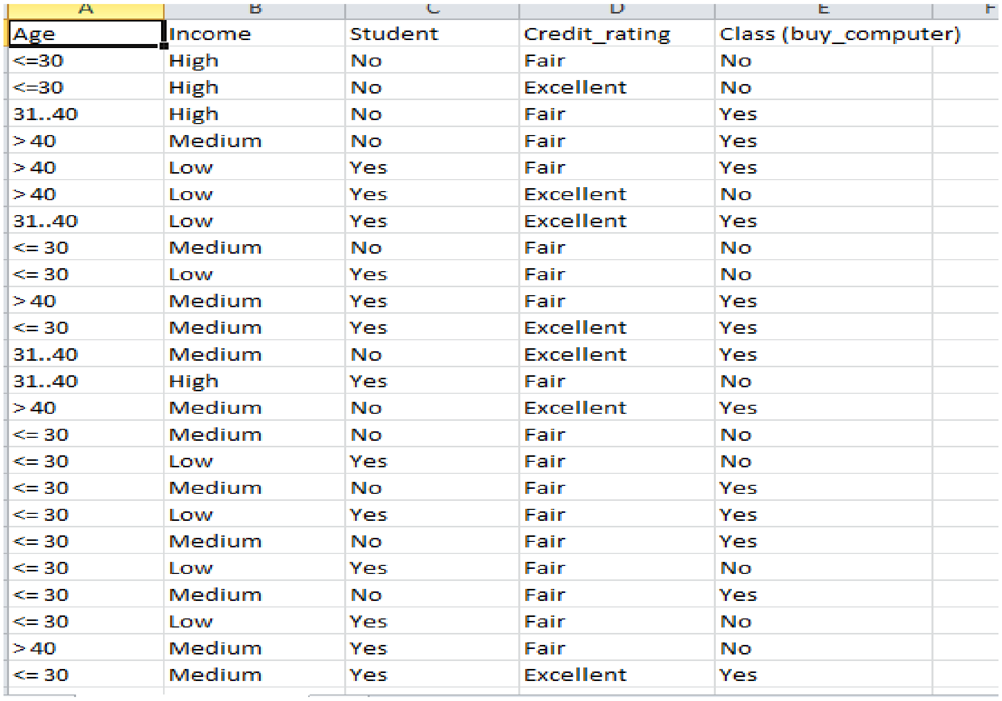
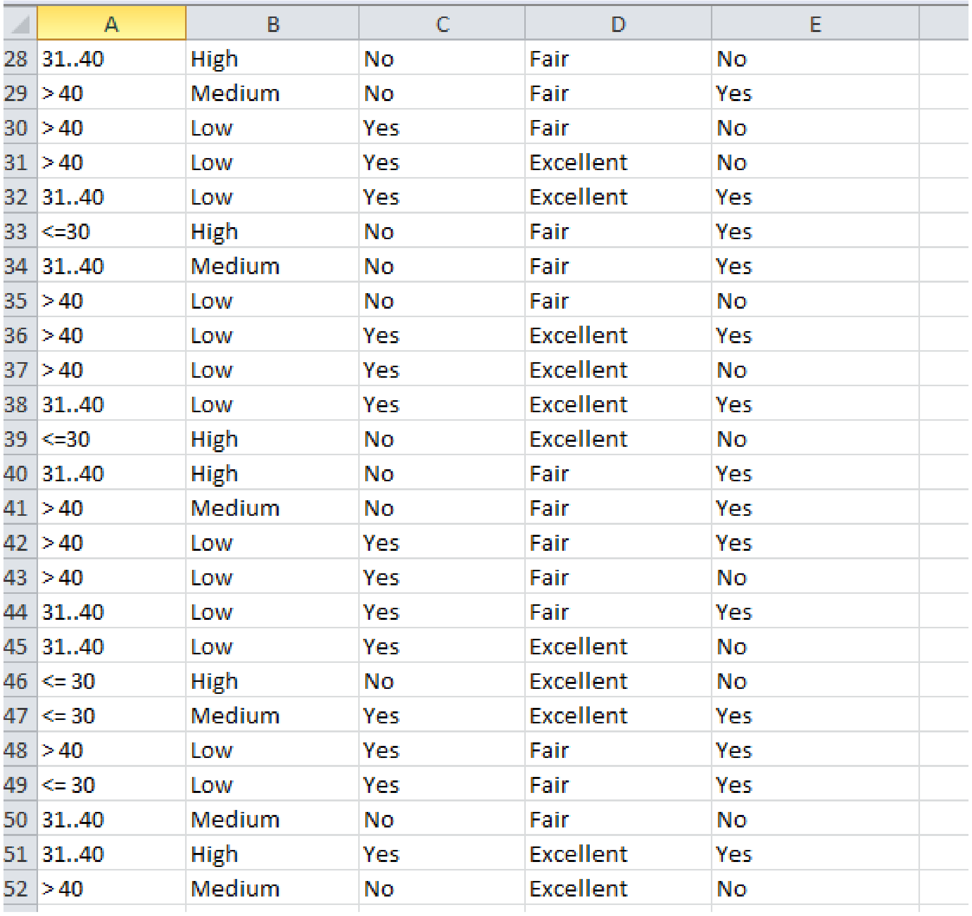
1. Data Set dan Hasil SS Jawaban dari Jupyter Notebook No 1

❖ Data Set No 1

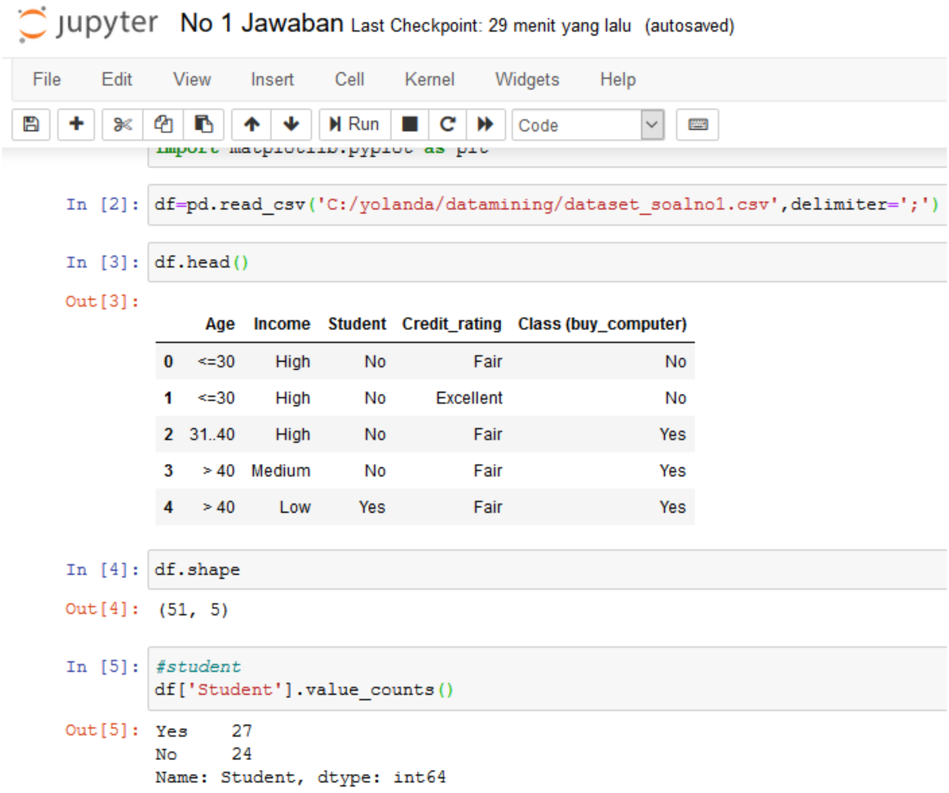


Age	Income	Student	Credit_rating	Class (buy_computer)
<=30	High	No	Fair	No
<=30	High	No	Excellent	No
31..40	High	No	Fair	Yes
> 40	Medium	No	Fair	Yes
> 40	Low	Yes	Fair	Yes
> 40	Low	Yes	Excellent	No
31..40	Low	Yes	Excellent	Yes
<= 30	Medium	No	Fair	No
<= 30	Low	Yes	Fair	No
> 40	Medium	Yes	Fair	Yes
<= 30	Medium	Yes	Excellent	Yes
31..40	Medium	No	Excellent	Yes
31..40	High	Yes	Fair	No
> 40	Medium	No	Excellent	Yes
<= 30	Medium	No	Fair	No
<= 30	Low	Yes	Fair	No
<= 30	Medium	No	Fair	Yes
<= 30	Low	Yes	Fair	Yes
<= 30	Medium	No	Fair	Yes
<= 30	Low	Yes	Fair	No
<= 30	Medium	No	Fair	Yes
<= 30	Low	Yes	Fair	No
> 40	Medium	Yes	Fair	No
<= 30	Medium	Yes	Excellent	Yes



	A	B	C	D	E
28	31..40	High	No	Fair	No
29	> 40	Medium	No	Fair	Yes
30	> 40	Low	Yes	Fair	No
31	> 40	Low	Yes	Excellent	No
32	31..40	Low	Yes	Excellent	Yes
33	<=30	High	No	Fair	Yes
34	31..40	Medium	No	Fair	Yes
35	> 40	Low	No	Fair	No
36	> 40	Low	Yes	Excellent	Yes
37	> 40	Low	Yes	Excellent	No
38	31..40	Low	Yes	Excellent	Yes
39	<=30	High	No	Excellent	No
40	31..40	High	No	Fair	Yes
41	> 40	Medium	No	Fair	Yes
42	> 40	Low	Yes	Fair	Yes
43	> 40	Low	Yes	Fair	No
44	31..40	Low	Yes	Fair	Yes
45	31..40	Low	Yes	Excellent	No
46	<= 30	High	No	Excellent	No
47	<= 30	Medium	Yes	Excellent	Yes
48	> 40	Low	Yes	Fair	Yes
49	<= 30	Low	Yes	Fair	Yes
50	31..40	Medium	No	Fair	No
51	31..40	High	Yes	Excellent	Yes
52	> 40	Medium	No	Excellent	No

❖ Hasil SS dari Jawaban No 1


jupyter
No 1 Jawaban Last Checkpoint: 29 menit yang lalu (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
import matplotlib.pyplot as plt
```

```
In [2]: df=pd.read_csv('C:/yolanda/datamining/dataset_soalno1.csv',delimiter=';')
```

```
In [3]: df.head()
```

Out[3]:

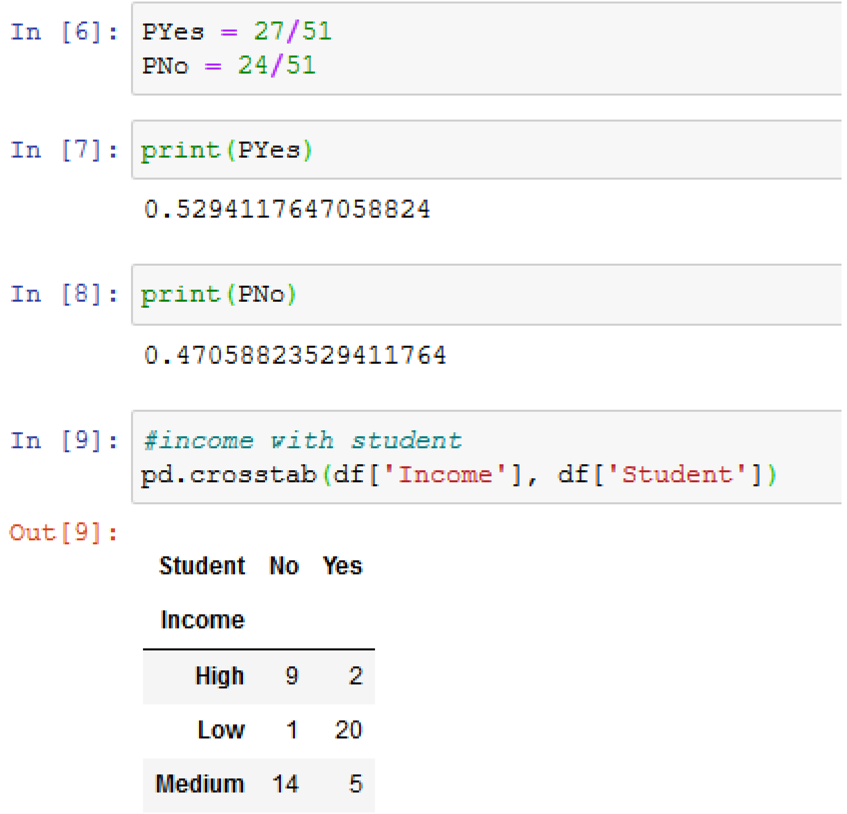
	Age	Income	Student	Credit_rating	Class (buy_computer)
0	<=30	High	No	Fair	No
1	<=30	High	No	Excellent	No
2	31..40	High	No	Fair	Yes
3	> 40	Medium	No	Fair	Yes
4	> 40	Low	Yes	Fair	Yes

```
In [4]: df.shape
```

```
Out[4]: (51, 5)
```

```
In [5]: #student
        df['Student'].value_counts()
```

```
Out[5]: Yes    27
        No     24
        Name: Student, dtype: int64
```



```
In [6]: PYes = 27/51
        PNo = 24/51
```

```
In [7]: print(PYes)

        0.5294117647058824
```

```
In [8]: print(PNo)

        0.47058823529411764
```

```
In [9]: #income with student
        pd.crosstab(df['Income'], df['Student'])
```

Out[9]:

Student	No	Yes
Income		
High	9	2
Low	1	20
Medium	14	5

In [14]:

```
print(PLowYes)
```

```
0.7407407407407407
```

In [15]:

```
print(PMediumYes)
```

```
0.18518518518518517
```

In [16]:

```
print(PHigh)
```

```
0.21568627450980393
```

In [17]:

```
print(PLow)
```

```
0.4117647058823529
```

In [18]:

```
print(PMedium)
```

```
0.37254901960784315
```

In [19]:

```
#credit rating with student
pd.crosstab(df['Credit_rating'], df['Student'])
```

Out[19]:

Student	No	Yes
Credit_rating		
Excellent	8	12
Fair	16	15

In [20]:

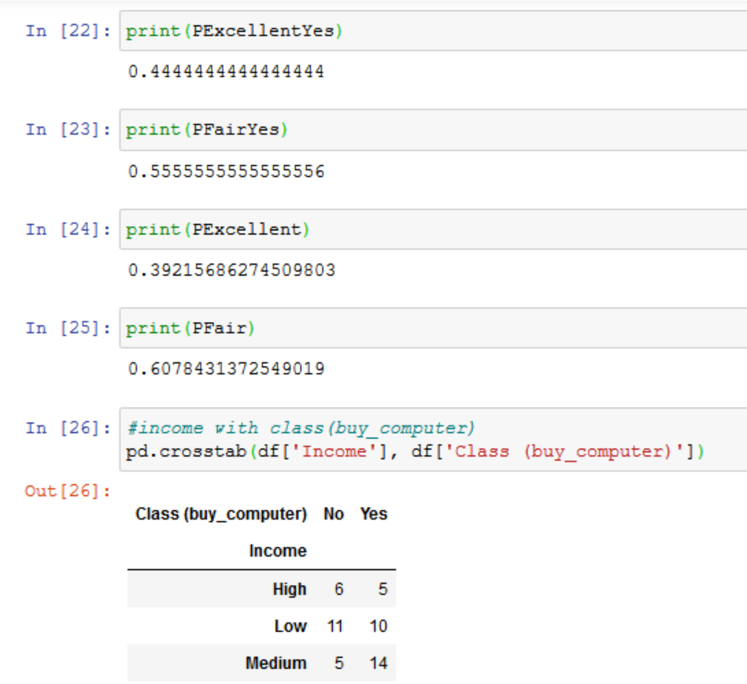
```
PExcellentNo = 8/24
PFairNo = 16/24

PExcellentYes = 12/27
PFairYes = 15/27

PExcellent = 20/51
PFair = 31/51

print(PExcellentNo)
```

```
0.3333333333333333
```

In [21]:

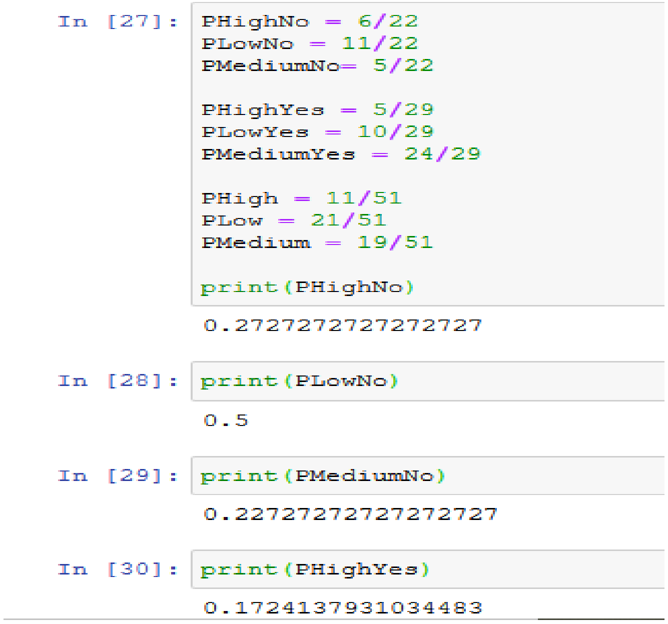
```
print(PFairNo)
```

```
0.6666666666666666
```

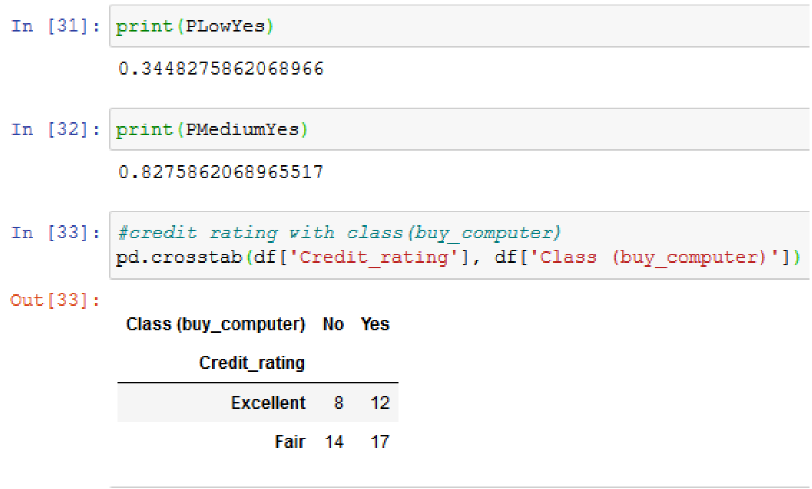


```
In [22]: print(PExcellentYes)

         0.4444444444444444

In [23]: print(PFairYes)

         0.5555555555555556

In [24]: print(PExcellent)

         0.39215686274509803

In [25]: print(PFair)

         0.6078431372549019

In [26]: #income with class(buy_computer)
         pd.crosstab(df['Income'], df['Class (buy_computer)'])

Out[26]:
```

Class (buy_computer)	No	Yes
Income		
High	6	5
Low	11	10
Medium	5	14

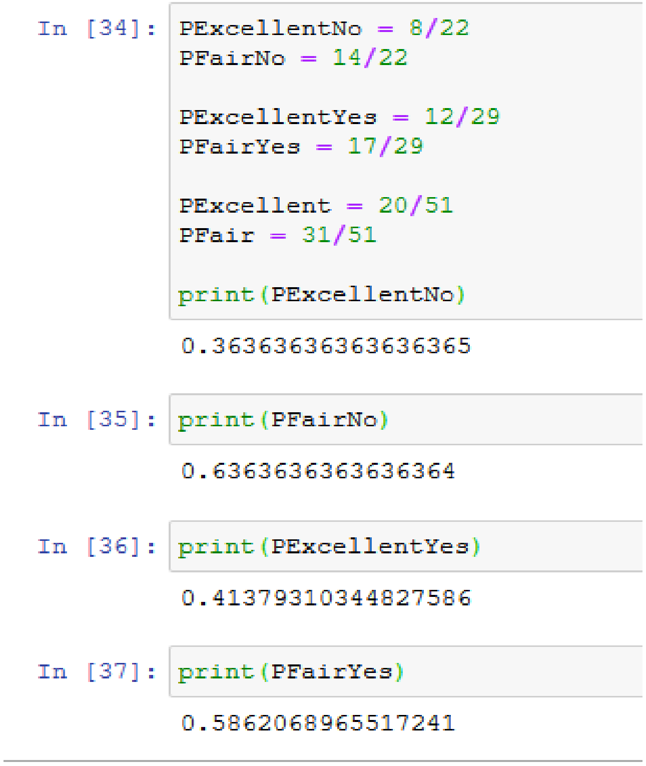


```
In [27]: PHighNo = 6/22
         PLowNo = 11/22
         PMediumNo= 5/22

         PHighYes = 5/29
         PLowYes = 10/29
         PMediumYes = 24/29

         PHigh = 11/51
         PLow = 21/51
         PMedium = 19/51

         print(PHighNo)

         0.2727272727272727

In [28]: print(PLowNo)

         0.5

In [29]: print(PMediumNo)

         0.22727272727272727

In [30]: print(PHighYes)

         0.1724137931034483
```



```
In [31]: print(PLowYes)

         0.3448275862068966

In [32]: print(PMediumYes)

         0.8275862068965517

In [33]: #credit rating with class(buy_computer)
         pd.crosstab(df['Credit_rating'], df['Class (buy_computer)'])

Out[33]:
```

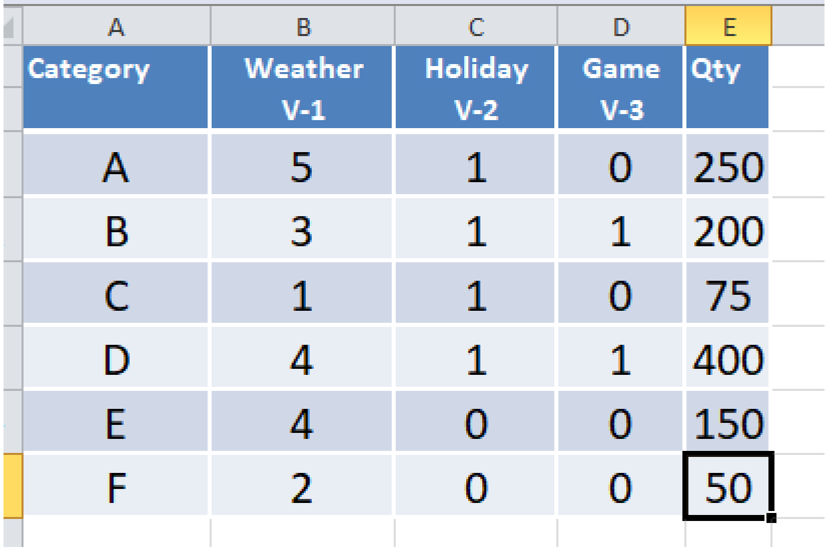
Class (buy_computer)	No	Yes
Credit_rating		
Excellent	8	12
Fair	14	17



```
In [34]: PExcellentNo = 8/22
         PFairNo = 14/22

         PExcellentYes = 12/29
         PFairYes = 17/29

         PExcellent = 20/51
         PFair = 31/51

         print(PExcellentNo)

         0.36363636363636365

In [35]: print(PFairNo)

         0.6363636363636364

In [36]: print(PExcellentYes)

         0.41379310344827586

In [37]: print(PFairYes)

         0.5862068965517241
```

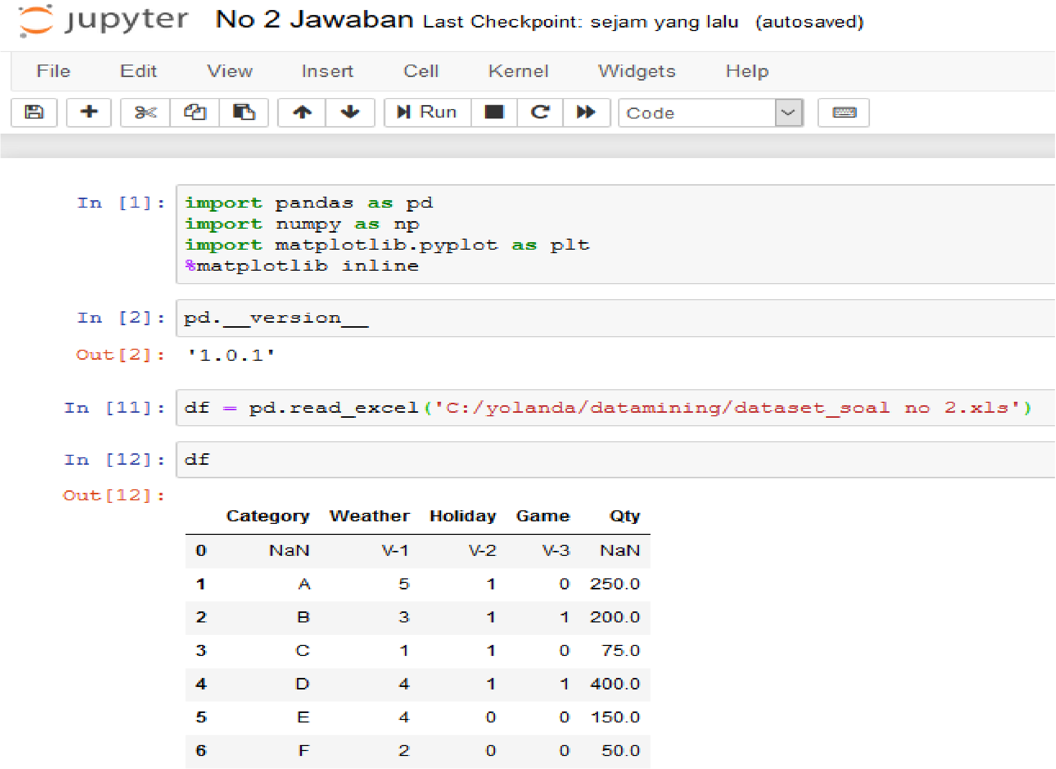
2. Berisi Data set, Hasil SS jawaban dari Jupyter Notebook No 2A

- ❖ Data Set No 2

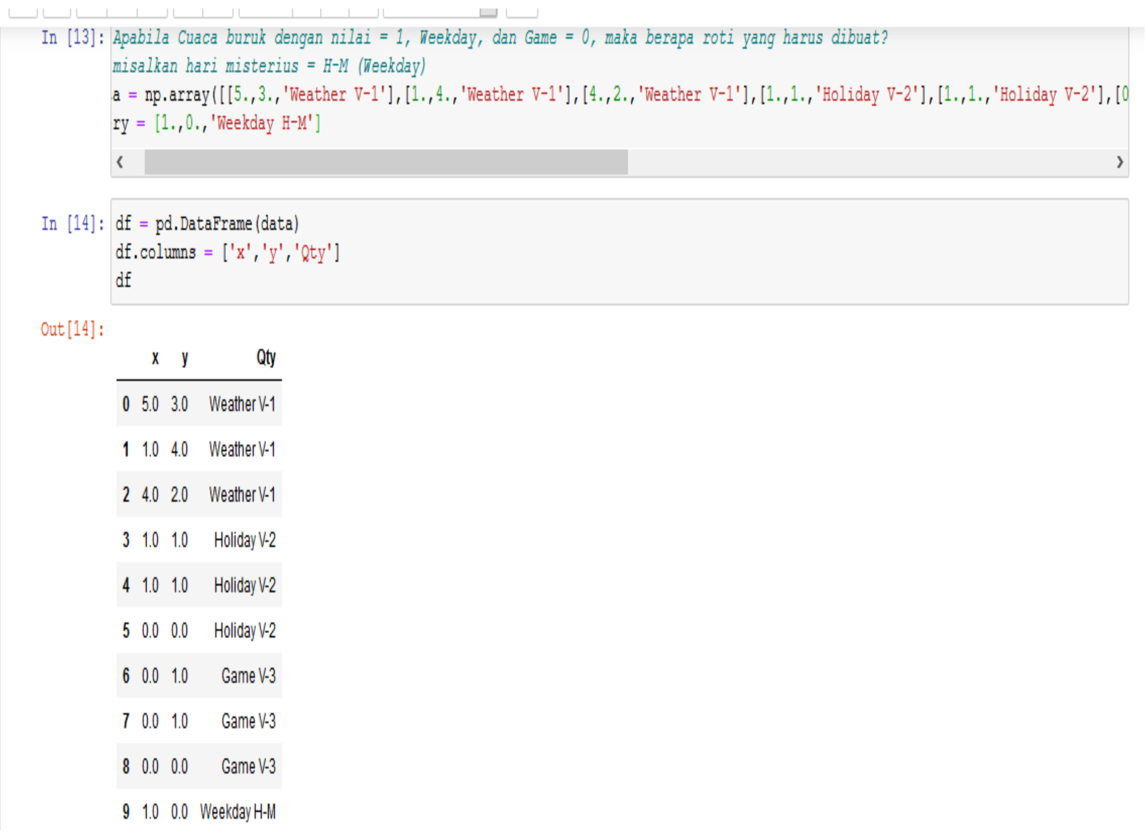


Category	Weather V-1	Holiday V-2	Game V-3	Qty
A	5	1	0	250
B	3	1	1	200
C	1	1	0	75
D	4	1	1	400
E	4	0	0	150
F	2	0	0	50

- ❖ Hasil SS dari Jawaban No 2



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: pd.__version__

Out[2]: '1.0.1'

In [11]: df = pd.read_excel('C:/yolanda/datamining/dataset_soal no 2.xls')

In [12]: df

Out[12]:
   Category  Weather  Holiday  Game    Qty
0       NaN      V-1      V-2   V-3    NaN
1         A        5        1     0  250.0
2         B        3        1     1  200.0
3         C        1        1     0   75.0
4         D        4        1     1  400.0
5         E        4        0     0  150.0
6         F        2        0     0   50.0
```

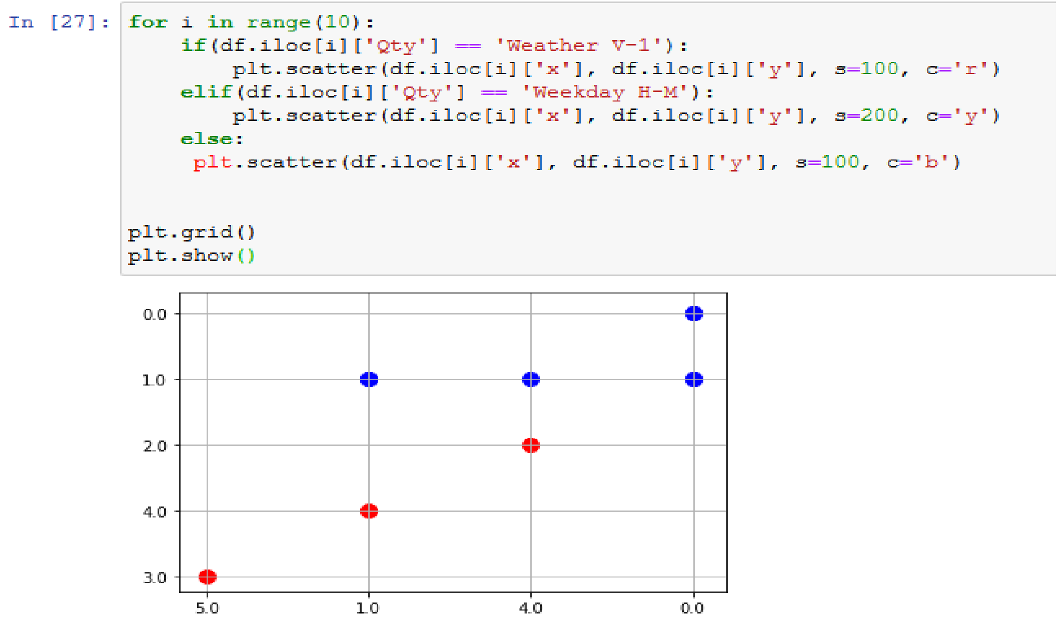


```
In [13]: Apabila Cuaca buruk dengan nilai = 1, Weekday, dan Game = 0, maka berapa roti yang harus dibuat?
         misalkan hari misterius = H-M (Weekday)
         a = np.array([[5.,3.,'Weather V-1'],[1.,4.,'Weather V-1'],[4.,2.,'Weather V-1'],[1.,1.,'Holiday V-2'],[1.,1.,'Holiday V-2'],[0
         ry = [1.,0.,'Weekday H-M']
```

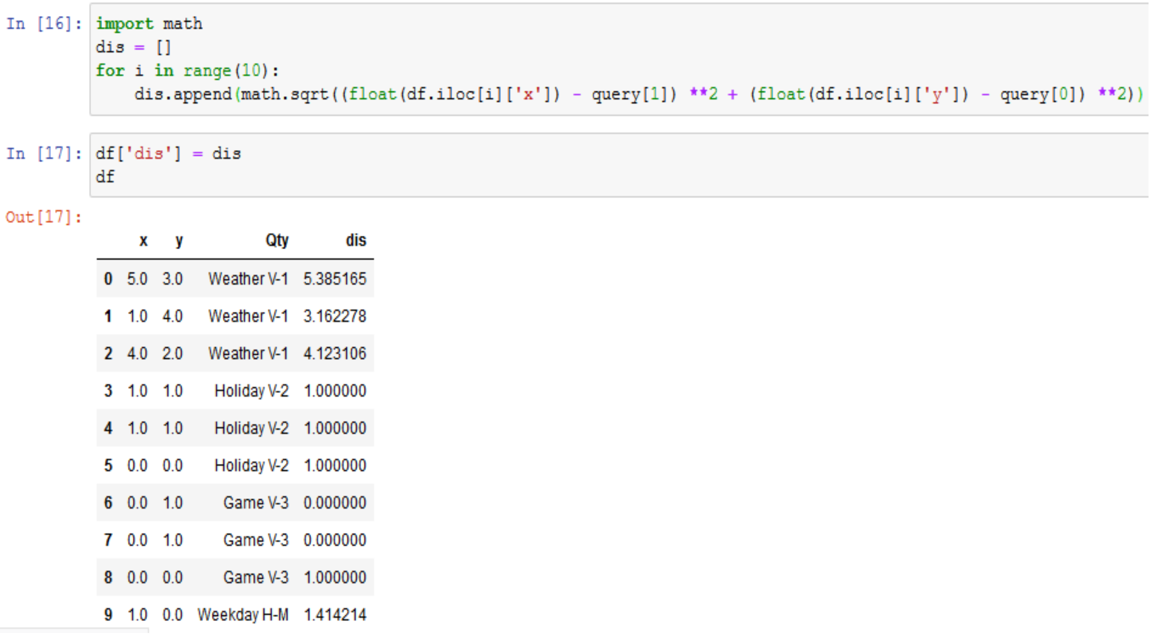
```
In [14]: df = pd.DataFrame(data)
         df.columns = ['x','y','Qty']
         df
```

Out[14]:

	x	y	Qty
0	5.0	3.0	Weather V-1
1	1.0	4.0	Weather V-1
2	4.0	2.0	Weather V-1
3	1.0	1.0	Holiday V-2
4	1.0	1.0	Holiday V-2
5	0.0	0.0	Holiday V-2
6	0.0	1.0	Game V-3
7	0.0	1.0	Game V-3
8	0.0	0.0	Game V-3
9	1.0	0.0	Weekday H-M



```
In [27]: for i in range(10):
             if(df.iloc[i]['Qty'] == 'Weather V-1'):
                 plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=100, c='r')
             elif(df.iloc[i]['Qty'] == 'Weekday H-M'):
                 plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=200, c='y')
             else:
              plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=100, c='b')


         plt.grid()
         plt.show()
```

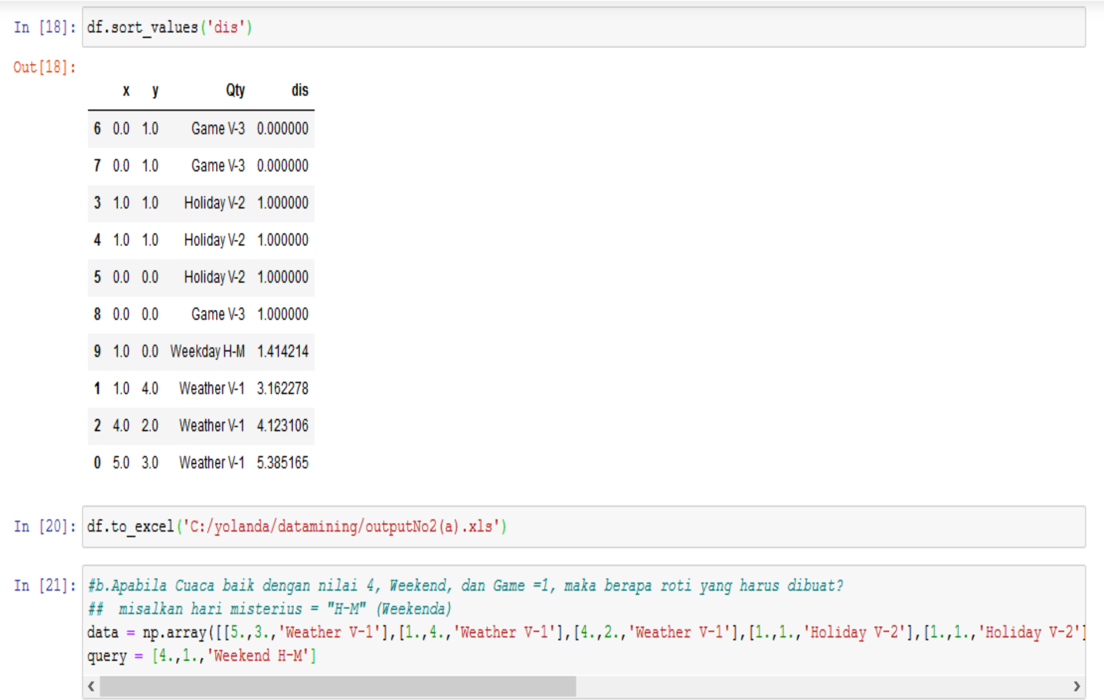


```
In [16]: import math
         dis = []
         for i in range(10):
             dis.append(math.sqrt((float(df.iloc[i]['x']) - query[1]) **2 + (float(df.iloc[i]['y']) - query[0]) **2))
```

```
In [17]: df['dis'] = dis
         df
```

Out[17]:

	x	y	Qty	dis
0	5.0	3.0	Weather V-1	5.385165
1	1.0	4.0	Weather V-1	3.162278
2	4.0	2.0	Weather V-1	4.123106
3	1.0	1.0	Holiday V-2	1.000000
4	1.0	1.0	Holiday V-2	1.000000
5	0.0	0.0	Holiday V-2	1.000000
6	0.0	1.0	Game V-3	0.000000
7	0.0	1.0	Game V-3	0.000000
8	0.0	0.0	Game V-3	1.000000
9	1.0	0.0	Weekday H-M	1.414214



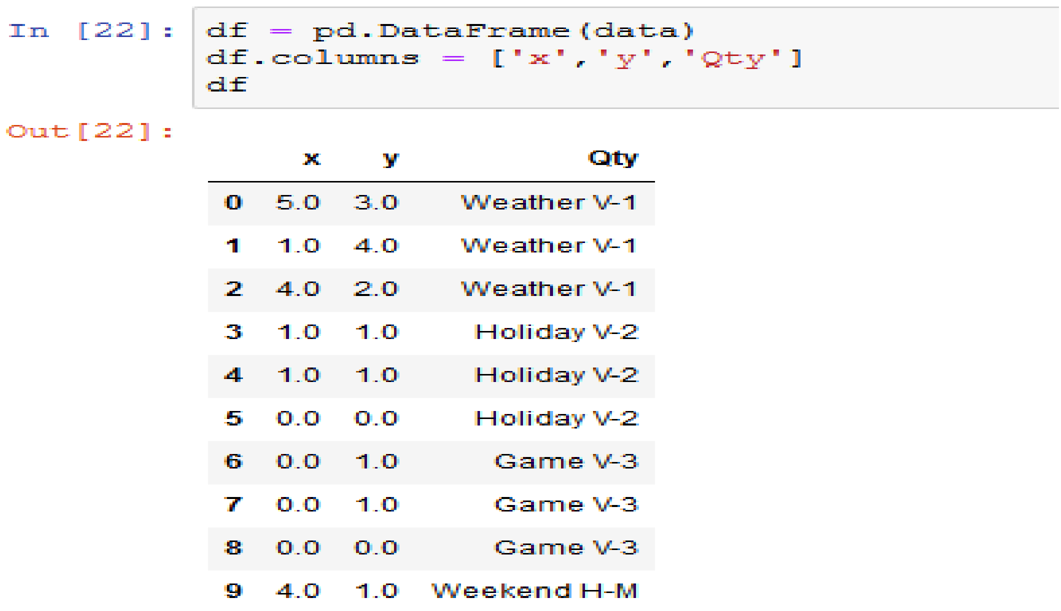
```
In [18]: df.sort_values('dis')
```

Out[18]:

	x	y	Qty	dis
6	0.0	1.0	Game V-3	0.000000
7	0.0	1.0	Game V-3	0.000000
3	1.0	1.0	Holiday V-2	1.000000
4	1.0	1.0	Holiday V-2	1.000000
5	0.0	0.0	Holiday V-2	1.000000
8	0.0	0.0	Game V-3	1.000000
9	1.0	0.0	Weekday H-M	1.414214
1	1.0	4.0	Weather V-1	3.162278
2	4.0	2.0	Weather V-1	4.123106
0	5.0	3.0	Weather V-1	5.385165

```
In [20]: df.to_excel('C:/yolanda/datamining/outputNo2(a).xls')
```

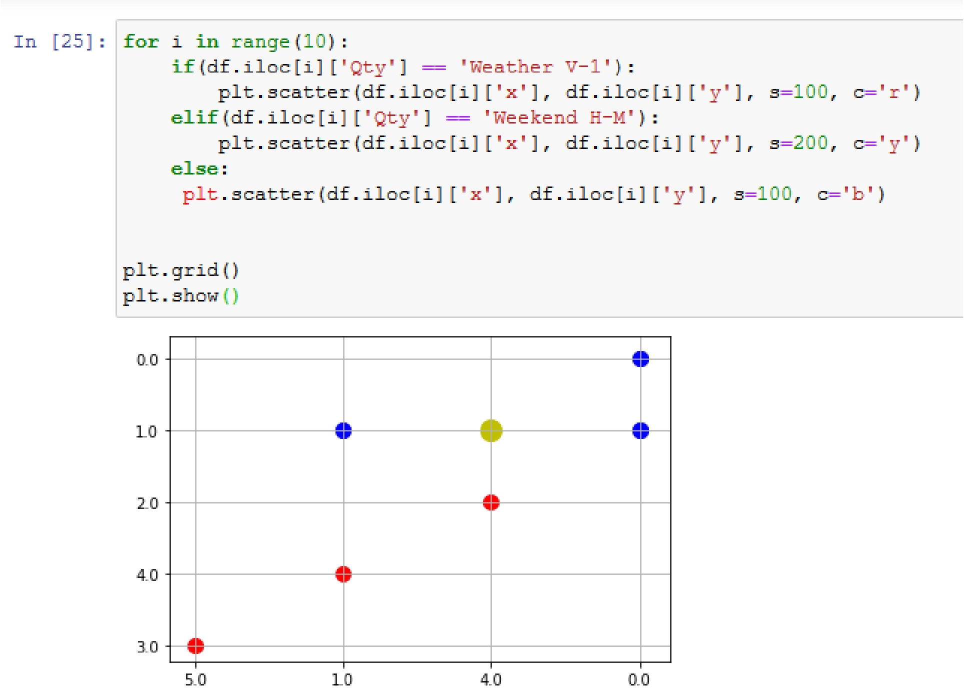
```
In [21]: #b.Apabila Cuaca baik dengan nilai 4, Weekend, dan Game =1, maka berapa roti yang harus dibuat?
         ##  misalkan hari misterius = "H-M" (Weekenda)
         data = np.array([[5.,3.,'Weather V-1'],[1.,4.,'Weather V-1'],[4.,2.,'Weather V-1'],[1.,1.,'Holiday V-2'],[1.,1.,'Holiday V-2']
         query = [4.,1.,'Weekend H-M']
```



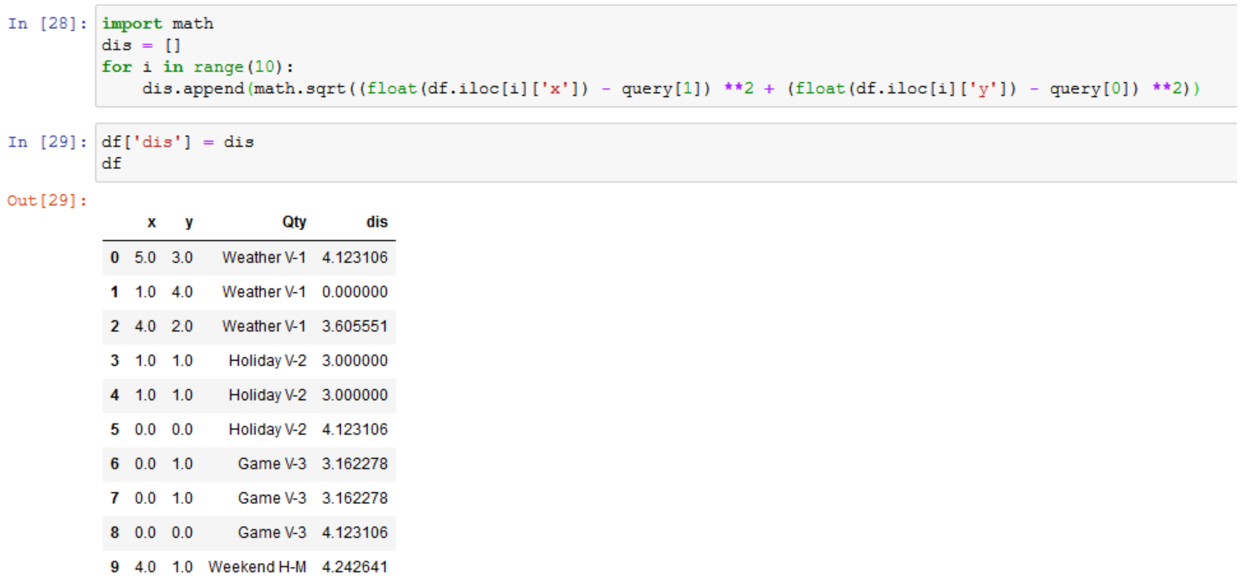
```
In [22]: df = pd.DataFrame(data)
         df.columns = ['x','y','Qty']
         df
```

Out[22]:

	x	y	Qty
0	5.0	3.0	Weather V-1
1	1.0	4.0	Weather V-1
2	4.0	2.0	Weather V-1
3	1.0	1.0	Holiday V-2
4	1.0	1.0	Holiday V-2
5	0.0	0.0	Holiday V-2
6	0.0	1.0	Game V-3
7	0.0	1.0	Game V-3
8	0.0	0.0	Game V-3
9	4.0	1.0	Weekend H-M



```
In [25]: for i in range(10):
             if(df.iloc[i]['Qty'] == 'Weather V-1'):
                 plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=100, c='r')
             elif(df.iloc[i]['Qty'] == 'Weekend H-M'):
                 plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=200, c='y')
             else:
              plt.scatter(df.iloc[i]['x'], df.iloc[i]['y'], s=100, c='b')


         plt.grid()
         plt.show()
```



In [28]:

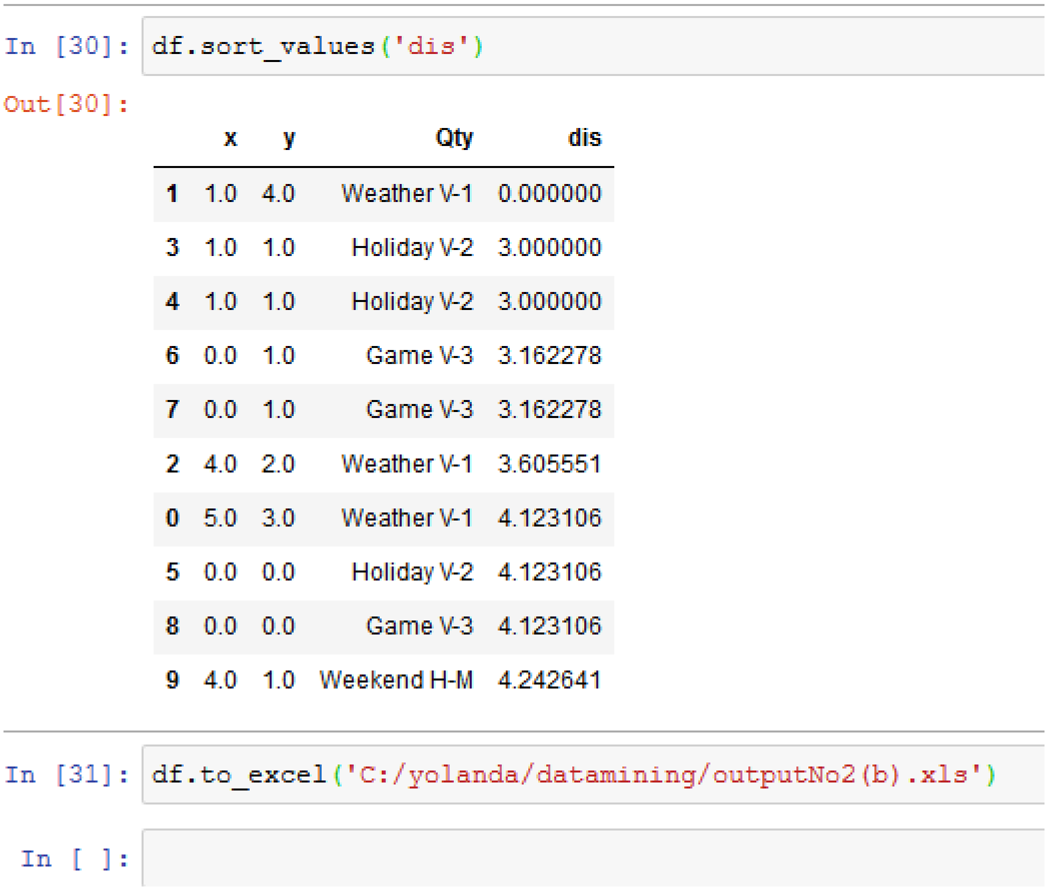
```
import math
dis = []
for i in range(10):
    dis.append(math.sqrt((float(df.iloc[i]['x']) - query[1]) **2 + (float(df.iloc[i]['y']) - query[0]) **2))
```

In [29]:

```
df['dis'] = dis
df
```

Out[29]:

	x	y	Qty	dis
0	5.0	3.0	Weather V-1	4.123106
1	1.0	4.0	Weather V-1	0.000000
2	4.0	2.0	Weather V-1	3.605551
3	1.0	1.0	Holiday V-2	3.000000
4	1.0	1.0	Holiday V-2	3.000000
5	0.0	0.0	Holiday V-2	4.123106
6	0.0	1.0	Game V-3	3.162278
7	0.0	1.0	Game V-3	3.162278
8	0.0	0.0	Game V-3	4.123106
9	4.0	1.0	Weekend H-M	4.242641



In [30]:

```
df.sort_values('dis')
```

Out[30]:

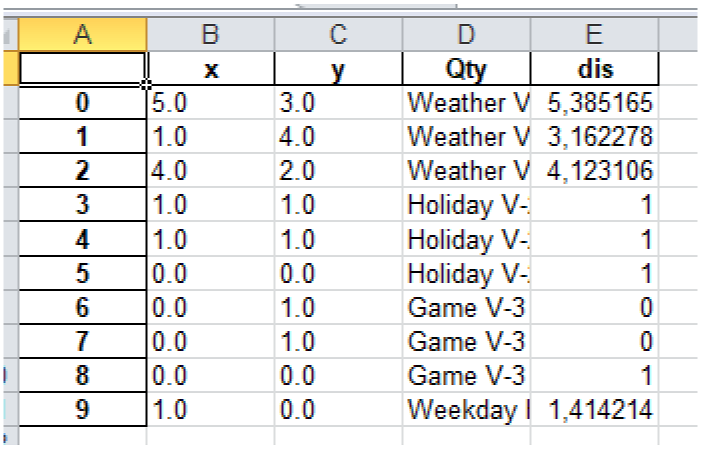
	x	y	Qty	dis
1	1.0	4.0	Weather V-1	0.000000
3	1.0	1.0	Holiday V-2	3.000000
4	1.0	1.0	Holiday V-2	3.000000
6	0.0	1.0	Game V-3	3.162278
7	0.0	1.0	Game V-3	3.162278
2	4.0	2.0	Weather V-1	3.605551
0	5.0	3.0	Weather V-1	4.123106
5	0.0	0.0	Holiday V-2	4.123106
8	0.0	0.0	Game V-3	4.123106
9	4.0	1.0	Weekend H-M	4.242641

In [31]:

```
df.to_excel('C:/yolanda/datamining/outputNo2(b).xls')
```

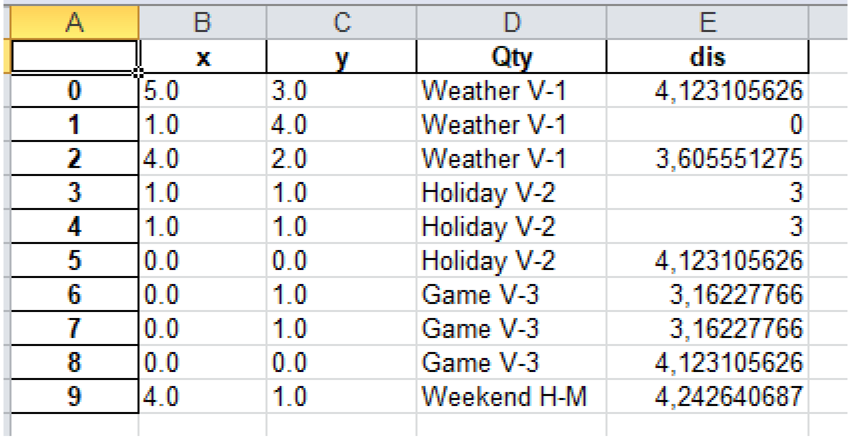
In [ ]:

- ❖ Data Output (2A)



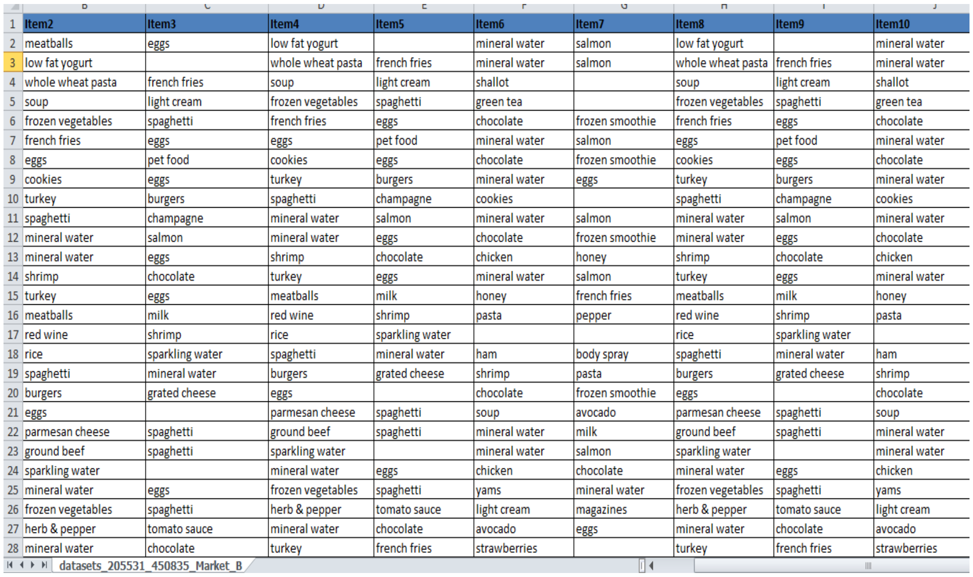
	A	B	C	D	E
		x	y	Qty	dis
	0	5.0	3.0	Weather V	5,385165
	1	1.0	4.0	Weather V	3,162278
	2	4.0	2.0	Weather V	4,123106
	3	1.0	1.0	Holiday V-:	1
	4	1.0	1.0	Holiday V-:	1
	5	0.0	0.0	Holiday V-:	1
	6	0.0	1.0	Game V-3	0
	7	0.0	1.0	Game V-3	0
	8	0.0	0.0	Game V-3	1
	9	1.0	0.0	Weekday I	1,414214

- ❖ Data Output(2A)

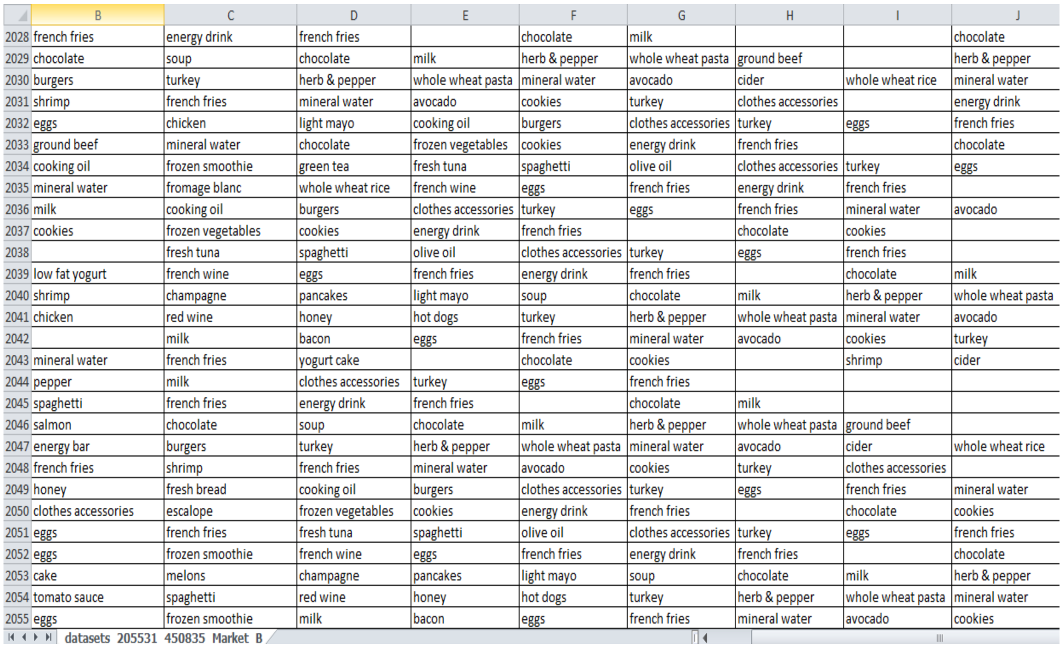


	A	B	C	D	E
		x	y	Qty	dis
	0	5.0	3.0	Weather V-1	4,123105626
	1	1.0	4.0	Weather V-1	0
	2	4.0	2.0	Weather V-1	3,605551275
	3	1.0	1.0	Holiday V-2	3
	4	1.0	1.0	Holiday V-2	3
	5	0.0	0.0	Holiday V-2	4,123105626
	6	0.0	1.0	Game V-3	3,16227766
	7	0.0	1.0	Game V-3	3,16227766
	8	0.0	0.0	Game V-3	4,123105626
	9	4.0	1.0	Weekend H-M	4,242640687

3. Berisi hasil Data set, hasil SS jawaban dari Jupyter Notebook No 3
    - ❖ Data Set No 3

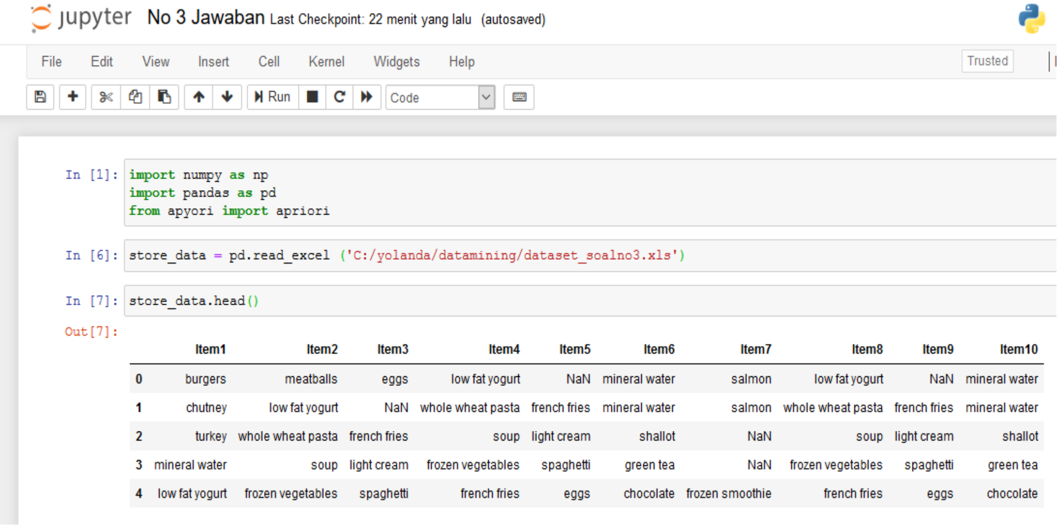


	B	C	D	E	F	G	H	I	J
1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	Item10
2	meatballs	eggs	low fat yogurt		mineral water	salmon	low fat yogurt		mineral water
3	low fat yogurt		whole wheat pasta	french fries	mineral water	salmon	whole wheat pasta	french fries	mineral water
4	whole wheat pasta	french fries	soup	light cream	shallot		soup	light cream	shallot
5	soup	light cream	frozen vegetables	spaghetti	green tea		frozen vegetables	spaghetti	green tea
6	frozen vegetables	spaghetti	french fries	eggs	chocolate	frozen smoothie	french fries	eggs	chocolate
7	french fries	eggs	eggs	pet food	mineral water	salmon	eggs	pet food	mineral water
8	eggs	pet food	cookies	eggs	chocolate	frozen smoothie	cookies	eggs	chocolate
9	cookies	eggs	turkey	burgers	mineral water	eggs	turkey	burgers	mineral water
10	turkey	burgers	spaghetti	champagne	cookies		spaghetti	champagne	cookies
11	spaghetti	champagne	mineral water	salmon	mineral water	salmon	mineral water	salmon	mineral water
12	mineral water	salmon	mineral water	eggs	chocolate	frozen smoothie	mineral water	eggs	chocolate
13	mineral water	eggs	shrimp	chocolate	chicken	honey	shrimp	chocolate	chicken
14	shrimp	chocolate	turkey	eggs	mineral water	salmon	turkey	eggs	mineral water
15	turkey	eggs	meatballs	milk	honey	french fries	meatballs	milk	honey
16	meatballs	milk	red wine	shrimp	pasta	pepper	red wine	shrimp	pasta
17	red wine	shrimp	rice	sparkling water			rice	sparkling water	
18	rice	sparkling water	spaghetti	mineral water	ham	body spray	spaghetti	mineral water	ham
19	spaghetti	mineral water	burgers	grated cheese	shrimp	pasta	burgers	grated cheese	shrimp
20	burgers	grated cheese	eggs		chocolate	frozen smoothie	eggs		chocolate
21	eggs		parmesan cheese	spaghetti	soup	avocado	parmesan cheese	spaghetti	soup
22	parmesan cheese	spaghetti	ground beef	spaghetti	mineral water	milk	ground beef	spaghetti	mineral water
23	ground beef	spaghetti	sparkling water		mineral water	salmon	sparkling water		mineral water
24	sparkling water		mineral water	eggs	chicken	chocolate	mineral water	eggs	chicken
25	mineral water	eggs	frozen vegetables	spaghetti	yams	mineral water	frozen vegetables	spaghetti	yams
26	frozen vegetables	spaghetti	herb & pepper	tomato sauce	light cream	magazines	herb & pepper	tomato sauce	light cream
27	herb & pepper	tomato sauce	mineral water	chocolate	avocado	eggs	mineral water	chocolate	avocado
28	mineral water	chocolate	turkey	french fries	strawberries		turkey	french fries	strawberries



	B	C	D	E	F	G	H	I	J
2028	french fries	energy drink	french fries		chocolate	milk			chocolate
2029	chocolate	soup	chocolate	milk	herb & pepper	whole wheat pasta	ground beef		herb & pepper
2030	burgers	turkey	herb & pepper	whole wheat pasta	mineral water	avocado	cider	whole wheat rice	mineral water
2031	shrimp	french fries	mineral water	avocado	cookies	turkey	clothes accessories		energy drink
2032	eggs	chicken	light mayo	cooking oil	burgers	clothes accessories	turkey	eggs	french fries
2033	ground beef	mineral water	chocolate	frozen vegetables	cookies	energy drink	french fries		chocolate
2034	cooking oil	frozen smoothie	green tea	fresh tuna	spaghetti	olive oil	clothes accessories	turkey	eggs
2035	mineral water	fromage blanc	whole wheat rice	french wine	eggs	french fries	energy drink	french fries	
2036	milk	cooking oil	burgers	clothes accessories	turkey	eggs	french fries	mineral water	avocado
2037	cookies	frozen vegetables	cookies	energy drink	french fries		chocolate	cookies	
2038		fresh tuna	spaghetti	olive oil	clothes accessories	turkey	eggs	french fries	
2039	low fat yogurt	french wine	eggs	french fries	energy drink	french fries		chocolate	milk
2040	shrimp	champagne	pancakes	light mayo	soup	chocolate	milk	herb & pepper	whole wheat pasta
2041	chicken	red wine	honey	hot dogs	turkey	herb & pepper	whole wheat pasta	mineral water	avocado
2042		milk	bacon	eggs	french fries	mineral water	avocado	cookies	turkey
2043	mineral water	french fries	yogurt cake		chocolate	cookies		shrimp	cider
2044	pepper	milk	clothes accessories	turkey	eggs	french fries			
2045	spaghetti	french fries	energy drink	french fries		chocolate	milk		
2046	salmon	chocolate	soup	chocolate	milk	herb & pepper	whole wheat pasta	ground beef	
2047	energy bar	burgers	turkey	herb & pepper	whole wheat pasta	mineral water	avocado	cider	whole wheat rice
2048	french fries	shrimp	french fries	mineral water	avocado	cookies	turkey	clothes accessories	
2049	honey	fresh bread	cooking oil	burgers	clothes accessories	turkey	eggs	french fries	mineral water
2050	clothes accessories	escalope	frozen vegetables	cookies	energy drink	french fries		chocolate	cookies
2051	eggs	french fries	fresh tuna	spaghetti	olive oil	clothes accessories	turkey	eggs	french fries
2052	eggs	frozen smoothie	french wine	eggs	french fries	energy drink	french fries		chocolate
2053	cake	melons	champagne	pancakes	light mayo	soup	chocolate	milk	herb & pepper
2054	tomato sauce	spaghetti	red wine	honey	hot dogs	turkey	herb & pepper	whole wheat pasta	mineral water
2055	eggs	frozen smoothie	milk	bacon	eggs	french fries	mineral water	avocado	cookies

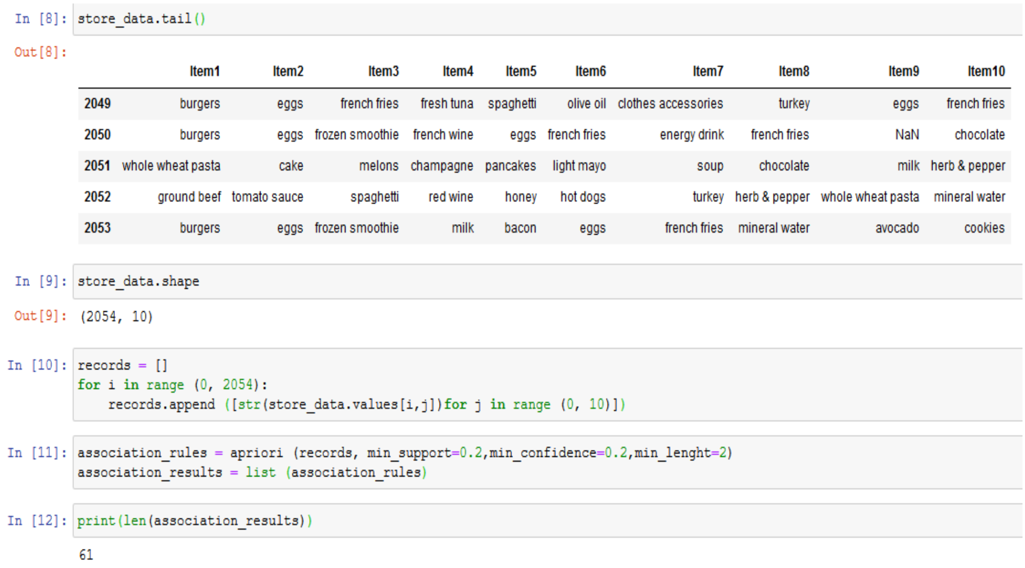
- ❖ Hasil SS dari Jawaban No 3



jupyter
No 3 Jawaban Last Checkpoint: 22 menit yang lalu (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Run
Code

```
In [1]: import numpy as np
        import pandas as pd
        from apyori import apriori

In [6]: store_data = pd.read_excel ('C:/yolanda/datamining/dataset_soalno3.xls')

In [7]: store_data.head()

Out[7]:
```

	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	Item10
0	burgers	meatballs	eggs	low fat yogurt	NaN	mineral water	salmon	low fat yogurt	NaN	mineral water
1	chutney	low fat yogurt	NaN	whole wheat pasta	french fries	mineral water	salmon	whole wheat pasta	french fries	mineral water
2	turkey	whole wheat pasta	french fries	soup	light cream	shallot	NaN	soup	light cream	shallot
3	mineral water	soup	light cream	frozen vegetables	spaghetti	green tea	NaN	frozen vegetables	spaghetti	green tea
4	low fat yogurt	frozen vegetables	spaghetti	french fries	eggs	chocolate	frozen smoothie	french fries	eggs	chocolate

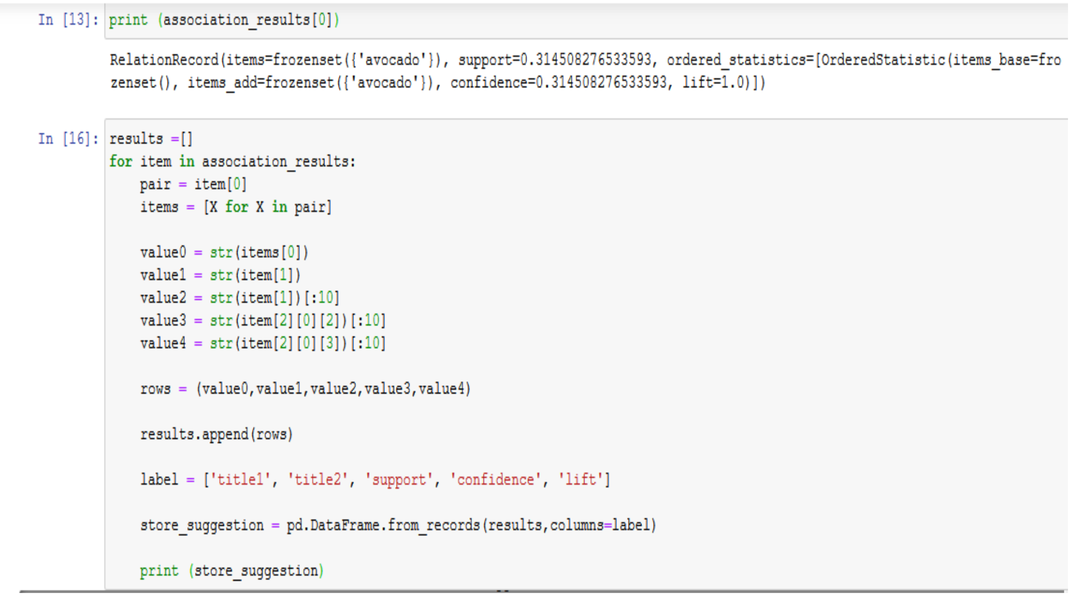


```
In [8]: store_data.tail()
```

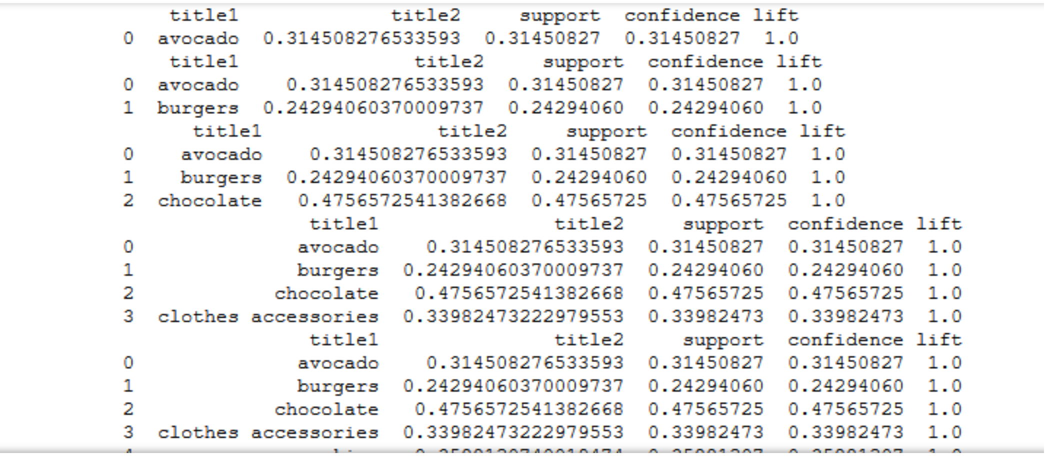
Out[8]:

	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	Item10
2049	burgers	eggs	french fries	fresh tuna	spaghetti	olive oil	clothes accessories	turkey	eggs	french fries
2050	burgers	eggs	frozen smoothie	french wine	eggs	french fries	energy drink	french fries	NaN	chocolate
2051	whole wheat pasta	cake	melons	champagne	pancakes	light mayo	soup	chocolate	milk	herb & pepper
2052	ground beef	tomato sauce	spaghetti	red wine	honey	hot dogs	turkey	herb & pepper	whole wheat pasta	mineral water
2053	burgers	eggs	frozen smoothie	milk	bacon	eggs	french fries	mineral water	avocado	cookies

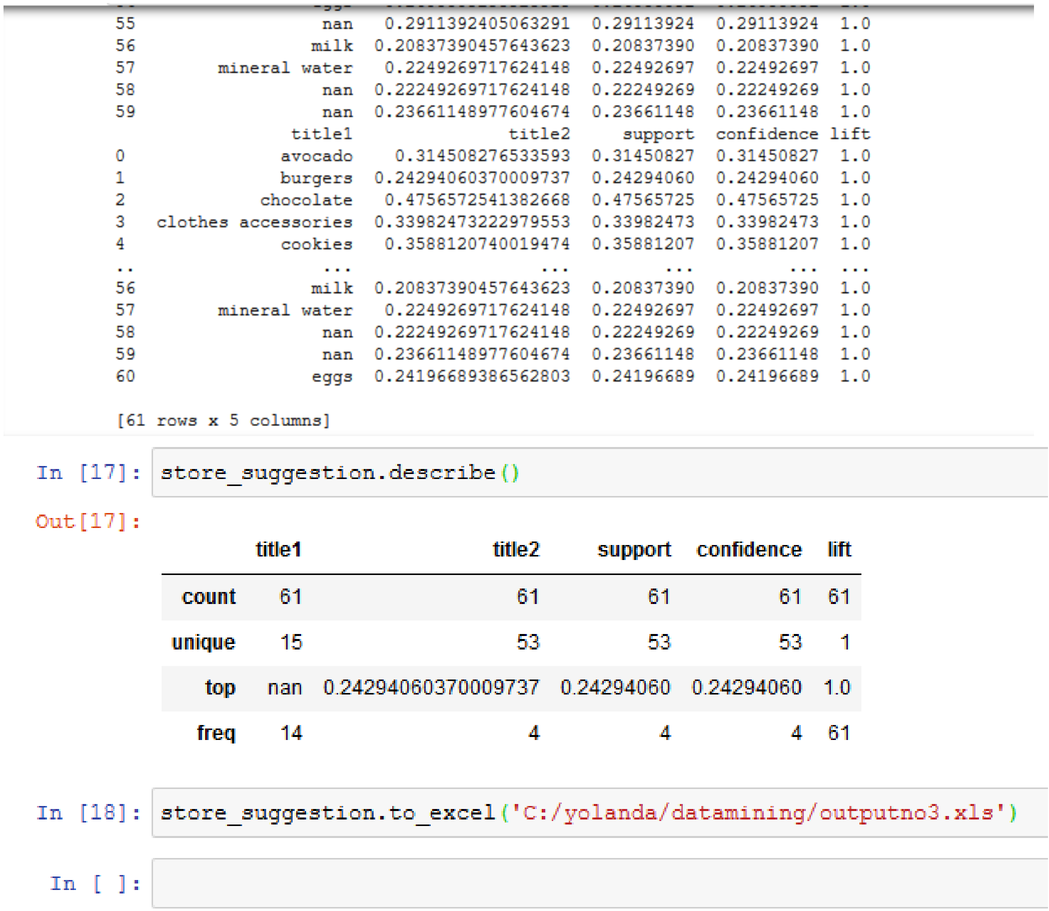
```
In [9]: store_data.shape

Out[9]: (2054, 10)

In [10]: records = []
         for i in range (0, 2054):
             records.append ([str(store_data.values[i,j])for j in range (0, 10)])

In [11]: association_rules = apriori (records, min_support=0.2,min_confidence=0.2,min_lenght=2)
         association_results = list (association_rules)

In [12]: print(len(association_results))

         61
```



```
In [13]: print (association_results[0])

         RelationRecord(items=frozenset({'avocado'}), support=0.314508276533593, ordered_statistics=[OrderedStatistic(items_base=fro
         zenset(), items_add=frozenset({'avocado'}), confidence=0.314508276533593, lift=1.0)])

In [16]: results =[]
         for item in association_results:
             pair = item[0]
             items = [X for X in pair]

             value0 = str(items[0])
             value1 = str(item[1])
             value2 = str(item[1])[:10]
             value3 = str(item[2][0][2])[:10]
             value4 = str(item[2][0][3])[:10]

             rows = (value0,value1,value2,value3,value4)

             results.append(rows)

             label = ['title1', 'title2', 'support', 'confidence', 'lift']

             store_suggestion = pd.DataFrame.from_records(results,columns=label)

             print (store_suggestion)
```



```
    title1             title2     support confidence lift
 0  avocado  0.314508276533593  0.31450827  0.31450827  1.0
    title1             title2     support confidence lift
 0  avocado  0.314508276533593  0.31450827  0.31450827  1.0
 1  burgers 0.24294060370009737  0.24294060  0.24294060  1.0
      title1               title2     support confidence lift
 0    avocado    0.314508276533593  0.31450827  0.31450827  1.0
 1    burgers  0.24294060370009737  0.24294060  0.24294060  1.0
 2  chocolate   0.4756572541382668  0.47565725  0.47565725  1.0
                title1                title2     support confidence lift
 0             avocado     0.314508276533593  0.31450827  0.31450827  1.0
 1             burgers   0.24294060370009737  0.24294060  0.24294060  1.0
 2           chocolate    0.4756572541382668  0.47565725  0.47565725  1.0
 3  clothes accessories  0.33982473222979553  0.33982473  0.33982473  1.0
                title1                title2     support confidence lift
 0             avocado     0.314508276533593  0.31450827  0.31450827  1.0
 1             burgers   0.24294060370009737  0.24294060  0.24294060  1.0
 2           chocolate    0.4756572541382668  0.47565725  0.47565725  1.0
 3  clothes accessories  0.33982473222979553  0.33982473  0.33982473  1.0
```



```
55                 nan   0.2911392405063291  0.29113924  0.29113924  1.0
56                milk  0.20837390457643623  0.20837390  0.20837390  1.0
57       mineral water   0.2249269717624148  0.22492697  0.22492697  1.0
58                 nan  0.22249269717624148  0.22249269  0.22249269  1.0
59                 nan  0.23661148977604674  0.23661148  0.23661148  1.0
                title1               title2     support  confidence lift
0              avocado    0.314508276533593  0.31450827  0.31450827  1.0
1              burgers  0.24294060370009737  0.24294060  0.24294060  1.0
2            chocolate   0.4756572541382668  0.47565725  0.47565725  1.0
3   clothes accessories  0.33982473222979553  0.33982473  0.33982473  1.0
4              cookies   0.3588120740019474  0.35881207  0.35881207  1.0
..                 ...                  ...         ...         ...  ...
56                milk  0.20837390457643623  0.20837390  0.20837390  1.0
57       mineral water   0.2249269717624148  0.22492697  0.22492697  1.0
58                 nan  0.22249269717624148  0.22249269  0.22249269  1.0
59                 nan  0.23661148977604674  0.23661148  0.23661148  1.0
60                eggs  0.24196689386562803  0.24196689  0.24196689  1.0

[61 rows x 5 columns]
```

```
In [17]: store_suggestion.describe()
```

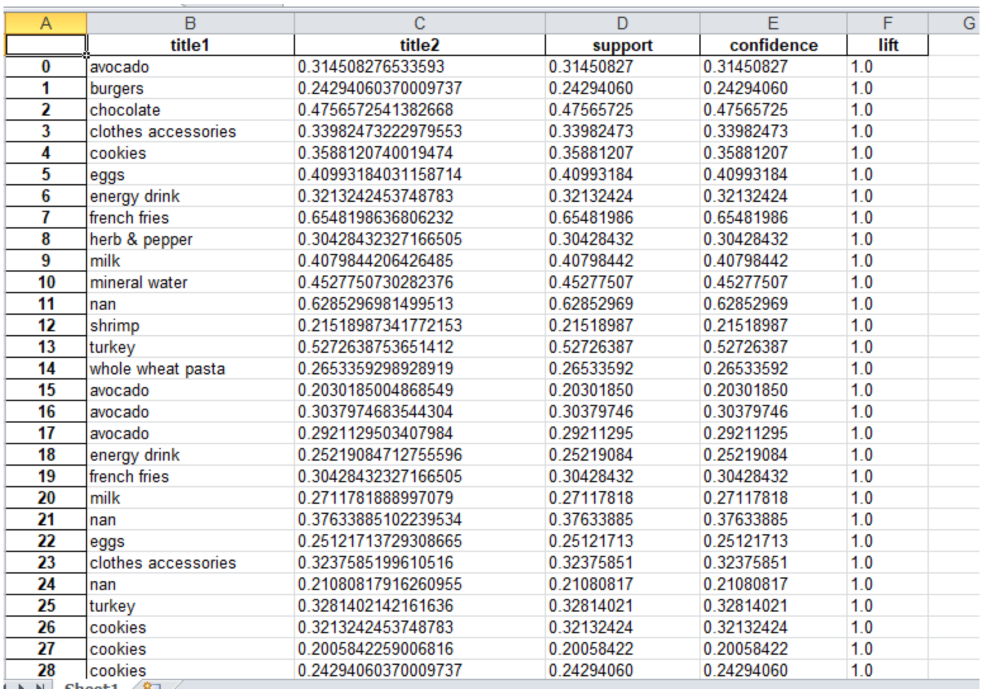
Out[17]:

	title1	title2	support	confidence	lift
count	61	61	61	61	61
unique	15	53	53	53	1
top	nan	0.24294060370009737	0.24294060	0.24294060	1.0
freq	14	4	4	4	61

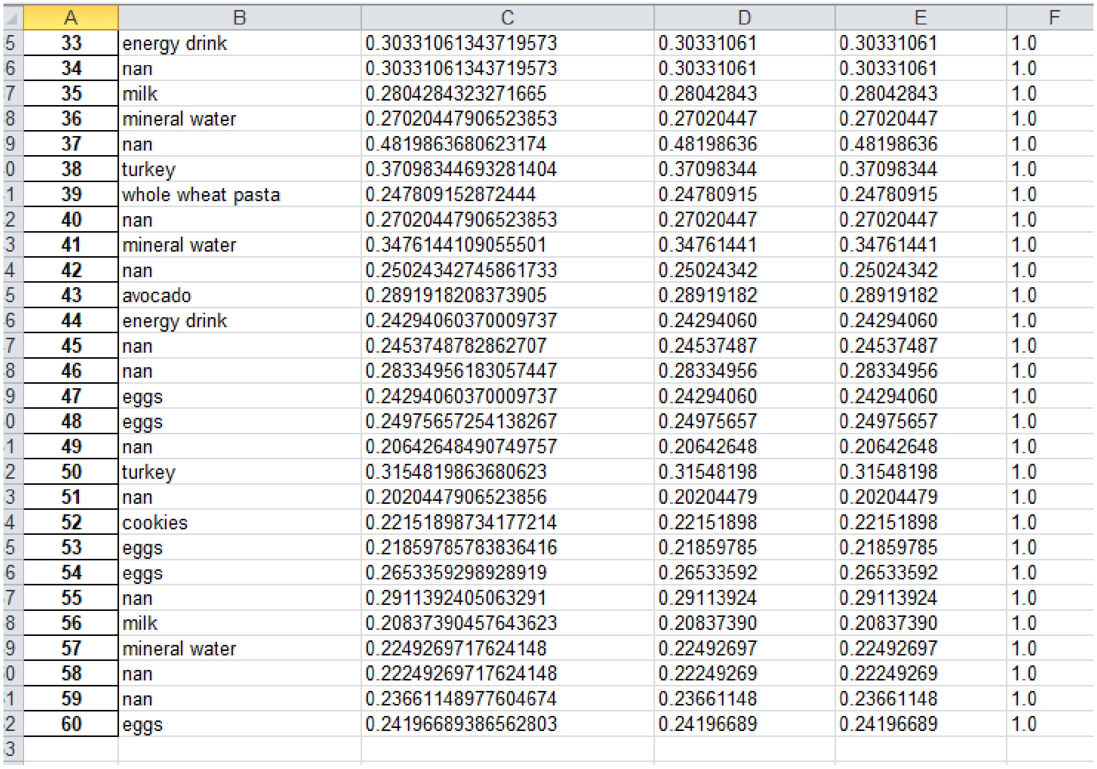
```
In [18]: store_suggestion.to_excel('C:/yolanda/datamining/outputno3.xls')
```

```
In [ ]:
```

- Hasil Output No 3

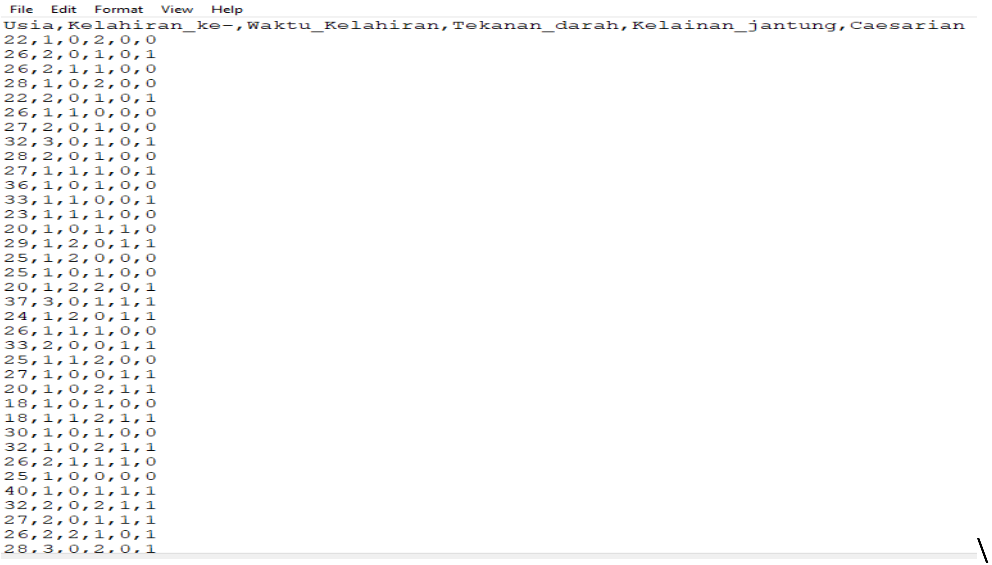


	title1	title2	support	confidence	lift
0	avocado	0.314508276533593	0.31450827	0.31450827	1.0
1	burgers	0.24294060370009737	0.24294060	0.24294060	1.0
2	chocolate	0.4756572541382668	0.47565725	0.47565725	1.0
3	clothes accessories	0.33982473222979553	0.33982473	0.33982473	1.0
4	cookies	0.3588120740019474	0.35881207	0.35881207	1.0
5	eggs	0.40993184031158714	0.40993184	0.40993184	1.0
6	energy drink	0.3213242453748783	0.32132424	0.32132424	1.0
7	french fries	0.6548198636806232	0.65481986	0.65481986	1.0
8	herb & pepper	0.30428432327166505	0.30428432	0.30428432	1.0
9	milk	0.4079844206426485	0.40798442	0.40798442	1.0
10	mineral water	0.4527750730282376	0.45277507	0.45277507	1.0
11	nan	0.6285296981499513	0.62852969	0.62852969	1.0
12	shrimp	0.21518987341772153	0.21518987	0.21518987	1.0
13	turkey	0.5272638753651412	0.52726387	0.52726387	1.0
14	whole wheat pasta	0.2653359298928919	0.26533592	0.26533592	1.0
15	avocado	0.2030185004868549	0.20301850	0.20301850	1.0
16	avocado	0.3037974683544304	0.30379746	0.30379746	1.0
17	avocado	0.2921129503407984	0.29211295	0.29211295	1.0
18	energy drink	0.25219084712755596	0.25219084	0.25219084	1.0
19	french fries	0.30428432327166505	0.30428432	0.30428432	1.0
20	milk	0.2711781888997079	0.27117818	0.27117818	1.0
21	nan	0.37633885102239534	0.37633885	0.37633885	1.0
22	eggs	0.25121713729308665	0.25121713	0.25121713	1.0
23	clothes accessories	0.3237585199610516	0.32375851	0.32375851	1.0
24	nan	0.21080817916260955	0.21080817	0.21080817	1.0
25	turkey	0.3281402142161636	0.32814021	0.32814021	1.0
26	cookies	0.3213242453748783	0.32132424	0.32132424	1.0
27	cookies	0.2005842259006816	0.20058422	0.20058422	1.0
28	cookies	0.24294060370009737	0.24294060	0.24294060	1.0



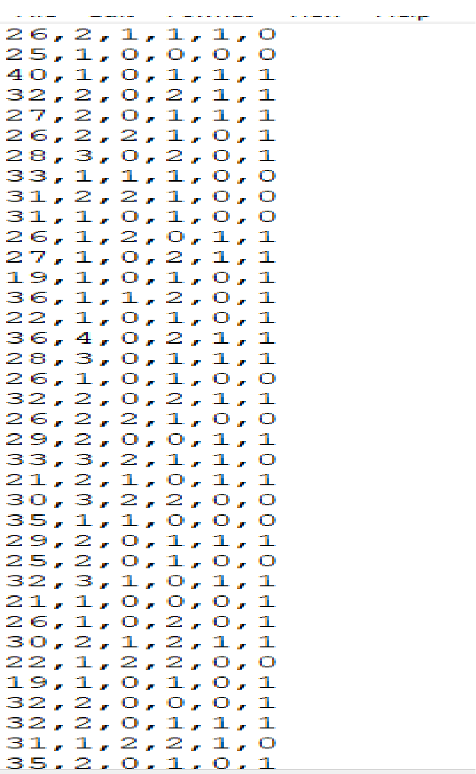
	A	B	C	D	E	F
5	33	energy drink	0.30331061343719573	0.30331061	0.30331061	1.0
6	34	nan	0.30331061343719573	0.30331061	0.30331061	1.0
7	35	milk	0.2804284323271665	0.28042843	0.28042843	1.0
8	36	mineral water	0.27020447906523853	0.27020447	0.27020447	1.0
9	37	nan	0.4819863680623174	0.48198636	0.48198636	1.0
0	38	turkey	0.37098344693281404	0.37098344	0.37098344	1.0
1	39	whole wheat pasta	0.247809152872444	0.24780915	0.24780915	1.0
2	40	nan	0.27020447906523853	0.27020447	0.27020447	1.0
3	41	mineral water	0.3476144109055501	0.34761441	0.34761441	1.0
4	42	nan	0.25024342745861733	0.25024342	0.25024342	1.0
5	43	avocado	0.2891918208373905	0.28919182	0.28919182	1.0
6	44	energy drink	0.24294060370009737	0.24294060	0.24294060	1.0
7	45	nan	0.245374878286270 7	0.24537487	0.24537487	1.0
8	46	nan	0.28334956183057447	0.28334956	0.28334956	1.0
9	47	eggs	0.24294060370009737	0.24294060	0.24294060	1.0
0	48	eggs	0.24975657254138267	0.24975657	0.24975657	1.0
1	49	nan	0.20642648490749757	0.20642648	0.20642648	1.0
2	50	turkey	0.3154819863680623	0.31548198	0.31548198	1.0
3	51	nan	0.2020447906523856	0.20204479	0.20204479	1.0
4	52	cookies	0.22151898734177214	0.22151898	0.22151898	1.0
5	53	eggs	0.21859785783836416	0.21859785	0.21859785	1.0
6	54	eggs	0.2653359298928919	0.26533592	0.26533592	1.0
7	55	nan	0.2911392405063291	0.29113924	0.29113924	1.0
8	56	milk	0.20837390457643623	0.20837390	0.20837390	1.0
9	57	mineral water	0.2249269717624148	0.22492697	0.22492697	1.0
0	58	nan	0.22249269717624148	0.22249269	0.22249269	1.0
1	59	nan	0.23661148977604674	0.23661148	0.23661148	1.0
2	60	eggs	0.24196689386562803	0.24196689	0.24196689	1.0
3						

4. Berisi Data set, hasil SS dari jupyter Notebook dan Output Dari No 4
   - ❖ Data Set

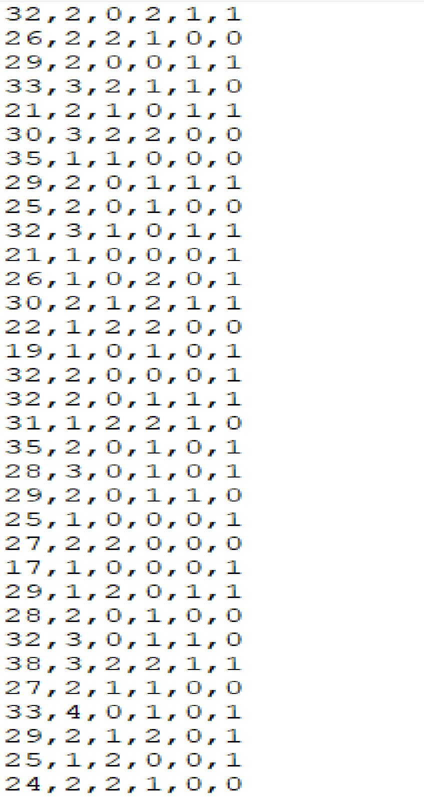


```
Usia,Kelahiran_ke-,Waktu_Kelahiran,Tekanan_darah,Kelainan_jantung,Caesarian
22,1,0,2,0,0
26,2,0,1,0,1
26,2,1,1,0,0
28,1,0,2,0,0
22,2,0,1,0,1
26,1,1,0,0,0
27,2,0,1,0,0
32,3,0,1,0,1
28,2,0,1,0,0
27,1,1,1,0,1
36,1,0,1,0,0
33,1,1,0,0,1
23,1,1,1,0,0
20,1,0,1,1,0
29,1,2,0,1,1
25,1,2,0,0,0
25,1,0,1,0,0
20,1,2,2,0,1
37,3,0,1,1,1
24,1,2,0,1,1
26,1,1,1,0,0
33,2,0,0,1,1
25,1,1,2,0,0
27,1,0,0,1,1
20,1,0,2,1,1
18,1,0,1,0,0
18,1,1,2,1,1
30,1,0,1,0,0
32,1,0,2,1,1
26,2,1,1,1,0
25,1,0,0,0,0
40,1,0,1,1,1
32,2,0,2,1,1
27,2,0,1,1,1
26,2,2,1,0,1
28,3,0,2,0,1
```

\

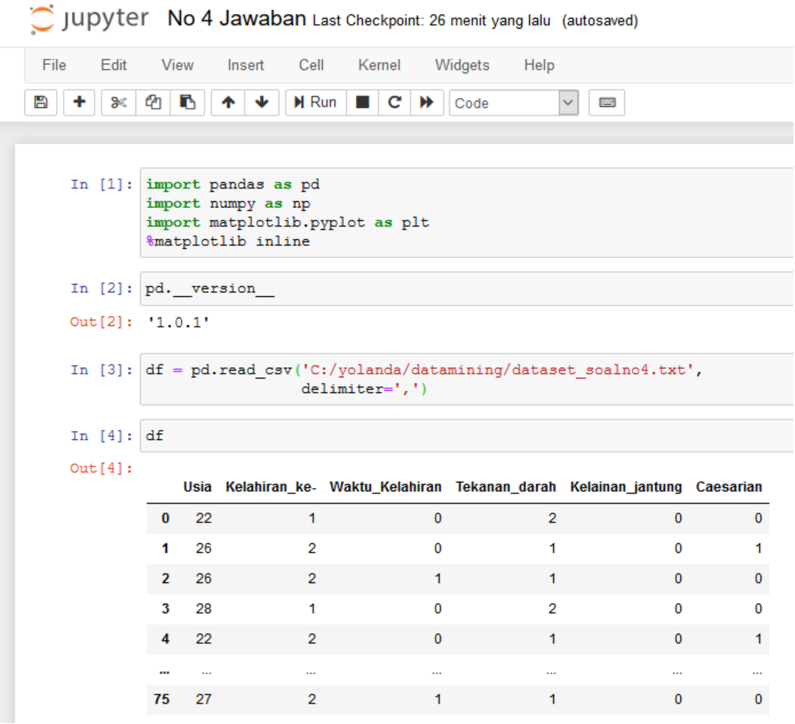


```
26,2,1,1,1,0
25,1,0,0,0,0
40,1,0,1,1,1
32,2,0,2,1,1
27,2,0,1,1,1
26,2,2,1,0,1
28,3,0,2,0,1
33,1,1,1,0,0
31,2,2,1,0,0
31,1,0,1,0,0
26,1,2,0,1,1
27,1,0,2,1,1
19,1,0,1,0,1
36,1,1,2,0,1
22,1,0,1,0,1
36,4,0,2,1,1
28,3,0,1,1,1
26,1,0,1,0,0
32,2,0,2,1,1
26,2,2,1,0,0
29,2,0,0,1,1
33,3,2,1,1,0
21,2,1,0,1,1
30,3,2,2,0,0
35,1,1,0,0,0
29,2,0,1,1,1
25,2,0,1,0,0
32,3,1,0,1,1
21,1,0,0,0,1
26,1,0,2,0,1
30,2,1,2,1,1
22,1,2,2,0,0
19,1,0,1,0,1
32,2,0,0,0,1
32,2,0,1,1,1
31,1,2,2,1,0
35,2,0,1,0,1
```



```
32,2,0,2,1,1
26,2,2,1,0,0
29,2,0,0,1,1
33,3,2,1,1,0
21,2,1,0,1,1
30,3,2,2,0,0
35,1,1,0,0,0
29,2,0,1,1,1
25,2,0,1,0,0
32,3,1,0,1,1
21,1,0,0,0,1
26,1,0,2,0,1
30,2,1,2,1,1
22,1,2,2,0,0
19,1,0,1,0,1
32,2,0,0,0,1
32,2,0,1,1,1
31,1,2,2,1,0
35,2,0,1,0,1
28,3,0,1,0,1
29,2,0,1,1,0
25,1,0,0,0,1
27,2,2,0,0,0
17,1,0,0,0,1
29,1,2,0,1,1
28,2,0,1,0,0
32,3,0,1,1,0
38,3,2,2,1,1
27,2,1,1,0,0
33,4,0,1,0,1
29,2,1,2,0,1
25,1,2,0,0,1
24,2,2,1,0,0
```

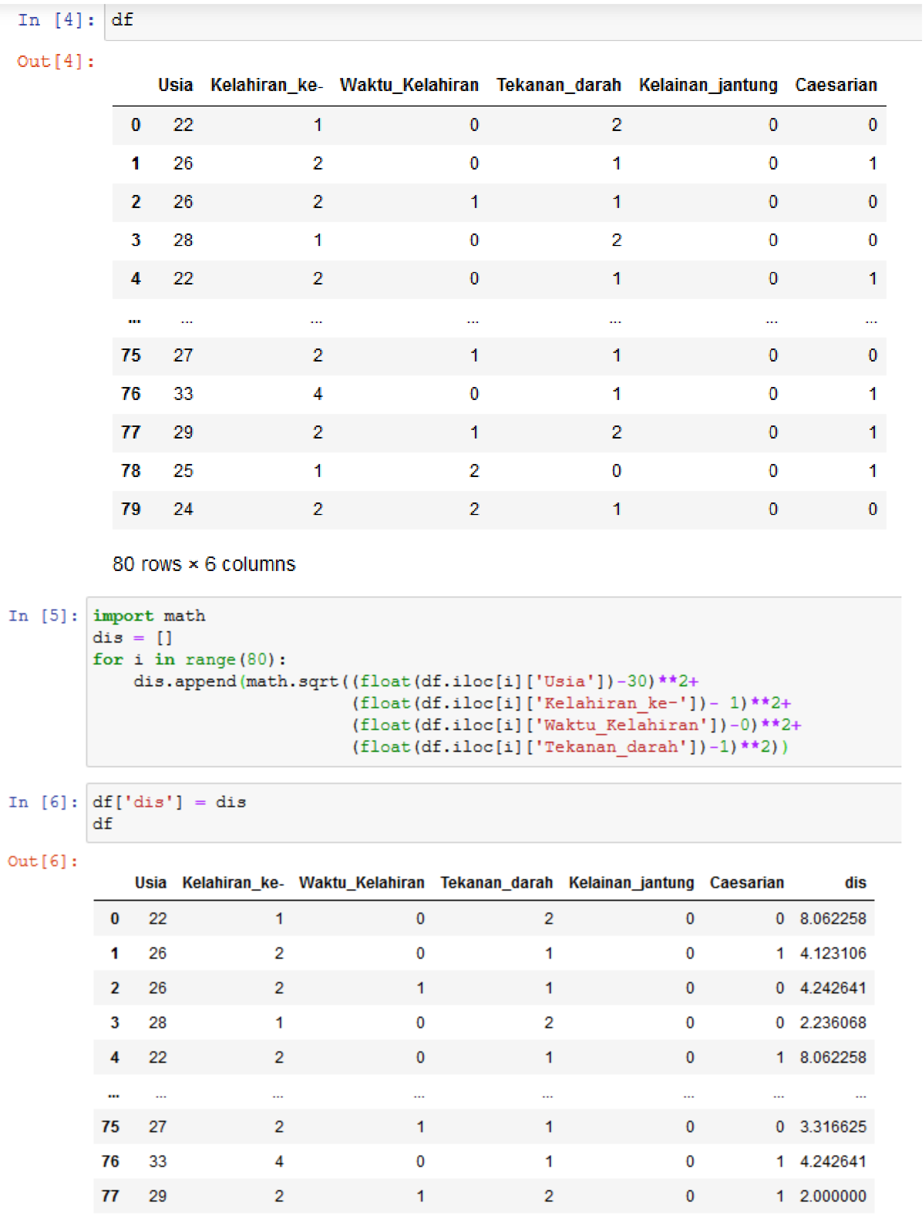
- Hasil SS dari Jawaban dari No 4



jupyter
No 4 Jawaban Last Checkpoint: 26 menit yang lalu (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: pd.__version__

Out[2]: '1.0.1'

In [3]: df = pd.read_csv('C:/yolanda/datamining/dataset_soalno4.txt',
                         delimiter=',')

In [4]: df

Out[4]:
```

	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian
0	22	1	0	2	0	0
1	26	2	0	1	0	1
2	26	2	1	1	0	0
3	28	1	0	2	0	0
4	22	2	0	1	0	1
...	...	...	...	...	...	...
75	27	2	1	1	0	0



In [4]:

```python
df
```

Out[4]:

	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian
0	22	1	0	2	0	0
1	26	2	0	1	0	1
2	26	2	1	1	0	0
3	28	1	0	2	0	0
4	22	2	0	1	0	1
...	...	...	...	...	...	...
75	27	2	1	1	0	0
76	33	4	0	1	0	1
77	29	2	1	2	0	1
78	25	1	2	0	0	1
79	24	2	2	1	0	0

80 rows × 6 columns

In [5]:

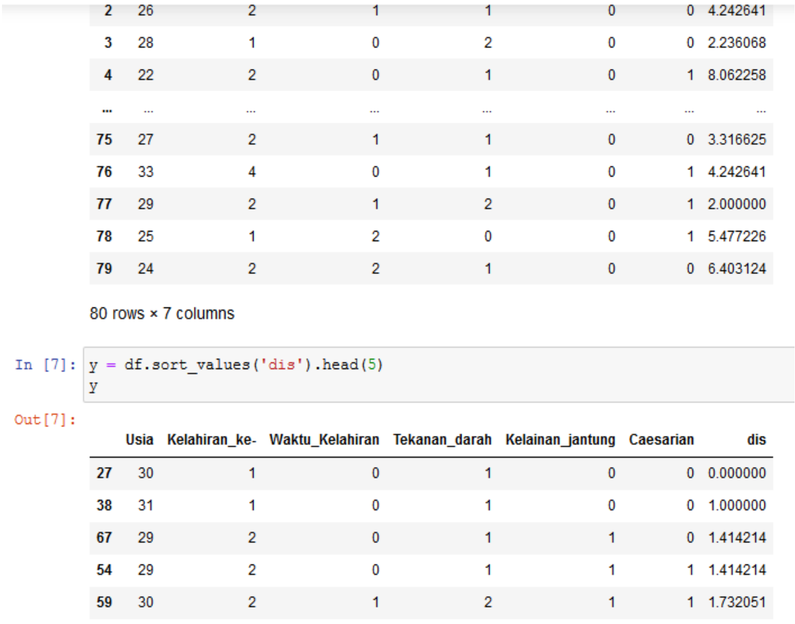
```python
import math
dis = []
for i in range(80):
    dis.append(math.sqrt((float(df.iloc[i]['Usia'])-30)**2+
                         (float(df.iloc[i]['Kelahiran_ke-'])- 1)**2+
                         (float(df.iloc[i]['Waktu_Kelahiran'])-0)**2+
                         (float(df.iloc[i]['Tekanan_darah'])-1)**2))
```

In [6]:

```python
df['dis'] = dis
df
```

Out[6]:

	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian	dis
0	22	1	0	2	0	0	8.062258
1	26	2	0	1	0	1	4.123106
2	26	2	1	1	0	0	4.242641
3	28	1	0	2	0	0	2.236068
4	22	2	0	1	0	1	8.062258
...	...	...	...	...	...	...	...
75	27	2	1	1	0	0	3.316625
76	33	4	0	1	0	1	4.242641
77	29	2	1	2	0	1	2.000000



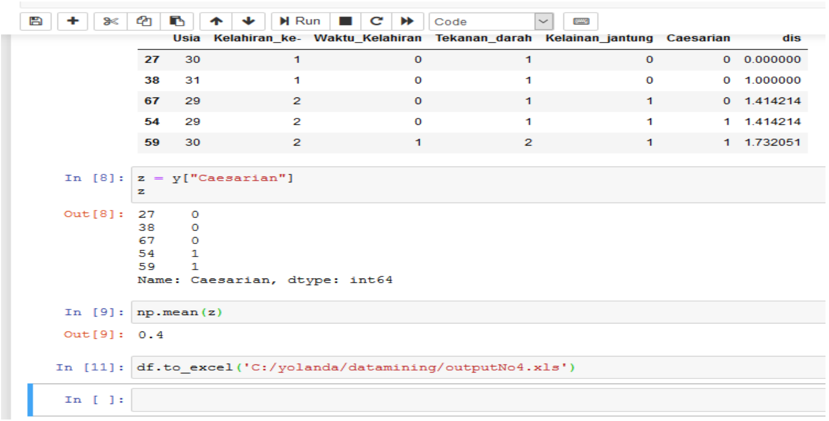
2	26	2	1	1	0	0	4.242641
3	28	1	0	2	0	0	2.236068
4	22	2	0	1	0	1	8.062258
...	...	...	...	...	...	...	...
75	27	2	1	1	0	0	3.316625
76	33	4	0	1	0	1	4.242641
77	29	2	1	2	0	1	2.000000
78	25	1	2	0	0	1	5.477226
79	24	2	2	1	0	0	6.403124

80 rows × 7 columns

```
In [7]: y = df.sort_values('dis').head(5)
        y
```

Out[7]:

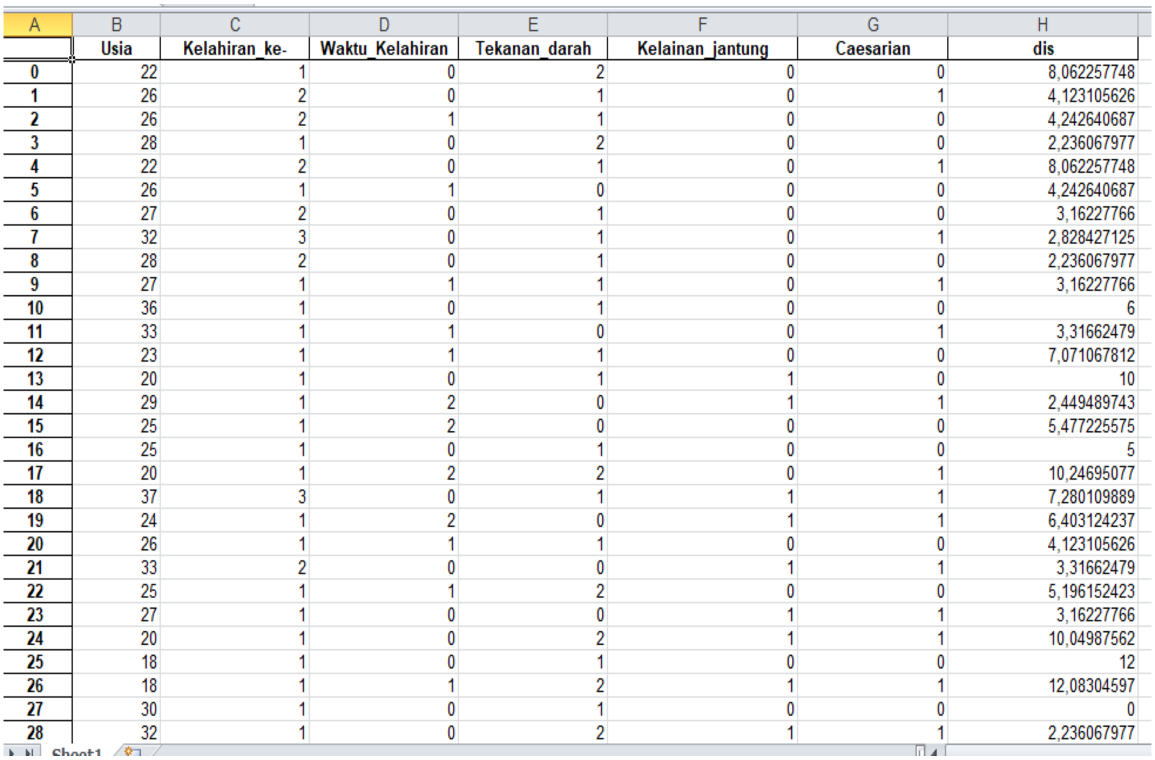
	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian	dis
27	30	1	0	1	0	0	0.000000
38	31	1	0	1	0	0	1.000000
67	29	2	0	1	1	0	1.414214
54	29	2	0	1	1	1	1.414214
59	30	2	1	2	1	1	1.732051



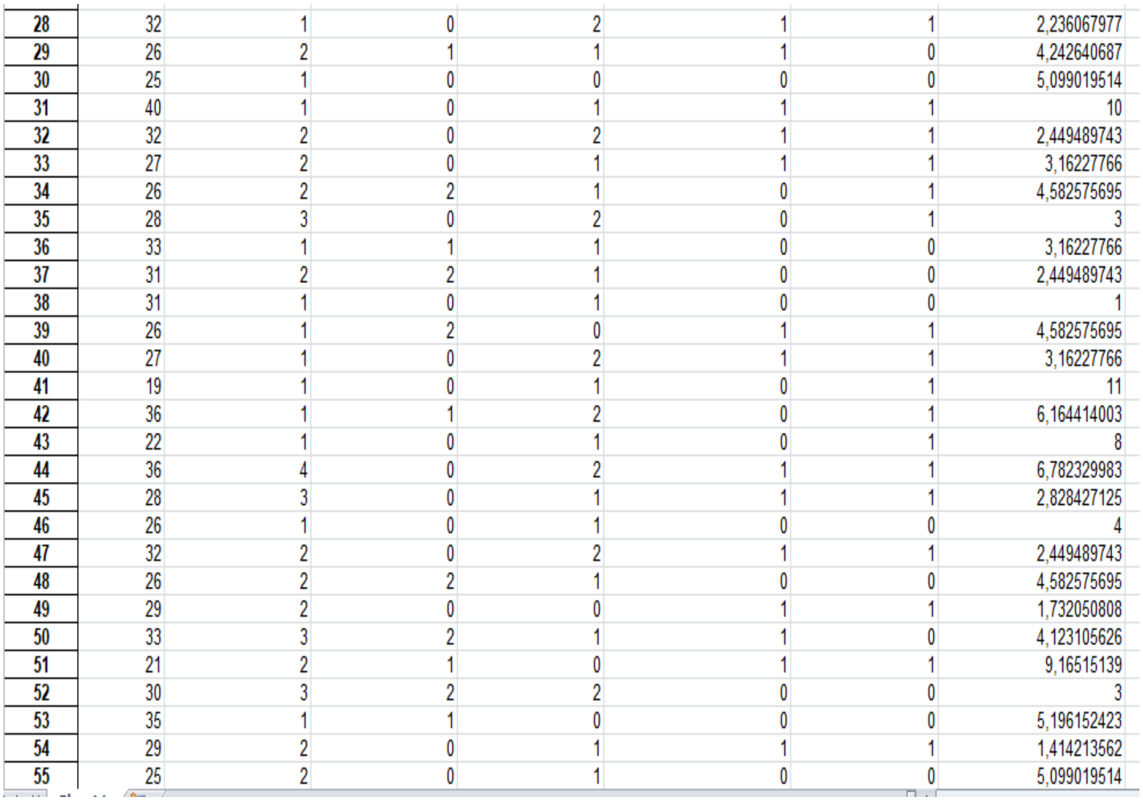
	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian	dis
27	30	1	0	1	0	0	0.000000
38	31	1	0	1	0	0	1.000000
67	29	2	0	1	1	0	1.414214
54	29	2	0	1	1	1	1.414214
59	30	2	1	2	1	1	1.732051

```
 In [8]: z = y["Caesarian"]
         z

Out[8]: 27    0
        38    0
        67    0
        54    1
        59    1
        Name: Caesarian, dtype: int64

 In [9]: np.mean(z)
Out[9]: 0.4

In [11]: df.to_excel('C:/yolanda/datamining/outputNo4.xls')

 In [ ]:
```

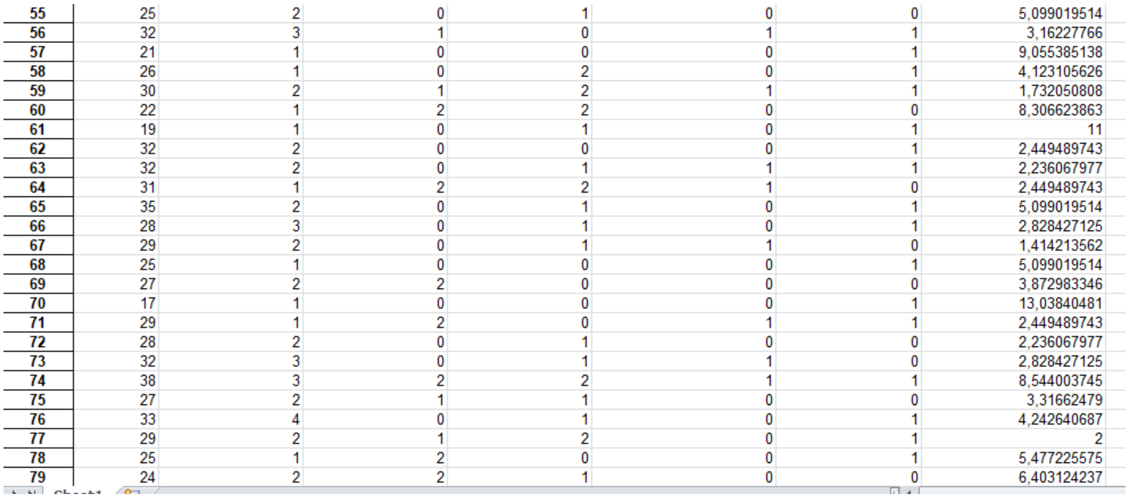
- Data Hasil Output No 4



A	B	C	D	E	F	G	H
	Usia	Kelahiran_ke-	Waktu_Kelahiran	Tekanan_darah	Kelainan_jantung	Caesarian	dis
0	22	1	0	2	0	0	8,062257748
1	26	2	0	1	0	1	4,123105626
2	26	2	1	1	0	0	4,242640687
3	28	1	0	2	0	0	2,236067977
4	22	2	0	1	0	1	8,062257748
5	26	1	1	0	0	0	4,242640687
6	27	2	0	1	0	0	3,16227766
7	32	3	0	1	0	1	2,828427125
8	28	2	0	1	0	0	2,236067977
9	27	1	1	1	0	1	3,16227766
10	36	1	0	1	0	0	6
11	33	1	1	0	0	1	3,31662479
12	23	1	1	1	0	0	7,071067812
13	20	1	0	1	1	0	10
14	29	1	2	0	1	1	2,449489743
15	25	1	2	0	0	0	5,477225575
16	25	1	0	1	0	0	5
17	20	1	2	2	0	1	10,24695077
18	37	3	0	1	1	1	7,280109889
19	24	1	2	0	1	1	6,403124237
20	26	1	1	1	0	0	4,123105626
21	33	2	0	0	1	1	3,31662479
22	25	1	1	2	0	0	5,196152423
23	27	1	0	0	1	1	3,16227766
24	20	1	0	2	1	1	10,04987562
25	18	1	0	1	0	0	12
26	18	1	1	2	1	1	12,08304597
27	30	1	0	1	0	0	0
28	32	1	0	2	1	1	2,236067977



28	32	1	0	2	1	1	2,236067977
29	26	2	1	1	1	0	4,242640687
30	25	1	0	0	0	0	5,099019514
31	40	1	0	1	1	1	10
32	32	2	0	2	1	1	2,449489743
33	27	2	0	1	1	1	3,16227766
34	26	2	2	1	0	1	4,582575695
35	28	3	0	2	0	1	3
36	33	1	1	1	0	0	3,16227766
37	31	2	2	1	0	0	2,449489743
38	31	1	0	1	0	0	1
39	26	1	2	0	1	1	4,582575695
40	27	1	0	2	1	1	3,16227766
41	19	1	0	1	0	1	11
42	36	1	1	2	0	1	6,164414003
43	22	1	0	1	0	1	8
44	36	4	0	2	1	1	6,782329983
45	28	3	0	1	1	1	2,828427125
46	26	1	0	1	0	0	4
47	32	2	0	2	1	1	2,449489743
48	26	2	2	1	0	0	4,582575695
49	29	2	0	0	1	1	1,732050808
50	33	3	2	1	1	0	4,123105626
51	21	2	1	0	1	1	9,16515139
52	30	3	2	2	0	0	3
53	35	1	1	0	0	0	5,196152423
54	29	2	0	1	1	1	1,414213562
55	25	2	0	1	0	0	5,099019514



55	25	2	0	1	0	0	5,099019514
56	32	3	1	0	1	1	3,16227766
57	21	1	0	0	0	1	9,055385138
58	26	1	0	2	0	1	4,123105626
59	30	2	1	2	1	1	1,732050808
60	22	1	2	2	0	0	8,306623863
61	19	1	0	1	0	1	11
62	32	2	0	0	0	1	2,449489743
63	32	2	0	1	1	1	2,236067977
64	31	1	2	2	1	0	2,449489743
65	35	2	0	1	0	1	5,099019514
66	28	3	0	1	0	1	2,828427125
67	29	2	0	1	1	0	1,414213562
68	25	1	0	0	0	1	5,099019514
69	27	2	2	0	0	0	3,872983346
70	17	1	0	0	0	1	13,03840481
71	29	1	2	0	1	1	2,449489743
72	28	2	0	1	0	0	2,236067977
73	32	3	0	1	1	0	2,828427125
74	38	3	2	2	1	1	8,544003745
75	27	2	1	1	0	0	3,31662479
76	33	4	0	1	0	1	4,242640687
77	29	2	1	2	0	1	2
78	25	1	2	0	0	1	5,477225575
79	24	2	2	1	0	0	6,403124237