1. Implement the following query using Hadoop streaming and python with the lineorder table.

<http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/SSBM_schema_hive.sql>

(**you still have to add the correct delimiter to table definitions**)

<http://rasinsrv07.cstcis.cti.depaul.edu/CSC555/SSBM1/lineorder.tbl>

SELECT lo\_quantity, AVG(lo\_revenue)

FROM (SELECT lo\_revenue, MAX(lo\_quantity) as lo\_quantity,

MAX(lo\_discount) as lo\_discount

FROM lineorder

WHERE lo\_orderpriority LIKE '%URGENT'

GROUP BY lo\_revenue)

WHERE lo\_discount BETWEEN 4 AND 9

GROUP BY lo\_quantity;

This requires running two different map reduce jobs. First, you would write a job that executes the subquery and produces an output. Then you would write a second job that uses output of the first job (in HDFS) as the input. Since we are only executing 2 MR jobs, we will do this transition manually.

**Single Node environment.**

**mapper1**

#!/usr/bin/python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

line = line.strip()

split = line.split('|')

lo\_revenue = (split[12])

lo\_discount = (split[11])

lo\_orderpriority = split[6]

lo\_quantity = (split[8]) # respective fields reading from the file.

if lo\_orderpriority is not None and lo\_orderpriority.endswith('URGENT'):

# checking for SQL like %URGENT

print (lo\_revenue + '\t' + lo\_quantity + '\t' + lo\_discount)

# key as lo\_revenue and other two values.

**reducer1**

#!/usr/bin/python

import sys

currentKey = None

qtyList = []

discountList = []

# input comes from STDIN

for line in sys.stdin:

split = line.strip().split('\t')

key = split[0] # ['Q11', 'Val1', 'Val2', 'Val3']

qtyvalue = int( split[1])

discountvalue = int(split[2])

if currentKey == key: # Same key

qtyList.append(qtyvalue)

discountList.append(discountvalue)

else:

if currentKey is not None:

# currentKey

print currentKey, '\t',max(qtyList),'\t',max(discountList)

qtyList = []

discountList = []

qtyList.append(qtyvalue)

discountList.append(discountvalue)

currentKey = key

if (len(qtyList) > 0 and len(discountList) > 0):

# Checking for the last key value pair which we need to print out

print currentKey, '\t',max(qtyList),'\t',max(discountList)

hadoop fs -rm -r /data/output

hadoop jar hadoop-streaming-2.6.4.jar -input /data -mapper mapper1.py -file mapper1.py -reducer reduce1.py -file reduce1.py -output /data/output

hadoop fs -cat /data/output/part-00000

Text

Description automatically generated with medium confidence

**mapper2**

#!/usr/bin/python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

line = line.strip()

split = line.split('\t')

lo\_revenue = split[0]

lo\_discount = int(split[2])

lo\_quantity = split[1] # default is string read so changing to int

if lo\_discount >= 4 and lo\_discount <= 9: #

print (lo\_quantity + '\t' + lo\_revenue)

**reducer2**

#!/usr/bin/python

import sys

currentKey = None

sumRevenue = 0

totalCount = 0

# input comes from STDIN

for line in sys.stdin:

split = line.strip().split('\t') # 'Q11 \t Val1 \t Val2 \t Val3'

key = split[0] # ['Q11', 'Val1', 'Val2', 'Val3']

value = int( split[1])

if currentKey == key: # Same key

sumRevenue = sumRevenue+value

totalCount = totalCount+1

else:

if currentKey:

# currentKey and value as AVG of revenue

print currentKey, '\t', sumRevenue / totalCount

sumRevenue = value

totalCount = 1

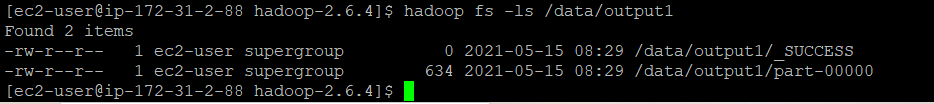
currentKey = key

if (sumRevenue > 0 and totalCount > 0):

# Checking for the last key value pair which we need to print out

print currentKey, '\t', sumRevenue / totalCount

$ hadoop jar hadoop-streaming-2.6.4.jar -input /data/output -mapper mapper2.py -file mapper2.py -reducer reduce2.py -file reduce2.py -output /data/output1



$ hadoop fs -cat /data/output1/part-00000

**Final Output.**

Graphical user interface, text

Description automatically generated