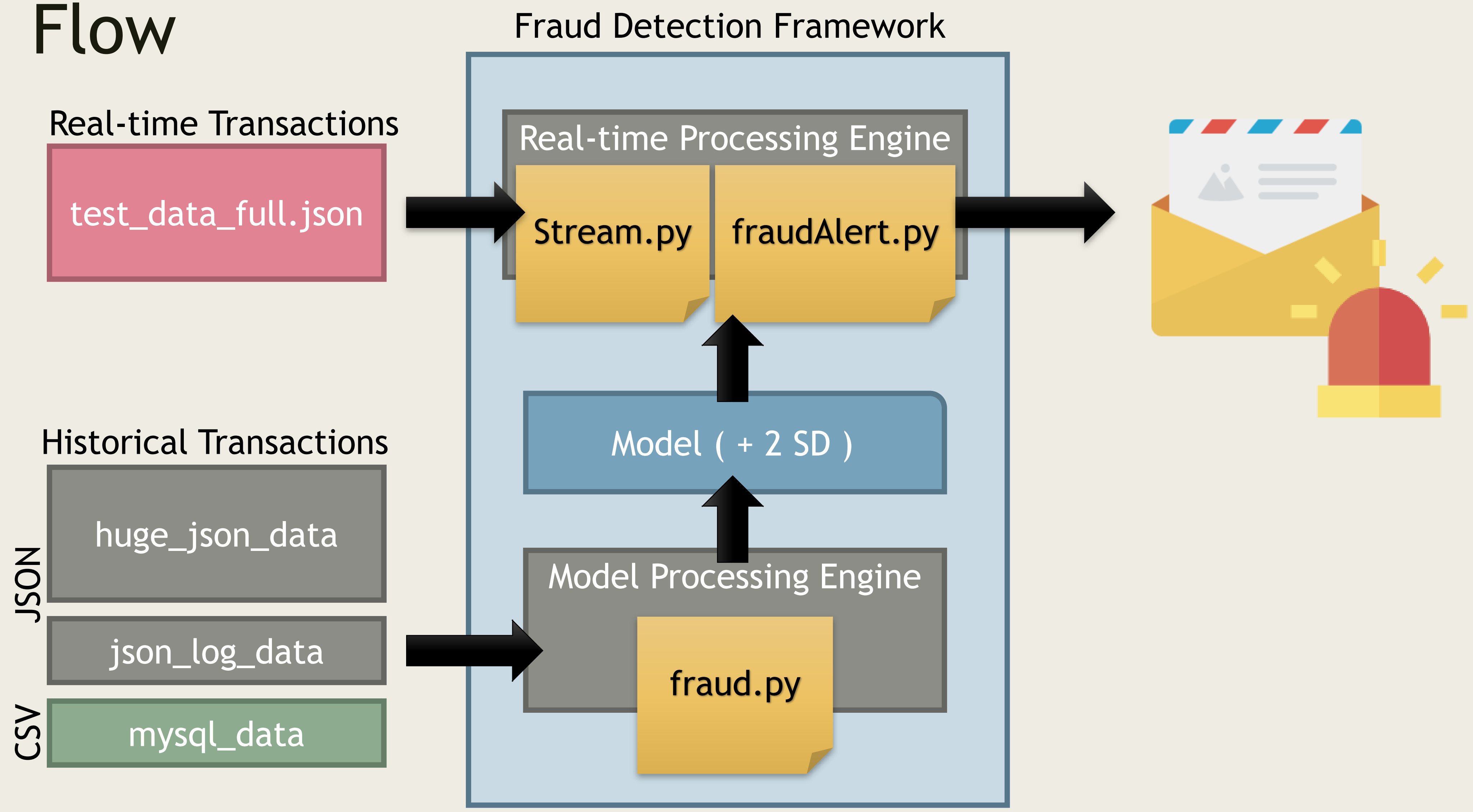
Near Real-time Transaction Fraud Detection

57070501009 – 57070501041 - 57070501049

Near Real-time Fraud Detection Framework

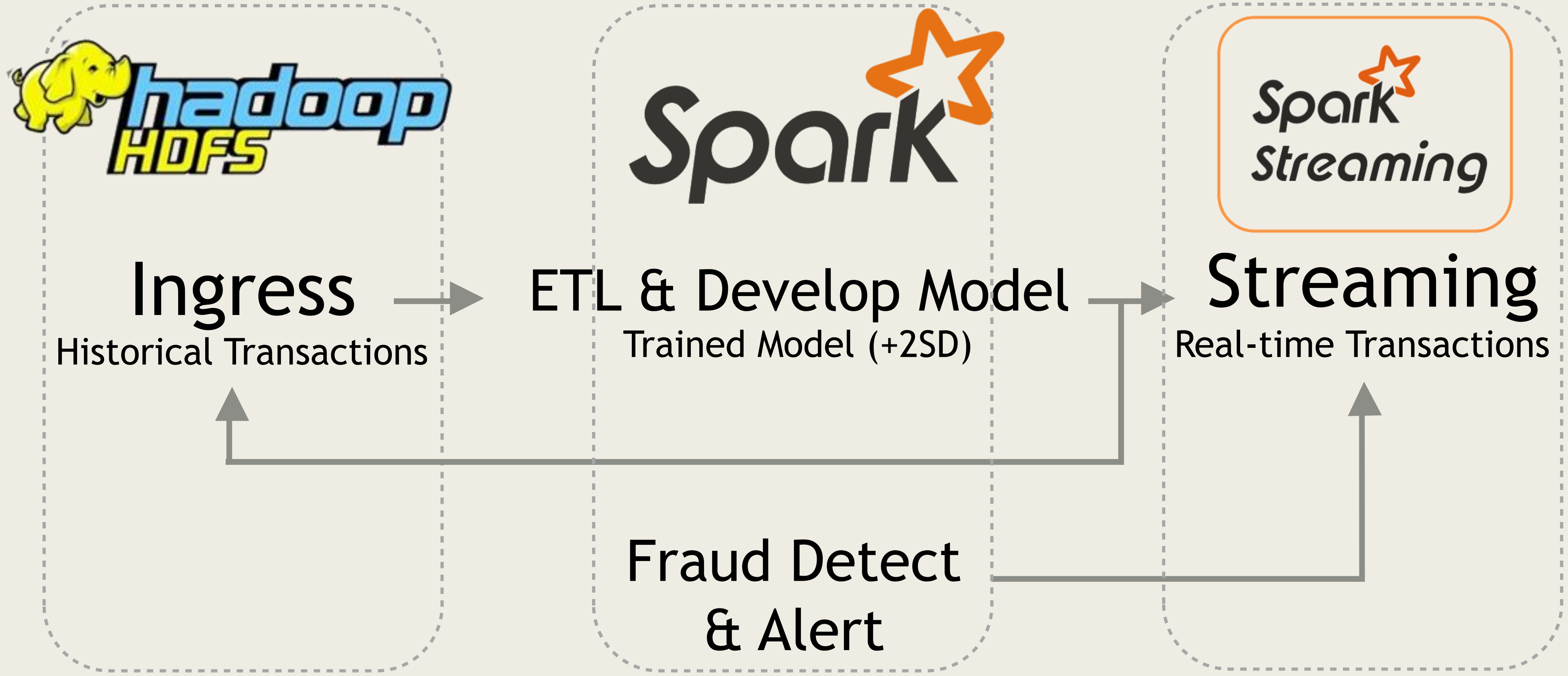


Flow

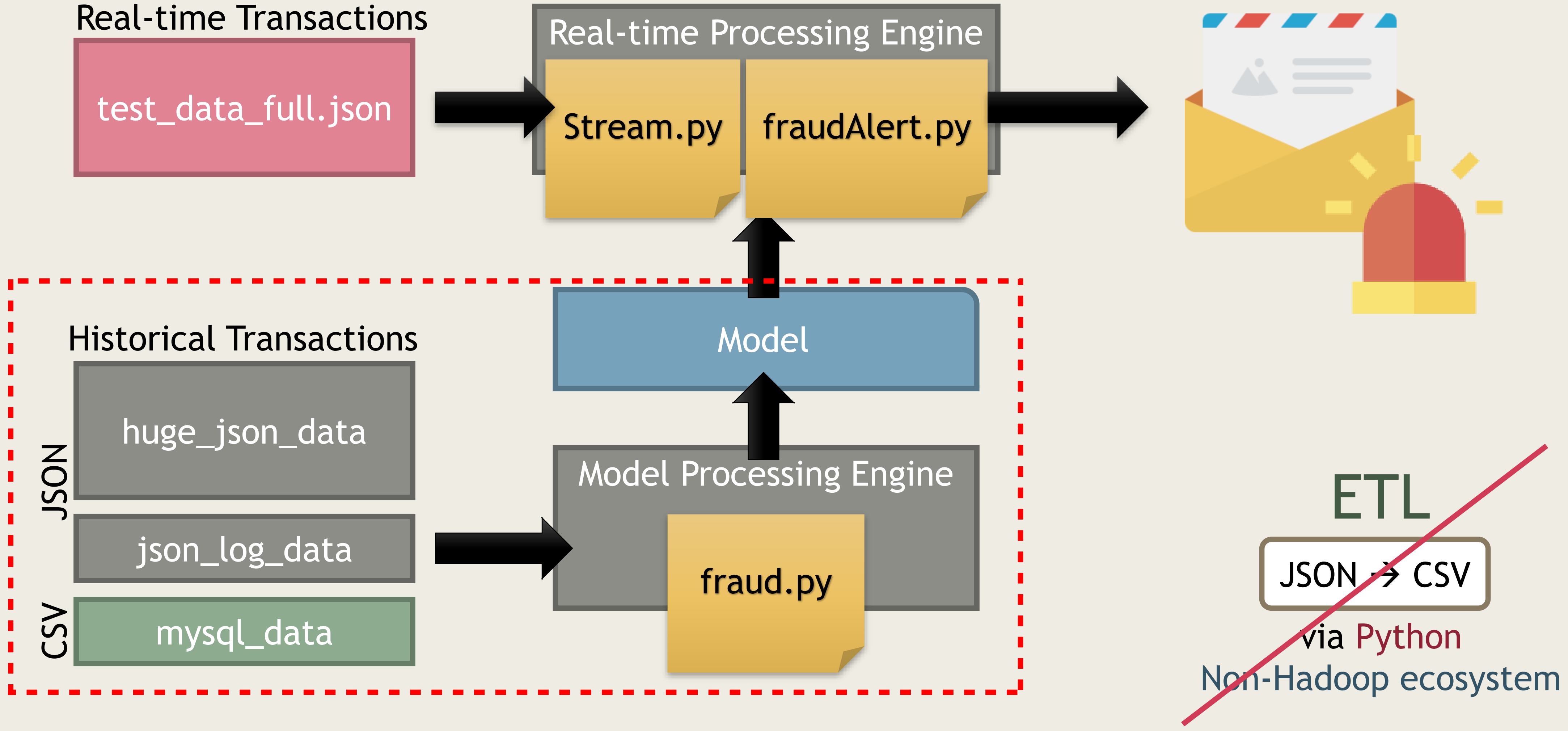


Fraud Detection Framework
Real-time Transactions
test_data_full.json
Real-time Processing Engine
Stream.py
fraudAlert.py
Model (+ 2 SD)
Historical Transactions
huge_json_data
json_log_data
mysql_data
JSON
CSV
Model Processing Engine
fraud.py

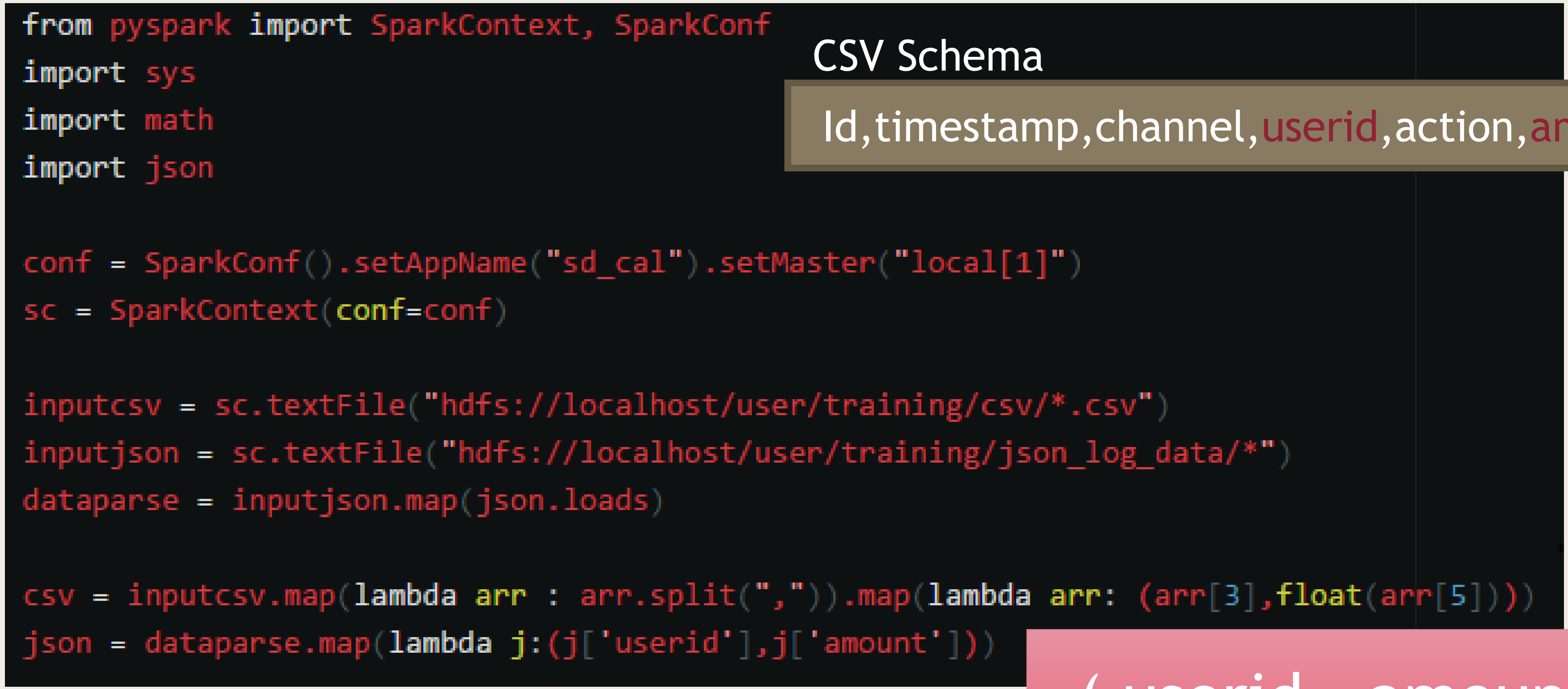
Tools


hadoop HDFS
Ingress
Historical Transactions
Spark
ETL & Develop Model
Trained Model (+2SD)
Spark Streaming
Streaming
Real-time Transactions
Fraud Detect & Alert

Flow – Model processing



Model processing : fraud.py



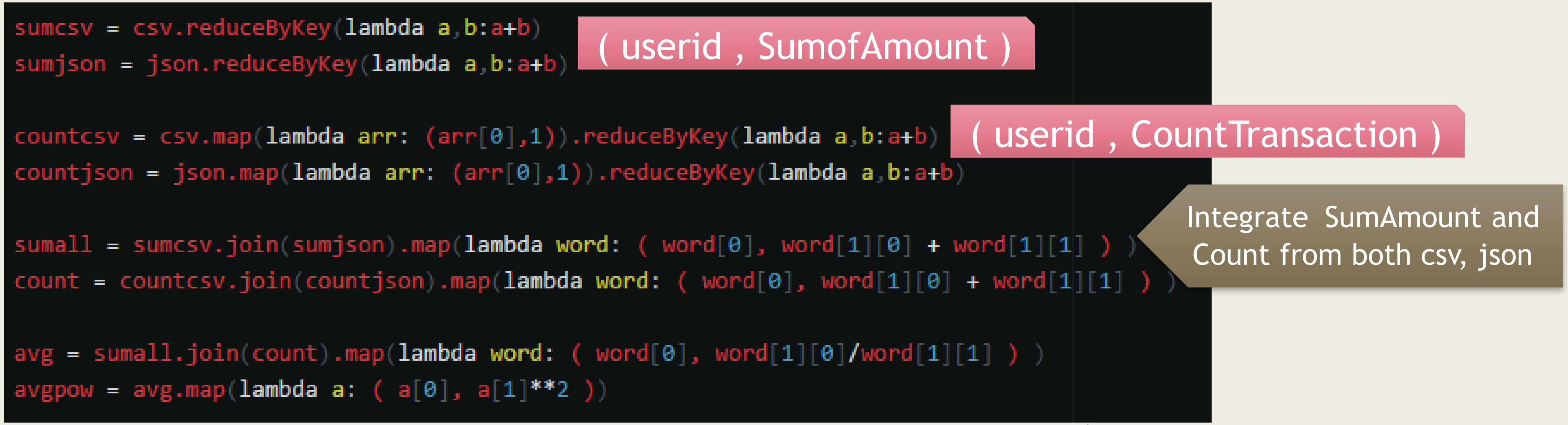
```python
from pyspark import SparkContext, SparkConf
import sys
import math
import json

conf = SparkConf().setAppName("sd_cal").setMaster("local[1]")
sc = SparkContext(conf=conf)

inputcsv = sc.textFile("hdfs://localhost/user/training/csv/*.csv")
inputjson = sc.textFile("hdfs://localhost/user/training/json_log_data/*")
dataparse = inputjson.map(json.loads)

csv = inputcsv.map(lambda arr : arr.split(",")).map(lambda arr: (arr[3],float(arr[5])))
json = dataparse.map(lambda j:(j['userid'],j['amount']))
```

CSV Schema

Id,timestamp,channel,userid,action,amount,location

(userid , amount)

Model processing : fraud.py

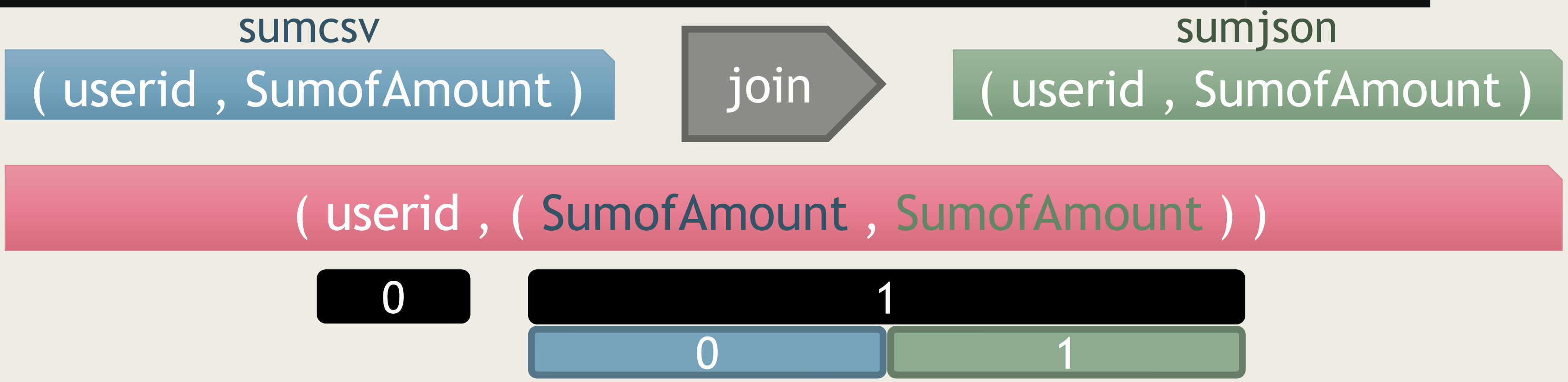


```
sumcsv = csv.reduceByKey(lambda a,b:a+b)
sumjson = json.reduceByKey(lambda a,b:a+b)

countcsv = csv.map(lambda arr: (arr[0],1)).reduceByKey(lambda a,b:a+b)
countjson = json.map(lambda arr: (arr[0],1)).reduceByKey(lambda a,b:a+b)

sumall = sumcsv.join(sumjson).map(lambda word: ( word[0], word[1][0] + word[1][1] ) )
count = countcsv.join(countjson).map(lambda word: ( word[0], word[1][0] + word[1][1] ) )

avg = sumall.join(count).map(lambda word: ( word[0], word[1][0]/word[1][1] ) )
avgpow = avg.map(lambda a: ( a[0], a[1]**2 ))
```

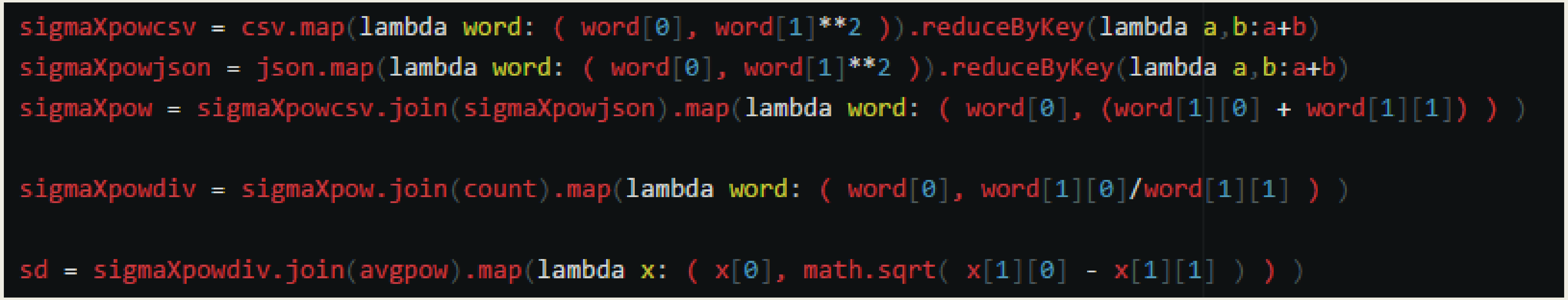
(userid , SumofAmount)

(userid , CountTransaction)

Integrate SumAmount and Count from both csv, json



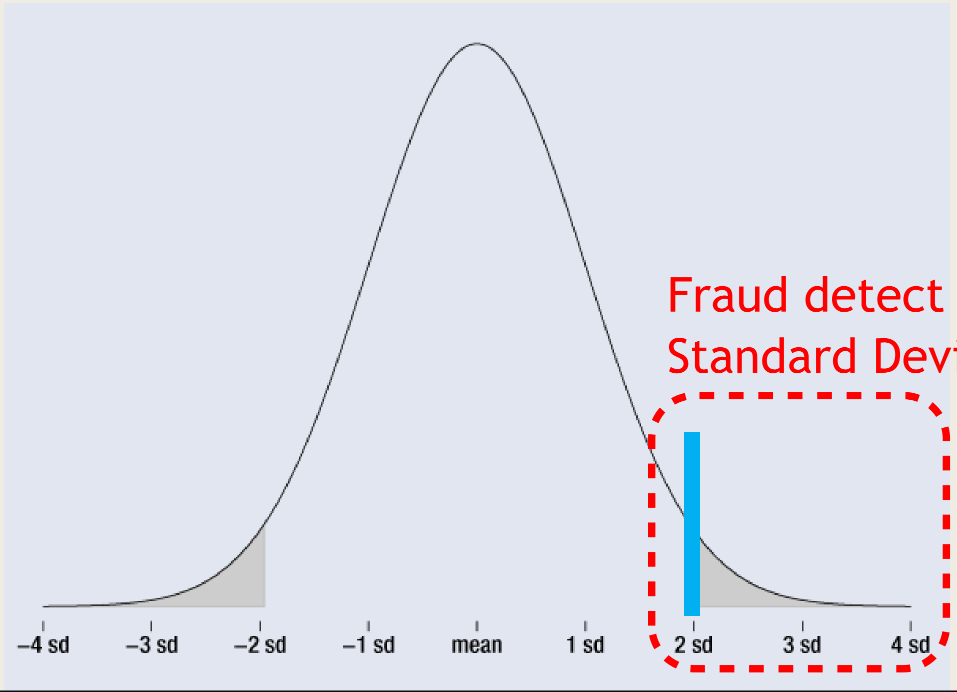
Model processing : fraud.py



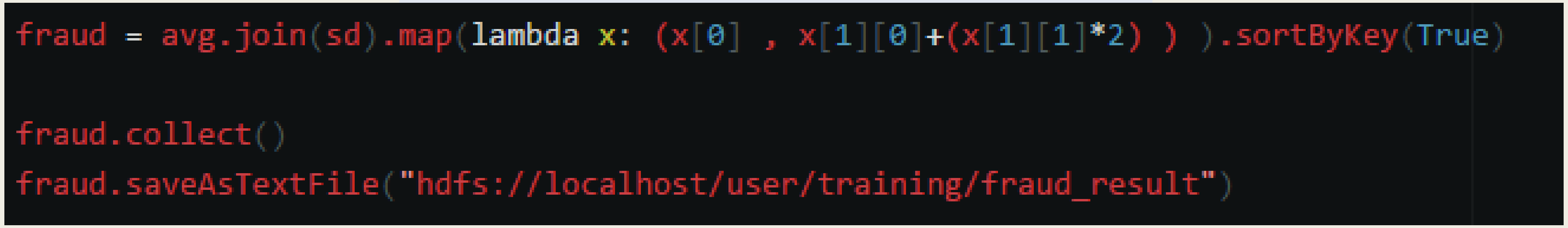
```
sigmaXpowcsv = csv.map(lambda word: ( word[0], word[1]**2 )).reduceByKey(lambda a,b:a+b)
sigmaXpowjson = json.map(lambda word: ( word[0], word[1]**2 )).reduceByKey(lambda a,b:a+b)
sigmaXpow = sigmaXpowcsv.join(sigmaXpowjson).map(lambda word: ( word[0], (word[1][0] + word[1][1]) ) )

sigmaXpowdiv = sigmaXpow.join(count).map(lambda word: ( word[0], word[1][0]/word[1][1] ) )

sd = sigmaXpowdiv.join(avgpow).map(lambda x: ( x[0], math.sqrt( x[1][0] - x[1][1] ) ) )
```

$$\sigma = \sqrt{\frac{\sum x^2}{N} - \bar{x}^2}$$

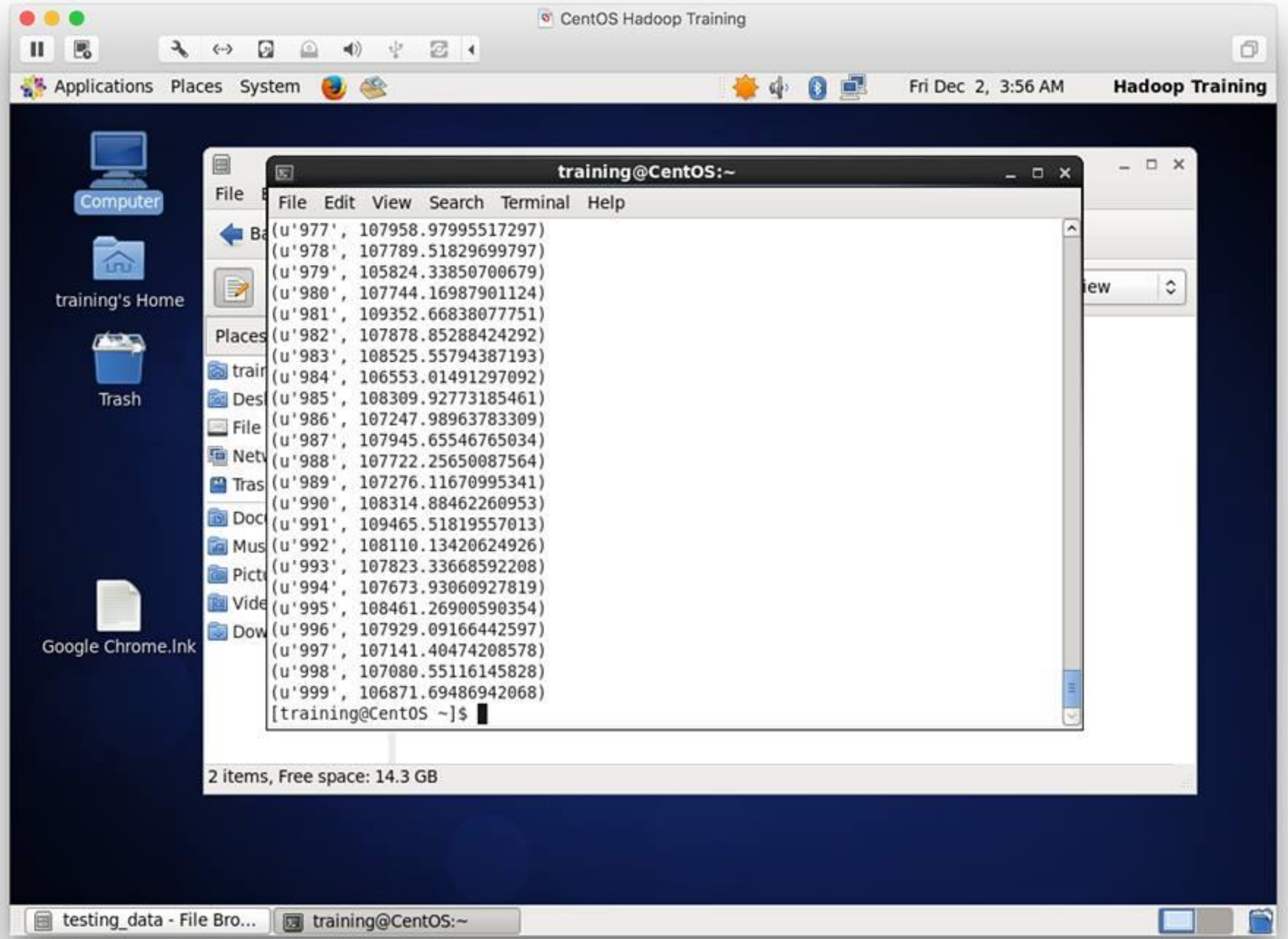
Model processing : fraud.py



Fraud detect
Standard Deviation +2 of each user

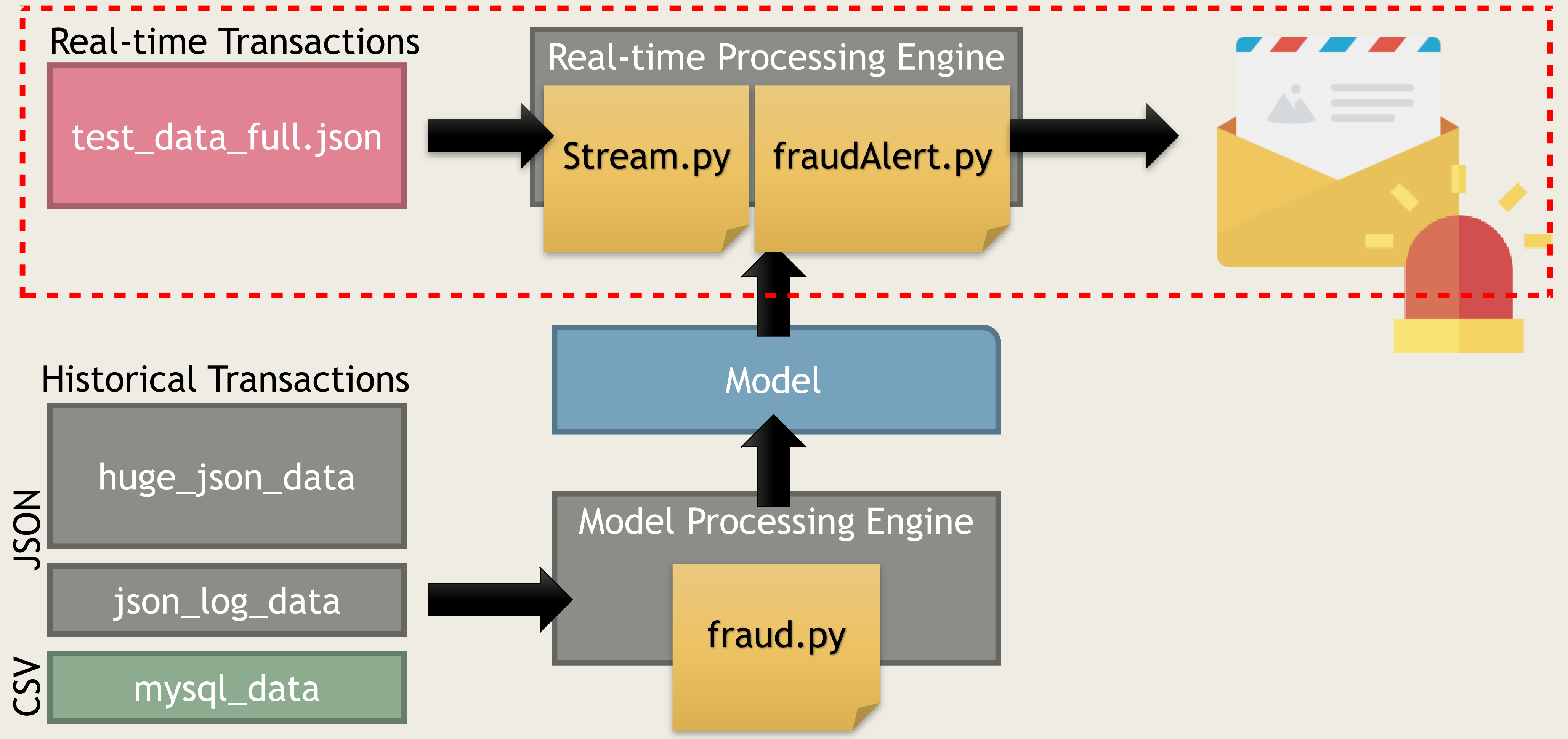


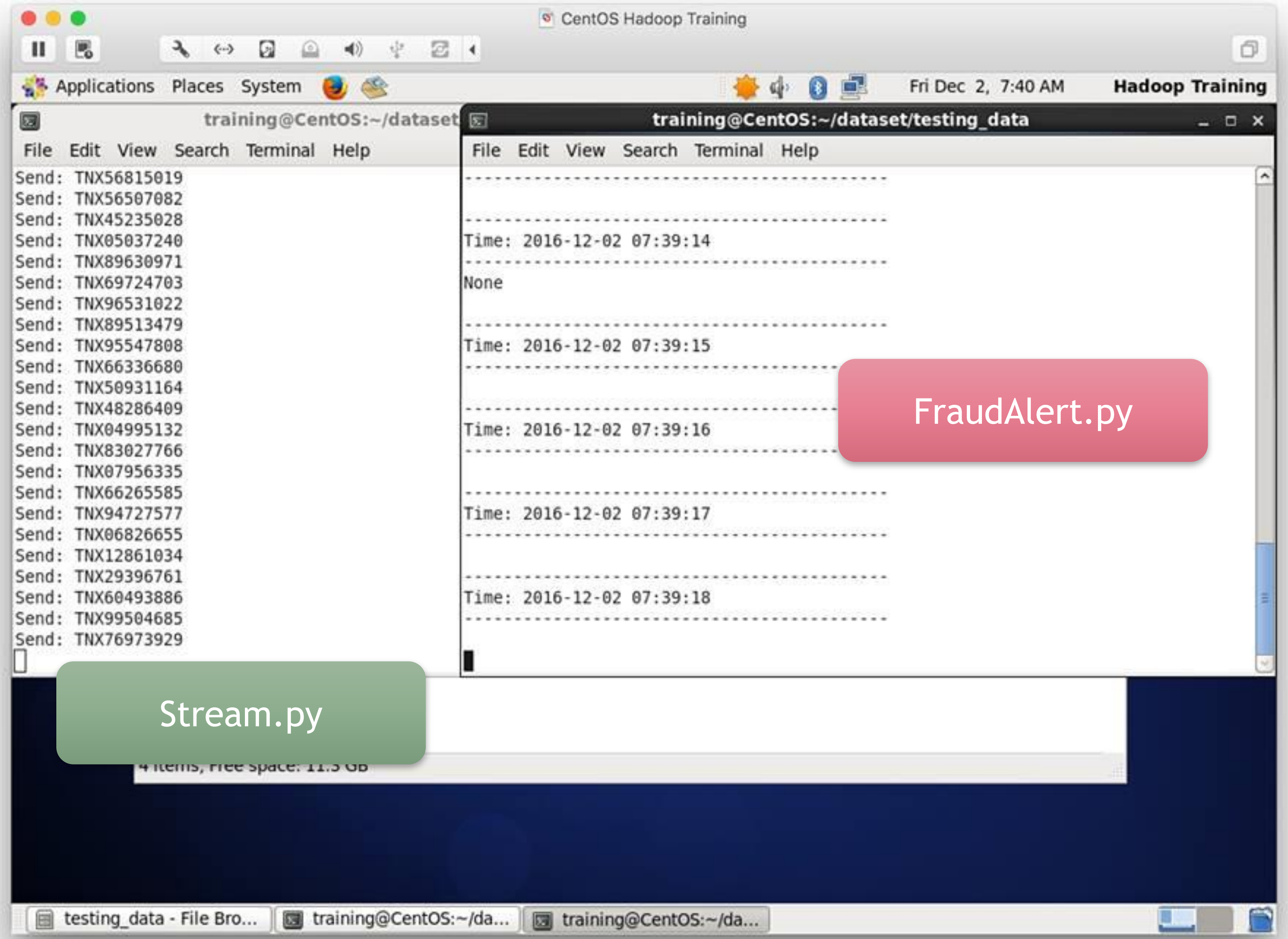
```
fraud = avg.join(sd).map(lambda x: (x[0] , x[1][0]+(x[1][1]*2) ) ).sortByKey(True)

fraud.collect()
fraud.saveAsTextFile("hdfs://localhost/user/training/fraud_result")
```



```
(u'977', 107958.97995517297)
(u'978', 107789.51829699797)
(u'979', 105824.33850700679)
(u'980', 107744.16987901124)
(u'981', 109352.66838077751)
(u'982', 107878.85288424292)
(u'983', 108525.55794387193)
(u'984', 106553.01491297092)
(u'985', 108309.92773185461)
(u'986', 107247.98963783309)
(u'987', 107945.65546765034)
(u'988', 107722.25650087564)
(u'989', 107276.11670995341)
(u'990', 108314.88462260953)
(u'991', 109465.51819557013)
(u'992', 108110.13420624926)
(u'993', 107823.33668592208)
(u'994', 107673.93060927819)
(u'995', 108461.26900590354)
(u'996', 107929.09166442597)
(u'997', 107141.40474208578)
(u'998', 107080.55116145828)
(u'999', 106871.69486942068)
[training@CentOS ~]$
```

Flow – Streaming & Detect fraud → Alert




CentOS Hadoop Training
Applications Places System
Fri Dec 2, 7:40 AM
Hadoop Training
training@CentOS:~/dataset
training@CentOS:~/dataset/testing_data
File Edit View Search Terminal Help
File Edit View Search Terminal Help
4 items, Free space: 11.3 GB
testing_data - File Bro...
training@CentOS:~/da...
training@CentOS:~/da...

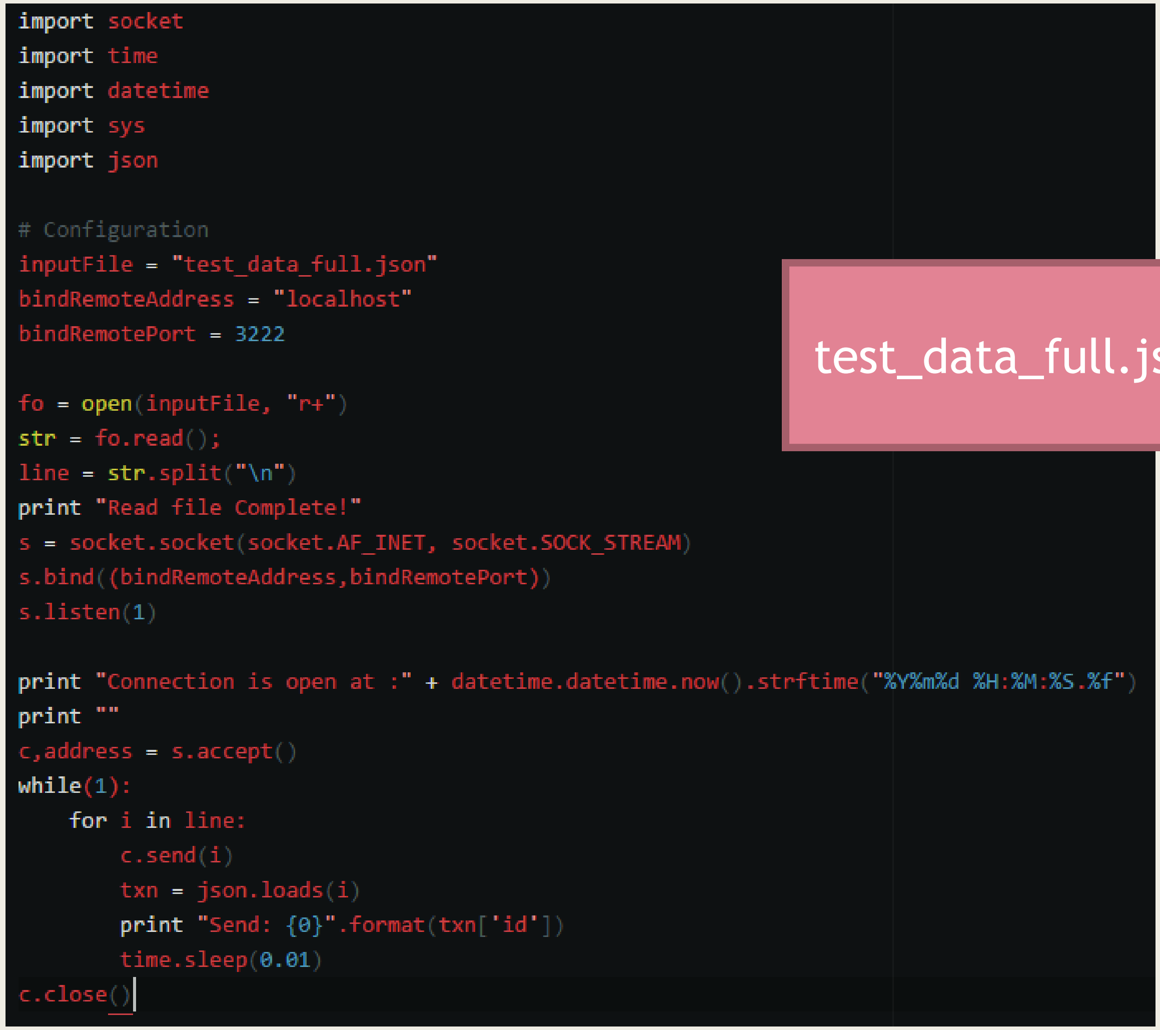
```
Send: TNX56815019
Send: TNX56507082
Send: TNX45235028
Send: TNX05037240
Send: TNX89630971
Send: TNX69724703
Send: TNX96531022
Send: TNX89513479
Send: TNX95547808
Send: TNX66336680
Send: TNX50931164
Send: TNX48286409
Send: TNX04995132
Send: TNX83027766
Send: TNX07956335
Send: TNX66265585
Send: TNX94727577
Send: TNX06826655
Send: TNX12861034
Send: TNX29396761
Send: TNX60493886
Send: TNX99504685
Send: TNX76973929
```

```
-------------------------------------------

-------------------------------------------
Time: 2016-12-02 07:39:14
-------------------------------------------
None

-------------------------------------------
Time: 2016-12-02 07:39:15
-------------------------------------------

-------------------------------------------
Time: 2016-12-02 07:39:16
-------------------------------------------

-------------------------------------------
Time: 2016-12-02 07:39:17
-------------------------------------------

-------------------------------------------
Time: 2016-12-02 07:39:18
-------------------------------------------
```

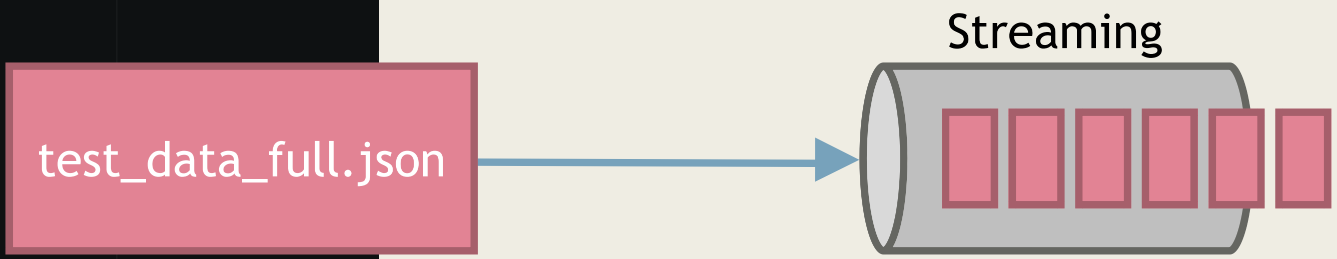
FraudAlert.py

Stream.py

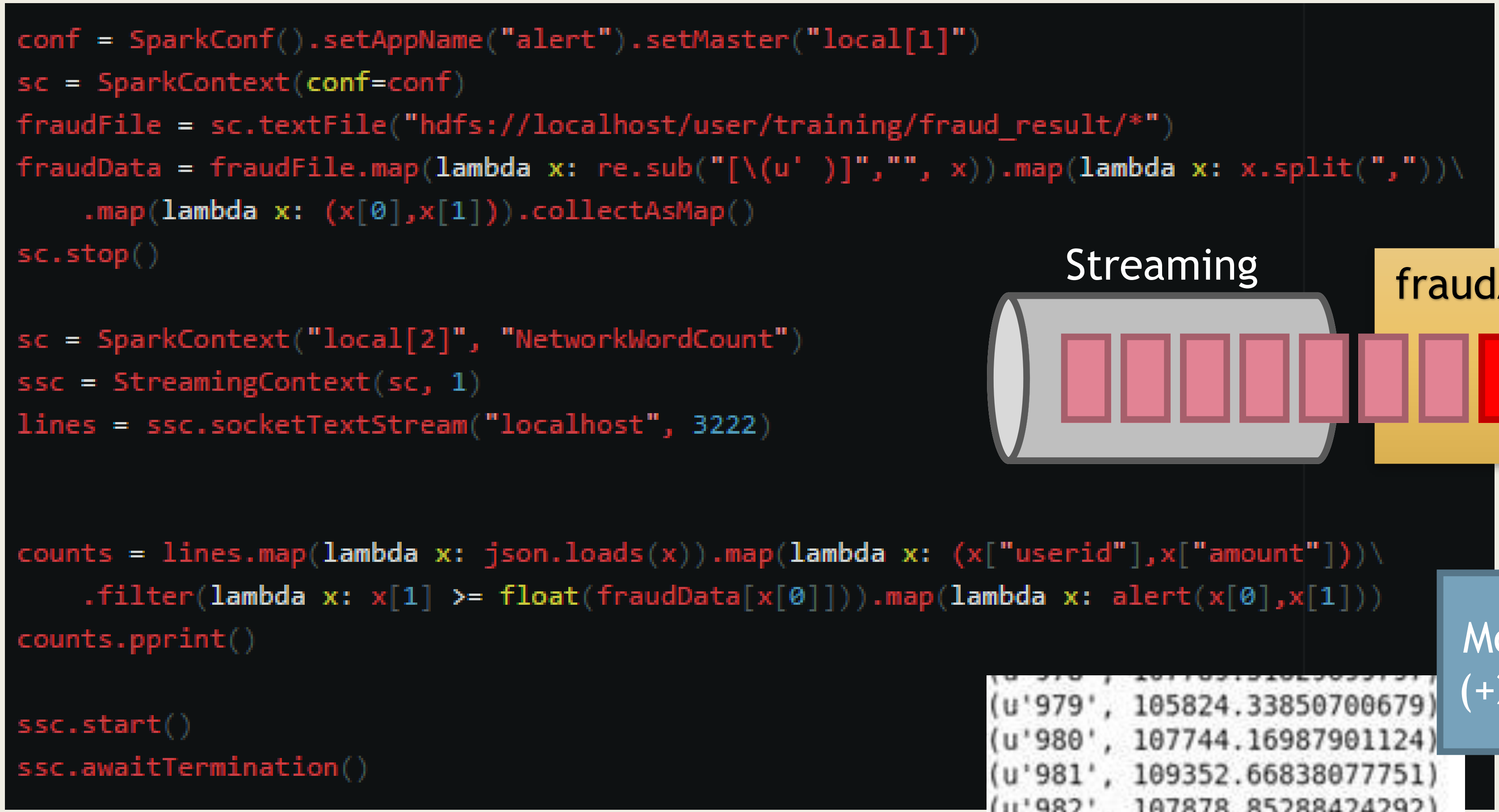
Streaming : Stream.py



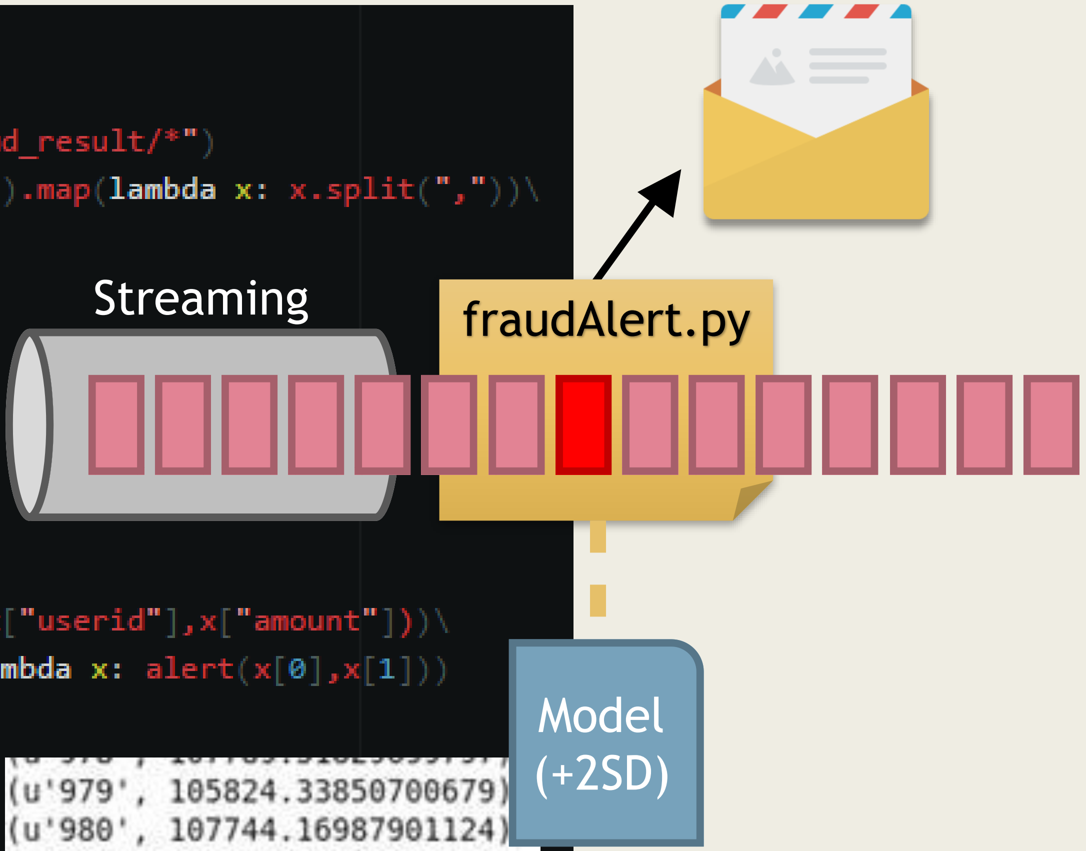
```python
import socket
import time
import datetime
import sys
import json

# Configuration
inputFile = "test_data_full.json"
bindRemoteAddress = "localhost"
bindRemotePort = 3222

fo = open(inputFile, "r+")
str = fo.read();
line = str.split("\n")
print "Read file Complete!"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((bindRemoteAddress,bindRemotePort))
s.listen(1)

print "Connection is open at :" + datetime.datetime.now().strftime("%Y%m%d %H:%M:%S.%f")
print ""
c,address = s.accept()
while(1):
    for i in line:
        c.send(i)
        txn = json.loads(i)
        print "Send: {0}".format(txn['id'])
        time.sleep(0.01)
c.close()
```

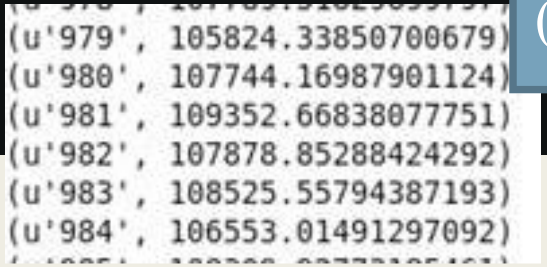


Detect fraud & Send E-mail : fraudAlert.py



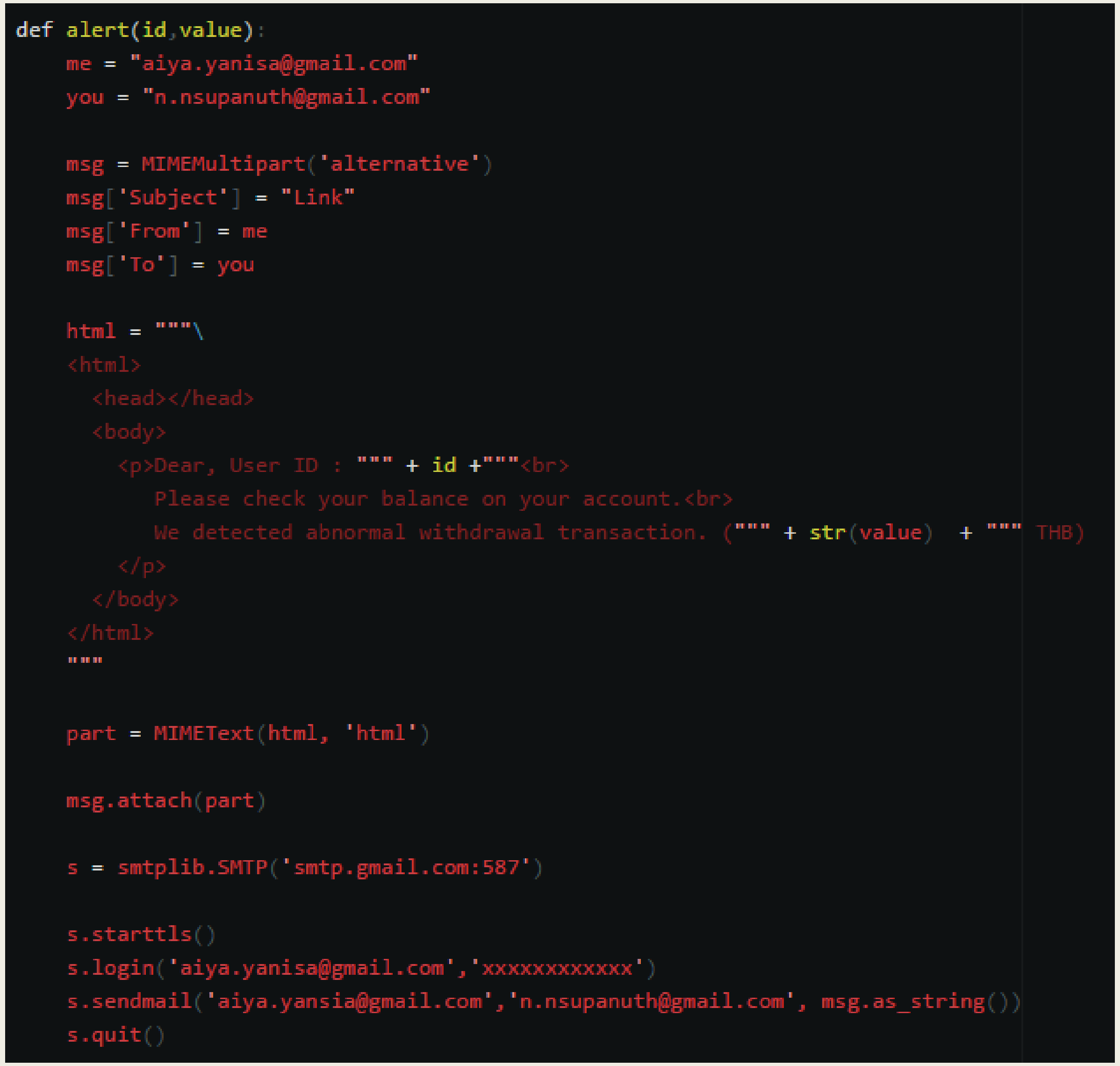
```
conf = SparkConf().setAppName("alert").setMaster("local[1]")
sc = SparkContext(conf=conf)
fraudFile = sc.textFile("hdfs://localhost/user/training/fraud_result/*")
fraudData = fraudFile.map(lambda x: re.sub("[\(u' )]","", x)).map(lambda x: x.split(","))\
    .map(lambda x: (x[0],x[1])).collectAsMap()
sc.stop()

sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
lines = ssc.socketTextStream("localhost", 3222)


counts = lines.map(lambda x: json.loads(x)).map(lambda x: (x["userid"],x["amount"]))\
    .filter(lambda x: x[1] >= float(fraudData[x[0]])).map(lambda x: alert(x[0],x[1]))
counts.pprint()

ssc.start()
ssc.awaitTermination()
```



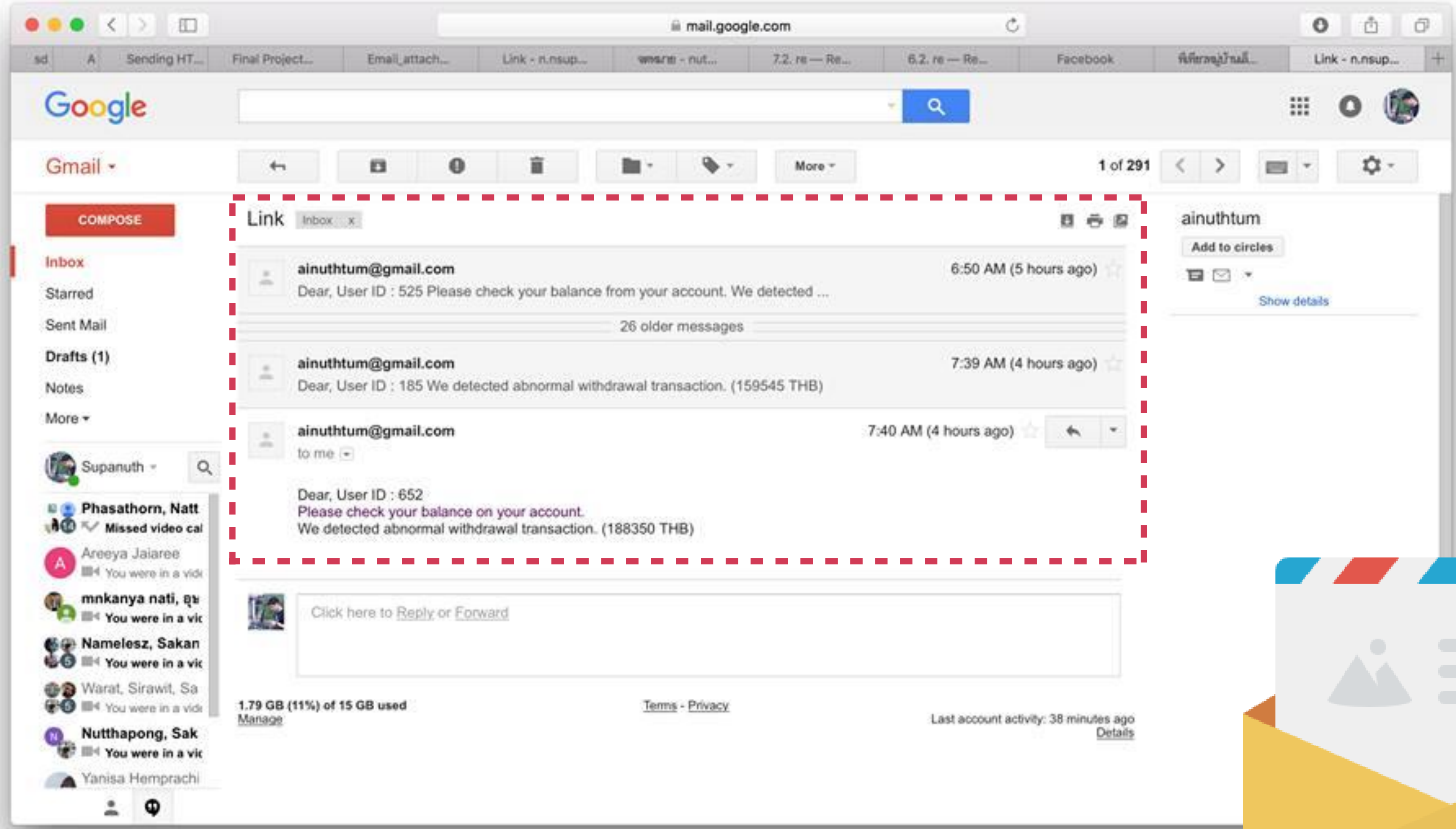


```
(u'979', 105824.33850700679)
(u'980', 107744.16987901124)
(u'981', 109352.66838077751)
(u'982', 107878.85288424292)
(u'983', 108525.55794387193)
(u'984', 106553.01491297092)
```

Detect fraud & Send E-mail : fraudAlert.py



```
def alert(id,value):
    me = "aiya.yanisa@gmail.com"
    you = "n.nsupanuth@gmail.com"

    msg = MIMEMultipart('alternative')
    msg['Subject'] = "Link"
    msg['From'] = me
    msg['To'] = you

    html = """\
    <html>
      <head></head>
      <body>
        <p>Dear, User ID : """ + id +"""<br>
           Please check your balance on your account.<br>
           We detected abnormal withdrawal transaction. (""" + str(value)  + """ THB)
        </p>
      </body>
    </html>
    """

    part = MIMEText(html, 'html')

    msg.attach(part)

    s = smtplib.SMTP('smtp.gmail.com:587')

    s.starttls()
    s.login('aiya.yanisa@gmail.com','xxxxxxxxxxxx')
    s.sendmail('aiya.yansia@gmail.com','n.nsupanuth@gmail.com', msg.as_string())
    s.quit()
```



Link Inbox

ainuthtum@gmail.com 6:50 AM (5 hours ago)
Dear, User ID : 525 Please check your balance from your account. We detected ...

26 older messages

ainuthtum@gmail.com 7:39 AM (4 hours ago)
Dear, User ID : 185 We detected abnormal withdrawal transaction. (159545 THB)

ainuthtum@gmail.com 7:40 AM (4 hours ago)
to me

Dear, User ID : 652
Please check your balance on your account.
We detected abnormal withdrawal transaction. (188350 THB)

Click here to Reply or Forward

1.79 GB (11%) of 15 GB used
Manage
Terms - Privacy
Last account activity: 38 minutes ago
Details

THANK YOU