LAMPIRAN

Lampiran I. Kode Python untuk metode AHP biasa

Catatan: Data menggunakan data pada literatur untuk menguji kebenaran *output*.

```
# IMPORT LIBRARY
import numpy as np
import pandas as pd
import time
time_start = time.clock()

# ALTERNATIVES
alt = ["S1","S2","S3","S4","S5","S6","S7","S8","S9","S10","S11"]
data = [
    [1.86, 1024, 835, 4, 146.8, 36, 20352, 4],
    [3.66, 2000, 1300, 1, 440.4, 36, 48200, 3],
    [1.86, 1024, 835, 4, 146, 36, 25320, 6],
    [1.6, 1024, 835, 4, 146, 36, 23405, 6],
    [1.65, 2000, 1300, 1, 146, 24, 45250, 4],
    [1.75, 1024, 1300, 2, 73, 36, 32250, 6],
    [1.86, 2000, 835, 2, 73.4, 24, 24250, 4],
    [1.68, 1024, 835, 1, 146, 36, 26400, 6],
    [1.6, 1024, 1300, 2, 73.4, 36, 31100, 4],
    [1.57, 1024, 835, 2, 146, 24, 23304, 7],
    [1.58, 1024,1300, 4, 146, 24, 32450, 4]
    ]

# CRITERIAS
criteria = ["Processor", "Memory", "Power", "Cache", "Storage", "Warranty", "Price", "Delivery"]
flag = [True, True, False, True, True, True, False, False]
criteriaPC = [[1, 1, 7, 5, 2, 7, 1, 7],
              [1, 1, 7, 5, 2, 7, 1, 7],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1],
              [1/5, 1/5, 2, 1, 1/2, 2, 1/5, 2],
              [1/2, 1/2, 3, 2, 1, 3, 1/2, 3],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1],
              [1, 1, 7, 5, 2, 7, 1, 7],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1]]

# RANDOM INDEX
randomIdx = [0, 0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.51, 1.48, 1.56, 1.57, 1.59]


def cvtData2PriorityMtx(data, flag):
    result = []
    for i in range(len(data)):
        pcMtx = pairwiseComparison(data[i],flag[i])
        if checkConsistency(pcMtx):
            result.append(cvtPcMtx2PriorityVec(pcMtx))
        else:
            print(criteria[i], "matrix is not consistent")

        #PRINT KE CONSOLE
        #print("================================")
        #print(criteria[i])
        #print("================================")
        #print(pd.DataFrame(pcMtx, index = alt, columns = alt))
        #print("\n")

    return np.array(result).T
```

```
56.
57. def pairwiseComparison(temp, flag):
58.     result = []
59.
60.     for i in range(len(temp)):
61.         row = []
62.         for j in range(len(temp)):
63.             row.append(temp[i]/temp[j])
64.         result.append(row)
65.
66.     if flag == True:
67.         return np.array(result)
68.     elif flag == False:
69.         return 1/np.array(result)
70.
71. def cvtPcMtx2PriorityVec(matrix):
72.     sumVec = sum(matrix)                          #sum by col
73.     normMtx = matrix/sumVec                       #normalized
74.     priorityVec = np.mean(normMtx, axis = 1)    #average by row
75.     return priorityVec
76.
77. def checkConsistency(matrix):
78.     vec = cvtPcMtx2PriorityVec(matrix)
79.     res = np.dot(matrix, vec)
80.     lamda = np.mean(res/vec)
81.     n = len(matrix)
82.     consistencyIdx = (lamda - n)/(n - 1)
83.
84.     if consistencyIdx/randomIdx[n] < 0.1:
85.         return True
86.     else:
87.         return False
88.
89. # MAIN
90. dataMtx = np.array(data).T
91. altMtx = cvtData2PriorityMtx(dataMtx, flag)
92. criteriaMtx = np.array(criteriaPC)
93.
94. if checkConsistency(criteriaMtx):
95.     criteriaVec = cvtPcMtx2PriorityVec(criteriaMtx)
96.     finalResult = np.dot(altMtx, criteriaVec)
97. else:
98.     print("Criteria matrix is not consistent")
99.
100. #PRINT CONSOLE
101. #print("==================================")
102. #print("COMBINED PRIORITY VECTOR")
103. #print("==================================")
104. #print(pd.DataFrame(altMtx, index = alt, columns = criteria))
105. #print("\n")
106. #print("==================================")
107. #print("FINAL RESULT")
108. #print("==================================")
109. print(pd.Series(finalResult, index = alt).sort_values(ascending = False))
110.
111. time_elapsed = (time.clock() - time_start)
112. print(time_elapsed)
113.
114.
115. file = open("log_ahp_normal.txt", "a")
116. file.write(str(time_elapsed)+"\n")
117. file.close()
```

Lampiran II. Kode Python untuk metode AHP dengan Pembobotan Linear

Catatan: Data menggunakan data pada literatur untuk menguji kebenaran *output*.

```
# IMPORT LIBRARY
#import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import time
time_start = time.clock()

# ALTERNATIVES
alt = ["S1","S2","S3","S4","S5","S6","S7","S8","S9","S10","S11"]
data = [
    [1.86, 1024, 835, 4, 146.8, 36, 20352, 4],
    [3.66, 2000, 1300, 1, 440.4, 36, 48200, 3],
    [1.86, 1024, 835, 4, 146, 36, 25320, 6],
    [1.6, 1024, 835, 4, 146, 36, 23405, 6],
    [1.65, 2000, 1300, 1, 146, 24, 45250, 4],
    [1.75, 1024, 1300, 2, 73, 36, 32250, 6],
    [1.86, 2000, 835, 2, 73.4, 24, 24250, 4],
    [1.68, 1024, 835, 1, 146, 36, 26400, 6],
    [1.6, 1024, 1300, 2, 73.4, 36, 31100, 4],
    [1.57, 1024, 835, 2, 146, 24, 23304, 7],
    [1.58, 1024,1300, 4, 146, 24, 32450, 4]
    ]

# CRITERIAS
criteria = ["Processor", "Memory", "Power", "Cache", "Storage", "Warranty", "Price", "Delivery"]
flag = [True, True, False, True, True, True, False, False]
criteriaPC = [[1, 1, 7, 5, 2, 7, 1, 7],
              [1, 1, 7, 5, 2, 7, 1, 7],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1],
              [1/5, 1/5, 2, 1, 1/2, 2, 1/5, 2],
              [1/2, 1/2, 3, 2, 1, 3, 1/2, 3],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1],
              [1, 1, 7, 5, 2, 7, 1, 7],
              [1/7, 1/7, 1, 1/2, 1/3, 1, 1/7, 1]]

# RANDOM INDEX
randomIdx = [0, 0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49, 1.51, 1.48, 1.56, 1.57, 1.59]


def cvtPcMtx2PriorityVec(matrix):
    sumVec = sum(matrix)                        #sum by col
    normMtx = matrix/sumVec                     #normalized
    priorityVec = np.mean(normMtx, axis = 1)    #average by row
    return priorityVec

def checkConsistency(matrix):
    vec = cvtPcMtx2PriorityVec(matrix)
    res = np.dot(matrix, vec)
    lamda = np.mean(res/vec)
    n = len(matrix)
    consistencyIdx = (lamda - n)/(n - 1)

    if consistencyIdx/randomIdx[n] < 0.1:
        return True
    else:
        return False

```

```
58. def linearWeightage(allVec, allBoolean):
59.     result = []
60.     for i in range(len(allVec)):
61.         minVal = min(allVec[i])
62.         maxVal = max(allVec[i])
63.         rangeVal = maxVal - minVal
64.         if allBoolean[i]:
65.             #min threshold
66.             resultVec = (allVec[i] - minVal)/rangeVal
67.         else:
68.             #max threshold
69.             resultVec = (maxVal - allVec[i])/rangeVal
70.         result.append(resultVec)
71.     return np.array(result)
72. 
73. 
74. # MAIN
75. dataMtx = np.array(data).T
76. altMtx = linearWeightage(dataMtx, flag)
77. criteriaMtx = np.array(criteriaPC)
78. 
79. if checkConsistency(criteriaMtx):
80.     criteriaVec = cvtPcMtx2PriorityVec(criteriaMtx)
81.     finalResult = np.dot(criteriaVec, altMtx)*100
82. else:
83.     print("Criteria matrix is not consistent")
84. 
85. print(pd.Series(finalResult, index = alt).sort_values(ascending = False))
86. 
87. time_elapsed = (time.clock() - time_start)
88. print(time_elapsed)
89. 
90. #plotGraph(alt, finalResult)
91. file = open("log_ahp_hybrid.txt", "a")
92. file.write(str(time_elapsed)+"\n")
93. file.close()
```

Lampiran III. Kode Python untuk membandingkan kecepatan komputasi

```
import numpy as np

def cvtTxt2List(filename):
    log_file = open(filename, "r")
    list_time = [float(x) for x in log_file.read().split('\n')[:-1]]
    return np.array(list_time)

def statisticSummary(listData):
    mean = np.mean(listData)
    median = np.median(listData)
    variance = np.var(listData)

    print("Mean: ", mean)
    print("Median: ", median)
    print("Var: ", variance)

def compare(normal, hybrid):
    normal_mean = np.mean(normal)
    hybrid_mean = np.mean(hybrid)
    diff = normal_mean - hybrid_mean
    percentage = abs(round(diff/normal_mean*100, 5))
    if diff > 0:
        print("Hybrid is " + str(percentage) + "% faster than normal")
    else:
        print("Hybrid is " + str(percentage) + "% slower than normal")


hybrid = cvtTxt2List("log_ahp_hybrid.txt")
normal = cvtTxt2List("log_ahp_normal.txt")

print("Summary for Normal AHP")
statisticSummary(normal)
print("Summary for Hybrid AHP")
statisticSummary(hybrid)
compare(normal, hybrid)
```

Lampiran IV. *Output* kode Python

Output dari AHP biasa

```
S2     0.132129
S7     0.100474
S1     0.099017
S3     0.091780
S4     0.090736
S5     0.087457
S10    0.084707
S8     0.082239
S11    0.081782
S9     0.074921
S6     0.074759
dtype: float64
0.1271613
```

Output dari AHP dengan Pembobotan Linear

```
S2     66.538400
S7     56.423546
S1     45.027603
S3     38.970907
S4     37.620011
S5     32.553780
S8     30.158911
S10    29.255048
S11    24.399105
S9     23.149246
S6     22.176345
dtype: float64
0.0306611
```

Output perbandingan kecepatan komputasi

```
Summary for Normal AHP
Mean:  0.025830634782608695
Median:  0.017933400000000002
Var:  0.0005346476406883554
Summary for Hybrid AHP
Mean:  0.01994298
Median:  0.0172203
Var:  2.8973761614266678e-05
Hybrid is 22.7933% faster than normal
```

Lampiran V. Hasil *export* perhitungan ke file pdf

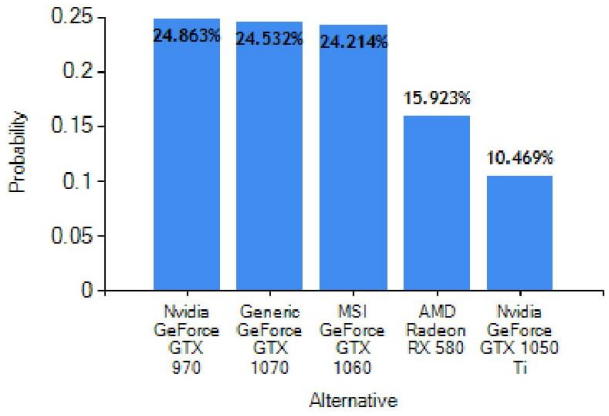
COMPARISON RESULT

Topic: GPU COMPARISON

DETAIL ALTERNATIVE

Alternative Name	Memory Size (MB)	Memory Bandwith (GB/s)	Clock Speed (MHz)	Pixel Rate (GPixel/s)	Floating-Point (GFLOPS)	Price (Rp)
AMD Radeon RX 580	8192	256	1257	42.88	6175	21730778
MSI GeForce GTX 1060	6144	192.2	1544	74.1	3953	4067770
Nvidia GeForce GTX 1050 Ti	4096	112.1	1752	41.3	1983	17381723
Nvidia GeForce GTX 970	4096	224.4	1753	58.8	3494	3551583
Generic GeForce GTX 1070	8192	256.3	1506	96.4	5783	5349338

COMPARISON



CONCLUSION

Best Alternative(s)

Nvidia GeForce GTX 970 (Best criteria: Price)

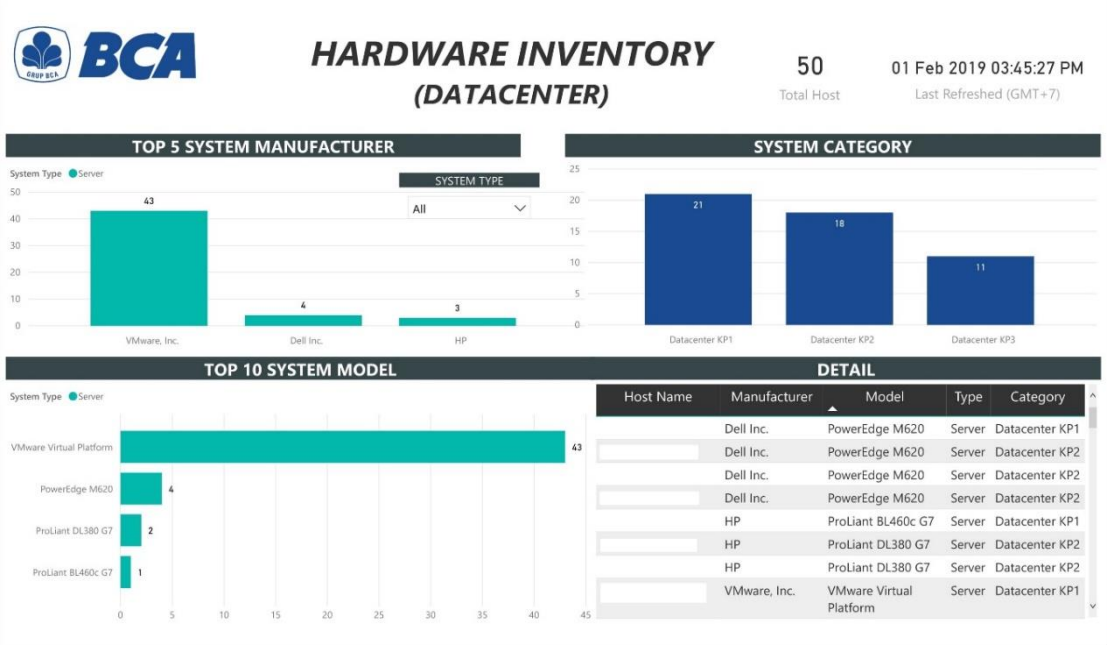
Worst Alternative(s)

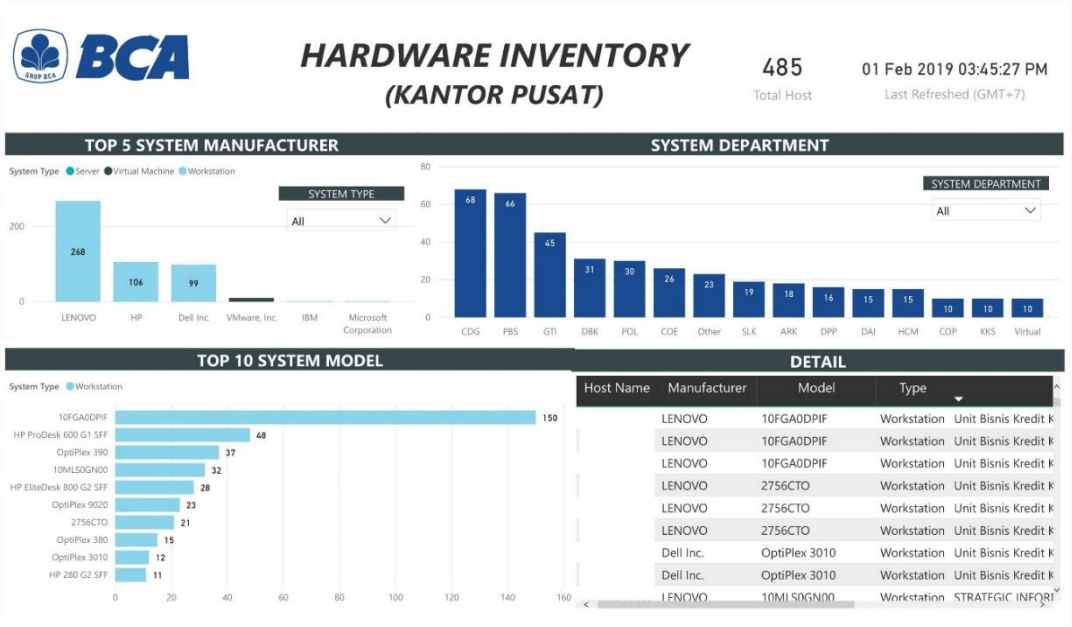
Nvidia GeForce GTX 1050 Ti (Worst criteria: Floating-Point)

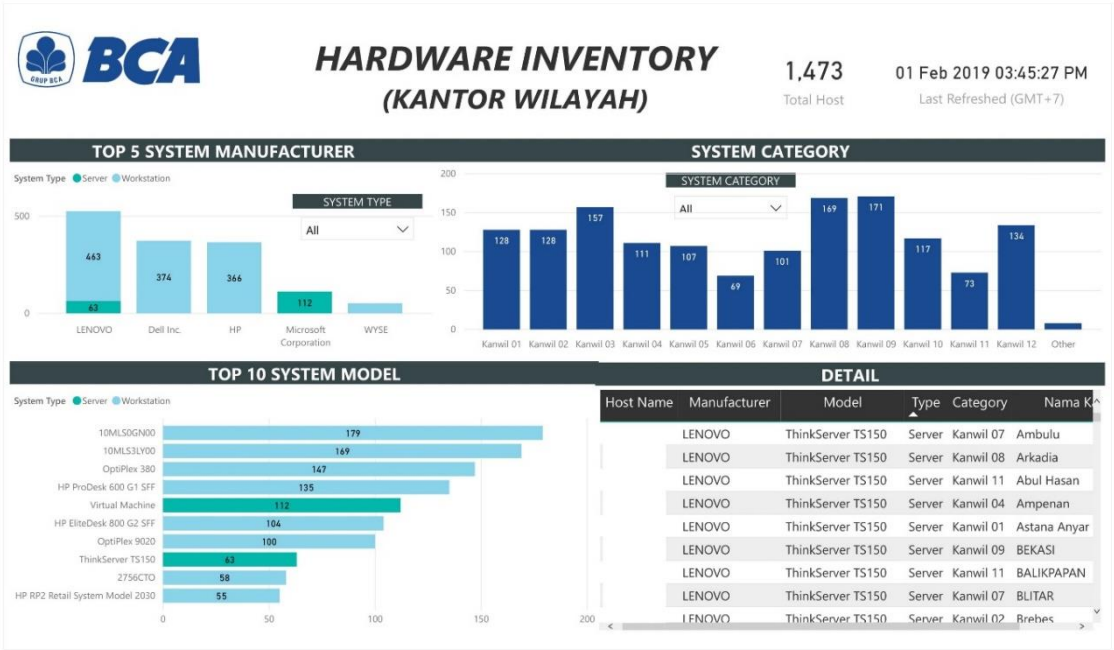
Generated automatically at 11/29/2018 11:06:36 AM by admin

Lampiran VI. *Dashboard* inventori *hardware* untuk *datacenter*, kantor pusat, dan kantor wilayah

Catatan: Data yang disajikan pada *dashboard* telah dimanipulasi dan disensor karena alasan privasi perusahaan.

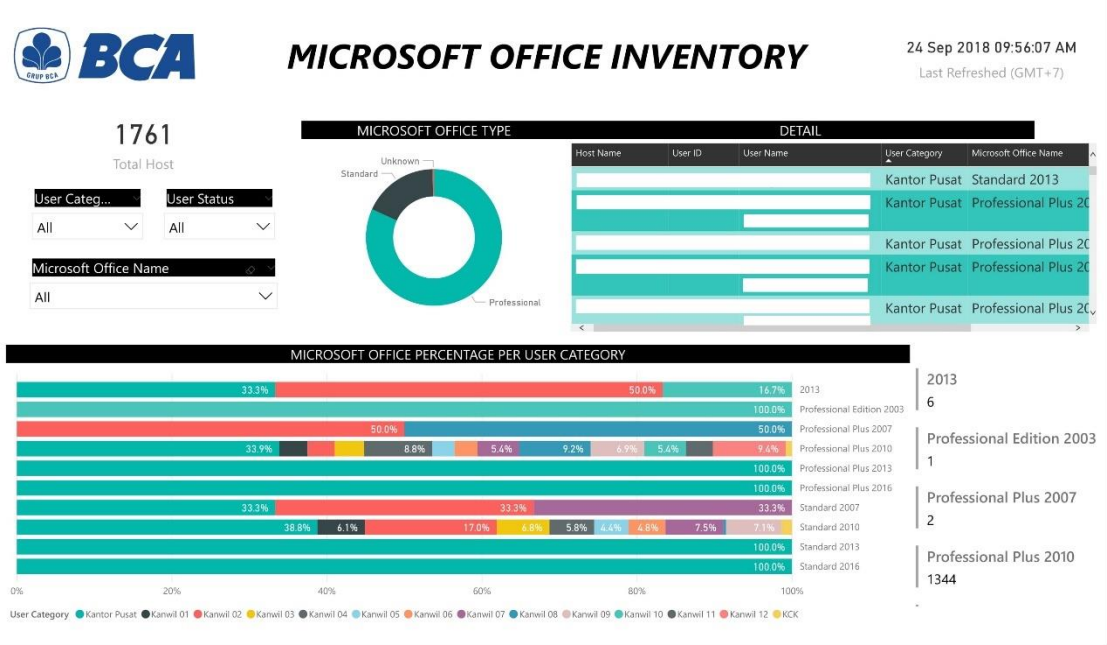


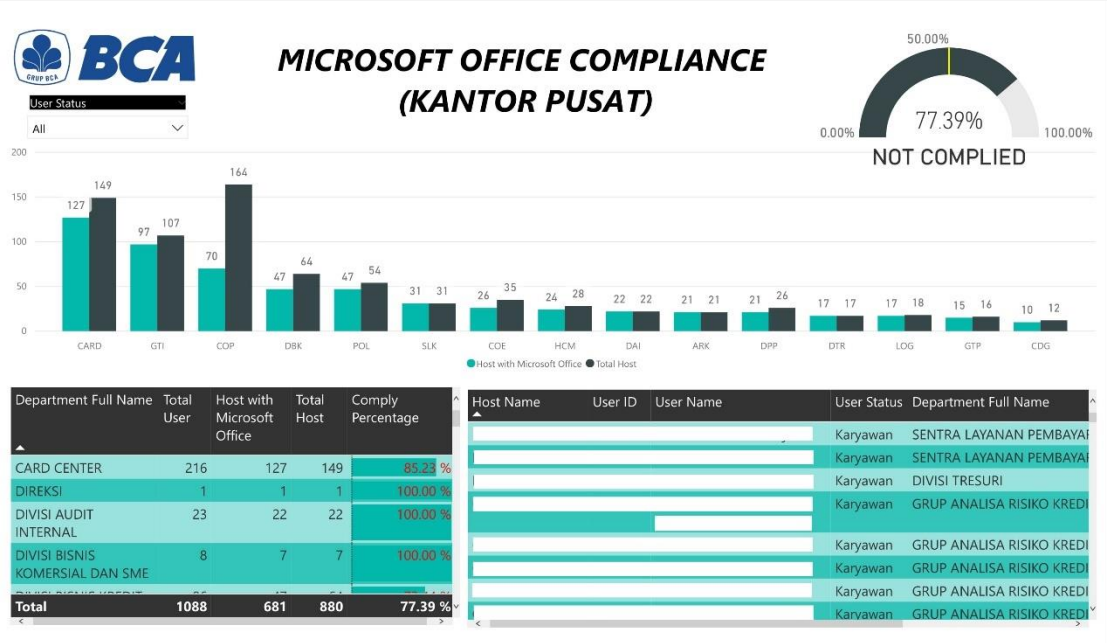


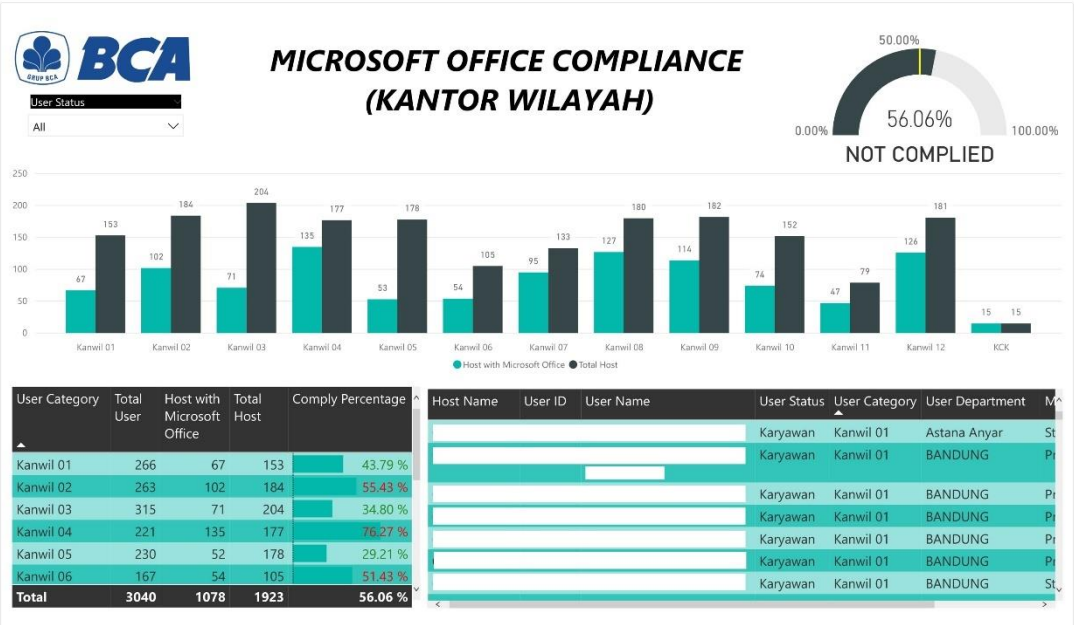

HARDWARE INVENTORY
(KANTOR WILAYAH)
1,473
Total Host
01 Feb 2019 03:45:27 PM
Last Refreshed (GMT+7)
TOP 5 SYSTEM MANUFACTURER
System Type Server Workstation
SYSTEM TYPE
All
500
0
463
63
374
366
112
LENOVO
Dell Inc.
HP
Microsoft Corporation
WYSE
SYSTEM CATEGORY
SYSTEM CATEGORY
All
200
150
100
50
0
128
128
157
111
107
69
101
169
171
117
73
134
Kanwil 01
Kanwil 02
Kanwil 03
Kanwil 04
Kanwil 05
Kanwil 06
Kanwil 07
Kanwil 08
Kanwil 09
Kanwil 10
Kanwil 11
Kanwil 12
Other
TOP 10 SYSTEM MODEL
System Type Server Workstation
10MLS0GN00 179
10MLS3LY00 169
OptiPlex 380 147
HP ProDesk 600 G1 SFF 135
Virtual Machine 112
HP EliteDesk 800 G2 SFF 104
OptiPlex 9020 100
ThinkServer TS150 63
2756CTO 58
HP RP2 Retail System Model 2030 55
0
50
100
150
200
DETAIL
Host Name Manufacturer Model Type Category Nama K
LENOVO ThinkServer TS150 Server Kanwil 07 Ambulu
LENOVO ThinkServer TS150 Server Kanwil 08 Arkadia
LENOVO ThinkServer TS150 Server Kanwil 11 Abul Hasan
LENOVO ThinkServer TS150 Server Kanwil 04 Ampenan
LENOVO ThinkServer TS150 Server Kanwil 01 Astana Anyar
LENOVO ThinkServer TS150 Server Kanwil 09 BEKASI
LENOVO ThinkServer TS150 Server Kanwil 11 BALIKPAPAN
LENOVO ThinkServer TS150 Server Kanwil 07 BLITAR
LENOVO ThinkServer TS150 Server Kanwil 02 Brebes

Lampiran VII. *Dashboard* Microsoft Office *Inventory and Compliance* untuk kantor pusat dan kantor wilayah

Catatan: Data yang disajikan pada *dashboard* telah dimanipulasi dan disensor karena alasan privasi perusahaan







User Category	Total User	Host with Microsoft Office	Total Host	Comply Percentage
Kanwil 01	266	67	153	43.79 %
Kanwil 02	263	102	184	55.43 %
Kanwil 03	315	71	204	34.80 %
Kanwil 04	221	135	177	76.27 %
Kanwil 05	230	52	178	29.21 %
Kanwil 06	167	54	105	51.43 %
Total	**3040**	**1078**	**1923**	**56.06 %**

Host Name	User ID	User Name	User Status	User Category	User Department	M
			Karyawan	Kanwil 01	Astana Anyar	St
			Karyawan	Kanwil 01	BANDUNG	Pr
			Karyawan	Kanwil 01	BANDUNG	Pr
			Karyawan	Kanwil 01	BANDUNG	Pr
			Karyawan	Kanwil 01	BANDUNG	Pr
			Karyawan	Kanwil 01	BANDUNG	Pr
			Karyawan	Kanwil 01	BANDUNG	St

Lampiran VIII. Kode Python untuk *scraping* tabel *telephone directory* BCA

Catatan: Link pada variabel baseLink dimanipulasi karena alasan privasi perusahaan.

```
1.  import pandas as pd
2.  import time
3.  import sys
4.  import os
5.
6.  baseLink = "http://.../telpdir/kantorbca/kantorbca.asp?"
7.
8.  def getTotalPages():
9.      tables = pd.read_html(baseLink)
10.     pageOfStr = tables[1][0][0]
11.     pageOfIdx = pageOfStr.find('of')+3
12.     return int(pageOfStr[pageOfIdx:])
13.
14. def getTableHeader():
15.     tables = pd.read_html(baseLink)
16.     return tables[2].iloc[0]
17.
18. def getTableData(pageNo):
19.     link = baseLink+"PageNo="+str(pageNo)
20.     tables = pd.read_html(link)
21.     return tables[2][1:]
22.
23. def main():
24.     # DATA SCRAPING
25.     try:
26.         startTime = time.time()
27.         print("Scraping Data...")
28.         data = []
29.         for i in range(1, getTotalPages()+1):
30.             data.append(getTableData(i))
31.         result = pd.concat(data).reset_index(drop=True)
32.         result.columns = getTableHeader()
33.         endTime = time.time()
34.         processTime = endTime - startTime
35.         print("Scraping Done! Time: {:.2f} seconds".format(processTime))
36.     except:
37.         print("Scraping Error, Check your connection access to Telephone Direc
    tory")
38.         raw_input()
39.         sys.exit(0)
40.
41.     # INPUT FILENAME
42.     filename = ""
43.     while len(filename) == 0:
44.         filename = raw_input("Input Excel Filename (.xlsx): ")
45.     if filename[-5:] != ".xlsx":
46.         filename += ".xlsx"
47.
48.     # WRITE DATA TO EXCEL
49.     try:
50.         writer = pd.ExcelWriter(filename)
51.         result.to_excel(writer, "Sheet1")
52.         writer.save()
53.
54.         print("Write "+filename+" Success")
55.
56.         temp = ""
57.         while temp != 'y' and temp != 'n':
58.             temp = raw_input("Do you want to open it (y or n)? ")
```

```
59.          if temp == 'y':
60.              os.startfile(filename)
61.      except:
62.          print("Write "+filename+" Error, Check your data")
63.          raw_input()
64.          sys.exit(0)
65.
66. main()
```

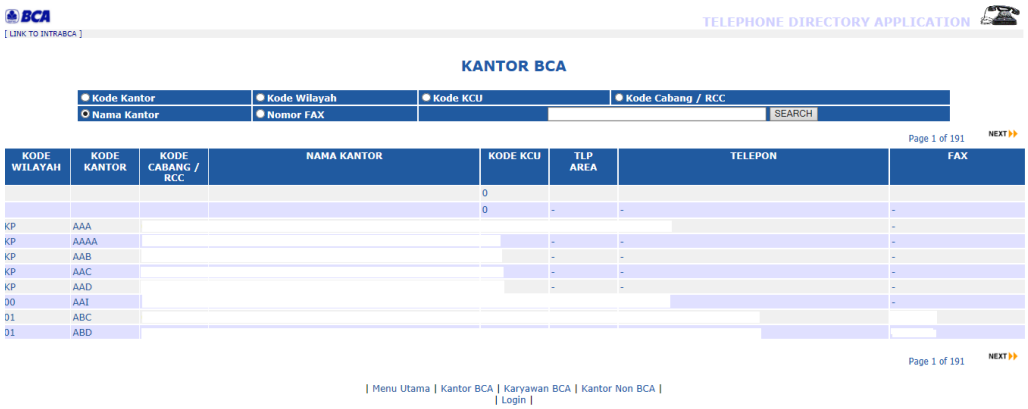
Lampiran IX. Hasil *scraping* tabel *telephone directory* BCA

Catatan: Data pada gambar disensor karena alasan privasi perusahaan

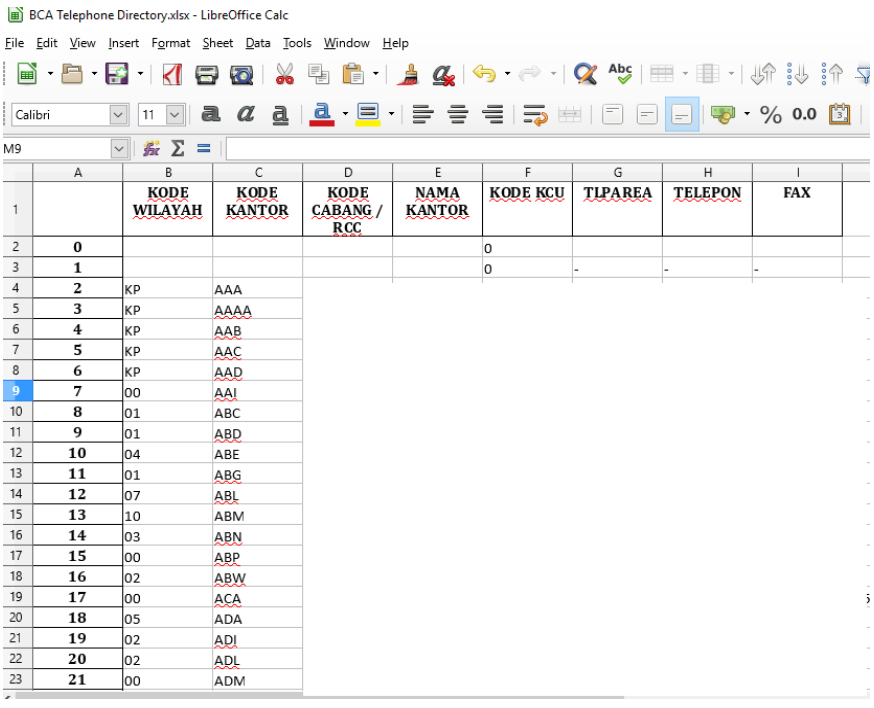
Hasil *Scraping* pada *Console* Python

```
Scraping Data...
Scraping Done! Time: 23.18 seconds
Input Excel Filename (.xlsx): BCA Telephone Directory
Write BCA Telephone Directory.xlsx Success
Do you want to open it (y or n)? y
```

Sumber Tabel HTML yang di*scraping*



Hasil *Scraping* Tabel HTML menjadi *File* Excel



	A	B	C	D	E	F	G	H	I
1		KODE WILAYAH	KODE KANTOR	KODE CABANG / RCC	NAMA KANTOR	KODE KCU	TLPAREA	TELEPON	FAX
2	0					0			
3	1					0	-	-	-
4	2	KP	AAA						
5	3	KP	AAAA						
6	4	KP	AAB						
7	5	KP	AAC						
8	6	KP	AAD						
9	7	00	AAI						
10	8	01	ABC						
11	9	01	ABD						
12	10	04	ABE						
13	11	01	ABG						
14	12	07	ABL						
15	13	10	ABM						
16	14	03	ABN						
17	15	00	ABP						
18	16	02	ABW						
19	17	00	ACA						
20	18	05	ADA						
21	19	02	ADI						
22	20	02	ADL						
23	21	00	ADM						