Lab 7 Association Rule Learning (10/6/2022)

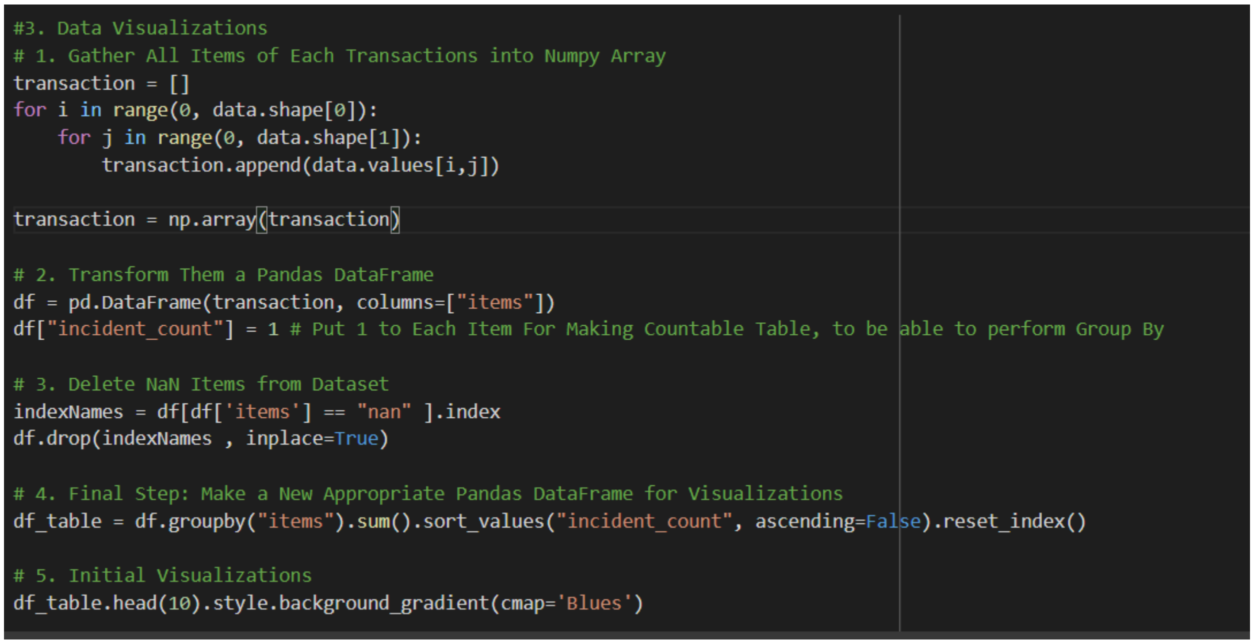
ລະຫັດນັກສຶກສາ: 205Q0010.19
ຊື່ ແລະ ນາມສະກຸນ:.ທ ນູຊື່ວ ເຮີ 3CW1

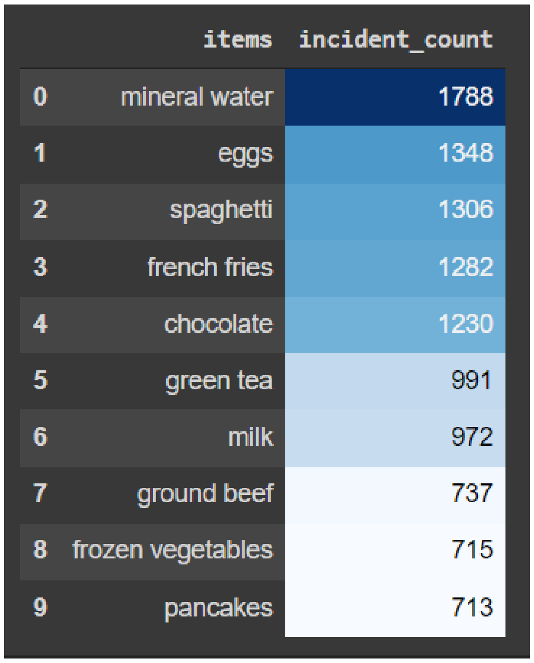
ຈົ່ງນຳໃຊ້ຄຳສັ່ງຂອງ Python ດ້ວຍ Association Rule Learning.ipynb ເພື່ອຕອບຄຳຖາມຕໍ່ໄປນີ້ໃຫ້ສຳເລັດ:

1.ຈາກຊຸດຂໍ້ມູນ Market_Basket_Optimisation.csv. ຈົ່ງບອກຂະໜາດຂອງຊຸດຂໍ້ມູນວ່າມີຈຳນວນແຖວ ແລະ ຖັນເທົ່າກັບເທົ່າໃດ?

Data.Shape	
Lows	7501
Columns	20

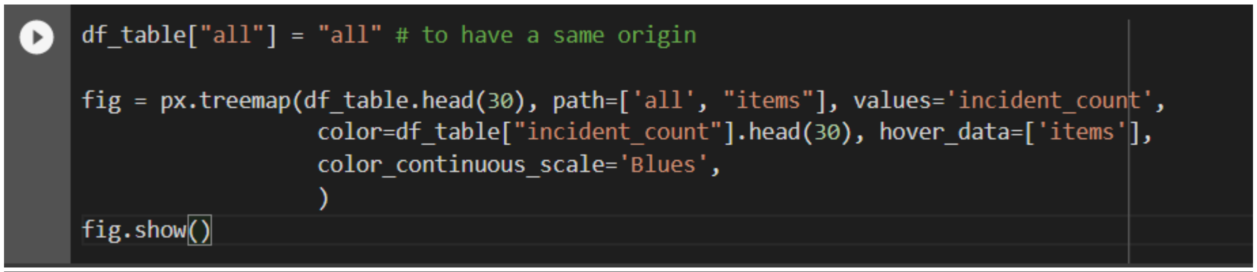
2. ຈົ່ງທຳການສຳຫຼວດຊຸດຂໍ້ມູນດ້ວຍການສະແດງຄ່າຄວາມຖີ່ຂອງ 10 ຜະລິດຕະພັນແຕ່ໃຫຍ່ຫານ້ອຍ.


```
#3. Data Visualizations
# 1. Gather All Items of Each Transactions into Numpy Array
transaction = []
for i in range(0, data.shape[0]):
    for j in range(0, data.shape[1]):
        transaction.append(data.values[i,j])

transaction = np.array(transaction)

# 2. Transform Them a Pandas DataFrame
df = pd.DataFrame(transaction, columns=["items"])
df["incident_count"] = 1 # Put 1 to Each Item For Making Countable Table, to be able to perform Group By

# 3. Delete NaN Items from Dataset
indexNames = df[df['items'] == "nan" ].index
df.drop(indexNames , inplace=True)

# 4. Final Step: Make a New Appropriate Pandas DataFrame for Visualizations
df_table = df.groupby("items").sum().sort_values("incident_count", ascending=False).reset_index()

# 5. Initial Visualizations
df_table.head(10).style.background_gradient(cmap='Blues')
```

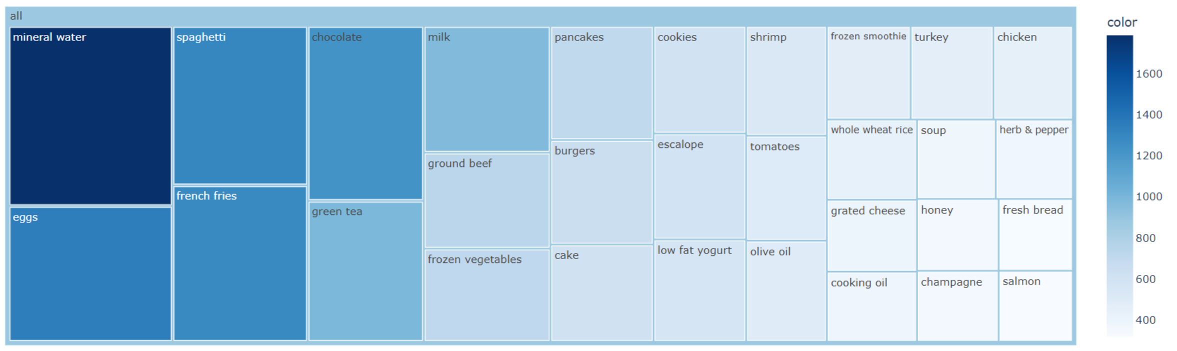


	items	incident_count
0	mineral water	1788
1	eggs	1348
2	spaghetti	1306
3	french fries	1282
4	chocolate	1230
5	green tea	991
6	milk	972
7	ground beef	737
8	frozen vegetables	715
9	pancakes	713

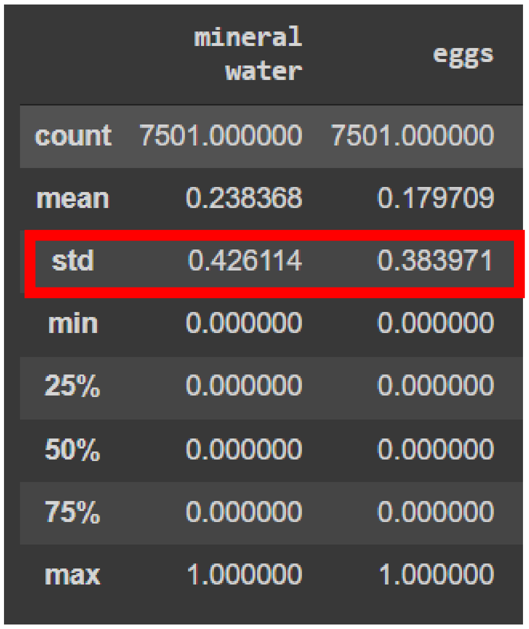
3.ຈົ່ງທຳການສຳຫຼວດຊຸດຂໍ້ມູນດ້ວຍ treemap ແລະ ສະແດງຄ່າຄວາມຖີ່ຂອງ 30 ຜະລິດຕະພັນແຕ່ໃຫຍ່ຫານ້ອຍ



```
df_table["all"] = "all" # to have a same origin

fig = px.treemap(df_table.head(30), path=['all', "items"], values='incident_count',
                  color=df_table["incident_count"].head(30), hover_data=['items'],
                  color_continuous_scale='Blues',
                  )
fig.show()
```



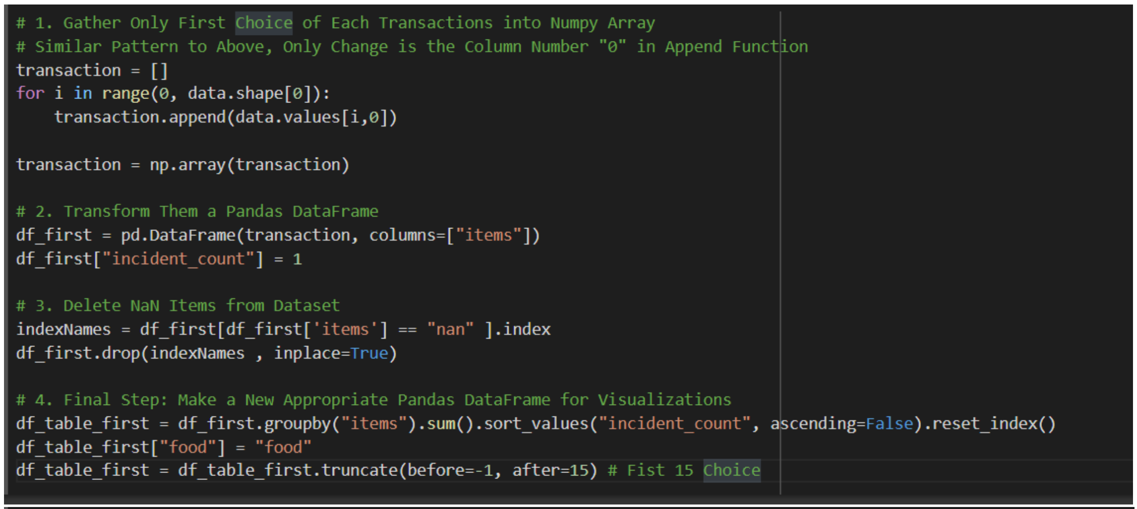
4. ຈົ່ງທຳການສຳຫຼວດຊຸດຂໍ້ມູນດ້ວຍ df_top20_multiple_record_check.describe(). ຖາມວ່າລະຫວ່າງ ຜະລິດຕະພັນ mineral water ແລະ eggs ຜະລິດຕະພັນໃດມີຄ່າຜັນປ່ຽນມາດຕະຖານນ້ອຍກ່ວາກັນ?



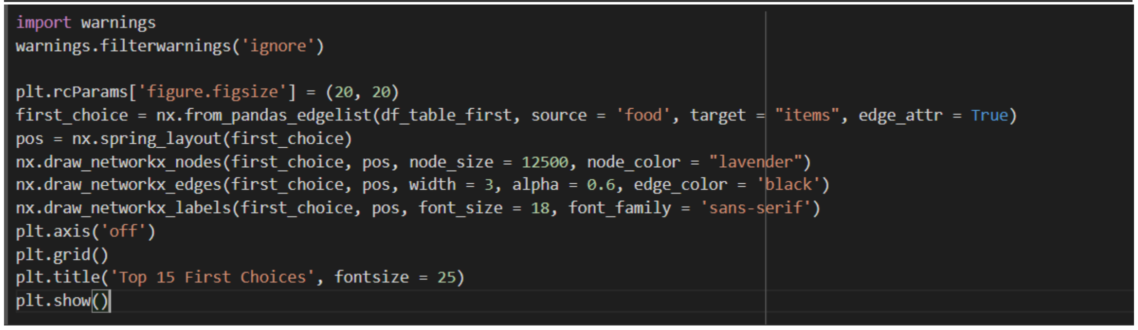
	mineral water	eggs
count	7501.000000	7501.000000
mean	0.238368	0.179709
std	0.426114	0.383971
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	1.000000

Eggs ມີຄ່າຜັນປ່ຽນມາດຕະຖານນ້ອຍກ່ວາ

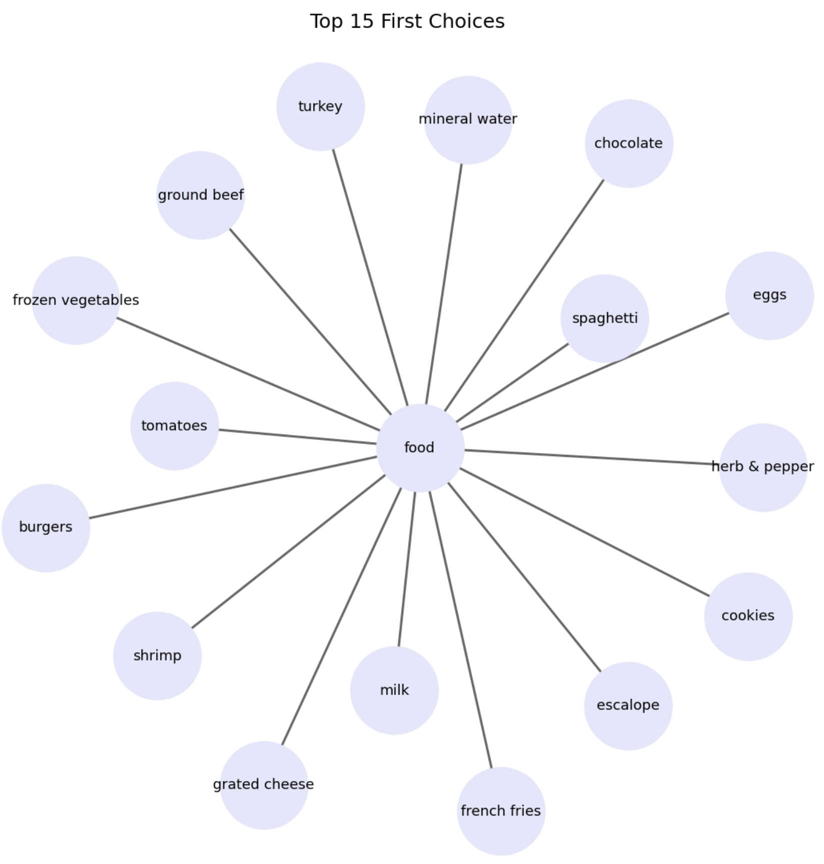
5. ຈົ່ງທຳການສຳຫຼວດຊຸດຂໍ້ມູນດ້ວຍການສະແດງ 15 ຜະລິດຕະພັນໃນການເລືອກຄັ້ງທຳອິດ (Top 15 First Choices) ແລະ 15 ຜະລິດຕະພັນໃນການເລືອກຄັ້ງທີ 2 (Top 15 Second Choices) ຂອງລູກຄ້າ.

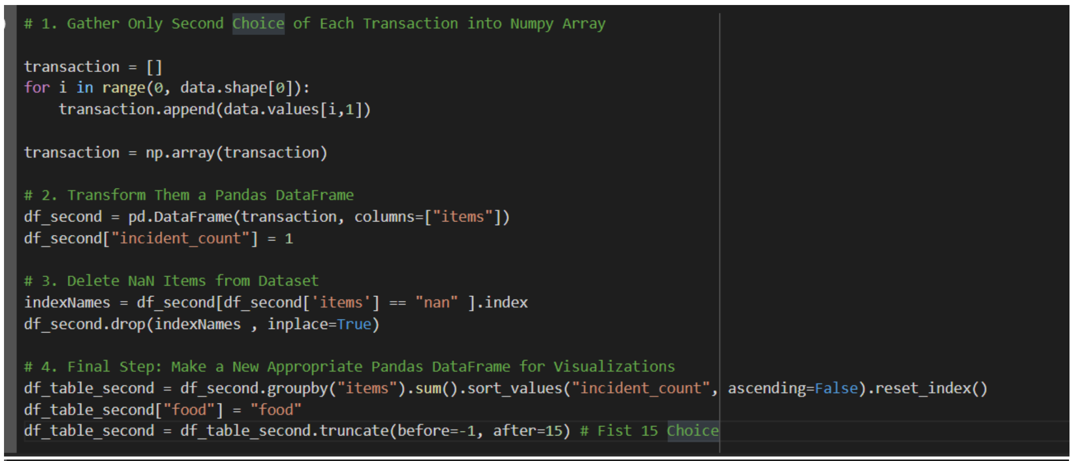


```python
# 1. Gather Only First Choice of Each Transactions into Numpy Array
# Similar Pattern to Above, Only Change is the Column Number "0" in Append Function
transaction = []
for i in range(0, data.shape[0]):
    transaction.append(data.values[i,0])

transaction = np.array(transaction)

# 2. Transform Them a Pandas DataFrame
df_first = pd.DataFrame(transaction, columns=["items"])
df_first["incident_count"] = 1

# 3. Delete NaN Items from Dataset
indexNames = df_first[df_first['items'] == "nan" ].index
df_first.drop(indexNames , inplace=True)

# 4. Final Step: Make a New Appropriate Pandas DataFrame for Visualizations
df_table_first = df_first.groupby("items").sum().sort_values("incident_count", ascending=False).reset_index()
df_table_first["food"] = "food"
df_table_first = df_table_first.truncate(before=-1, after=15) # Fist 15 Choice
```

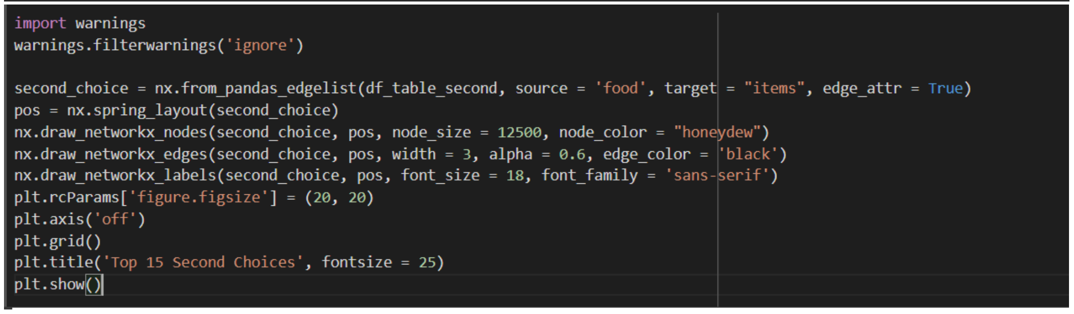


```python
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (20, 20)
first_choice = nx.from_pandas_edgelist(df_table_first, source = 'food', target = "items", edge_attr = True)
pos = nx.spring_layout(first_choice)
nx.draw_networkx_nodes(first_choice, pos, node_size = 12500, node_color = "lavender")
nx.draw_networkx_edges(first_choice, pos, width = 3, alpha = 0.6, edge_color = 'black')
nx.draw_networkx_labels(first_choice, pos, font_size = 18, font_family = 'sans-serif')
plt.axis('off')
plt.grid()
plt.title('Top 15 First Choices', fontsize = 25)
plt.show()
```

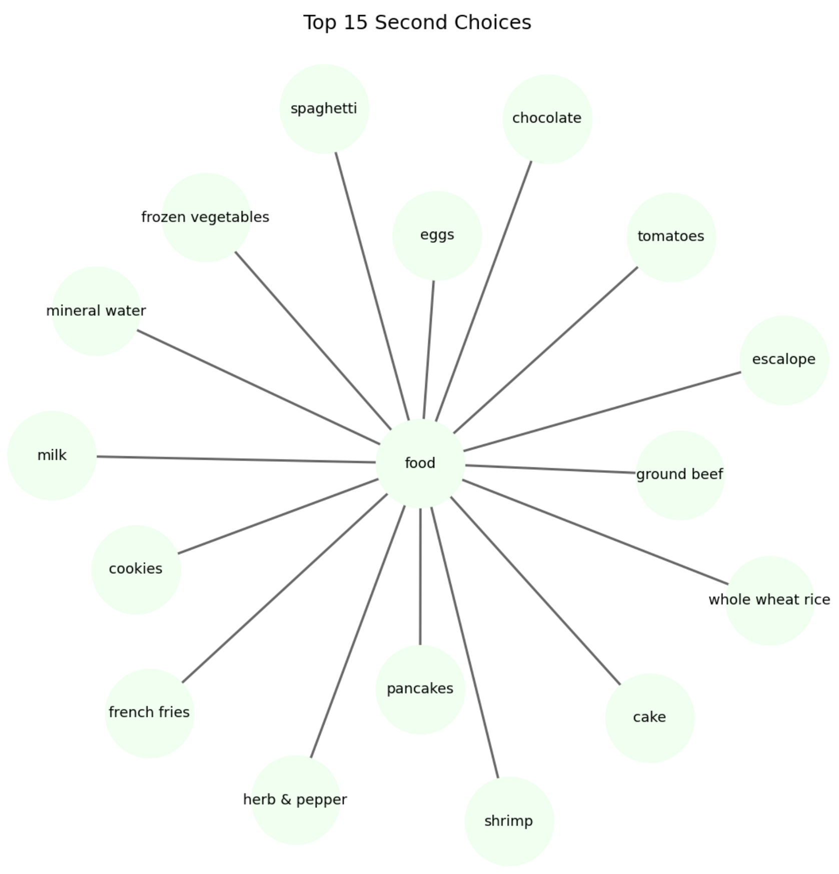




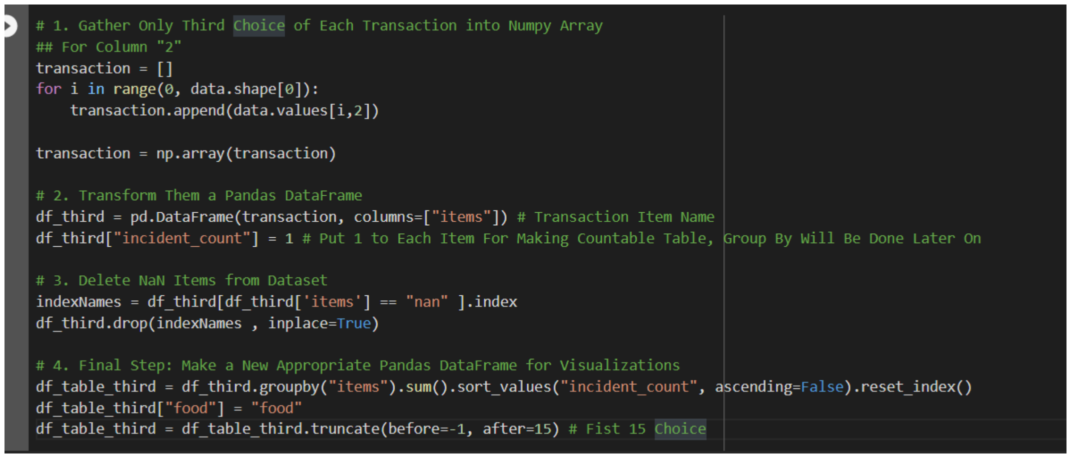
```
# 1. Gather Only Second Choice of Each Transaction into Numpy Array

transaction = []
for i in range(0, data.shape[0]):
    transaction.append(data.values[i,1])

transaction = np.array(transaction)

# 2. Transform Them a Pandas DataFrame
df_second = pd.DataFrame(transaction, columns=["items"])
df_second["incident_count"] = 1

# 3. Delete NaN Items from Dataset
indexNames = df_second[df_second['items'] == "nan" ].index
df_second.drop(indexNames , inplace=True)

# 4. Final Step: Make a New Appropriate Pandas DataFrame for Visualizations
df_table_second = df_second.groupby("items").sum().sort_values("incident_count", ascending=False).reset_index()
df_table_second["food"] = "food"
df_table_second = df_table_second.truncate(before=-1, after=15) # Fist 15 Choice
```



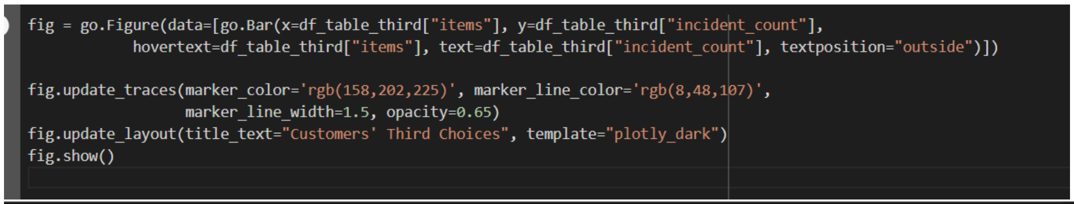
```
import warnings
warnings.filterwarnings('ignore')

second_choice = nx.from_pandas_edgelist(df_table_second, source = 'food', target = "items", edge_attr = True)
pos = nx.spring_layout(second_choice)
nx.draw_networkx_nodes(second_choice, pos, node_size = 12500, node_color = "honeydew")
nx.draw_networkx_edges(second_choice, pos, width = 3, alpha = 0.6, edge_color = 'black')
nx.draw_networkx_labels(second_choice, pos, font_size = 18, font_family = 'sans-serif')
plt.rcParams['figure.figsize'] = (20, 20)
plt.axis('off')
plt.grid()
plt.title('Top 15 Second Choices', fontsize = 25)
plt.show()
```



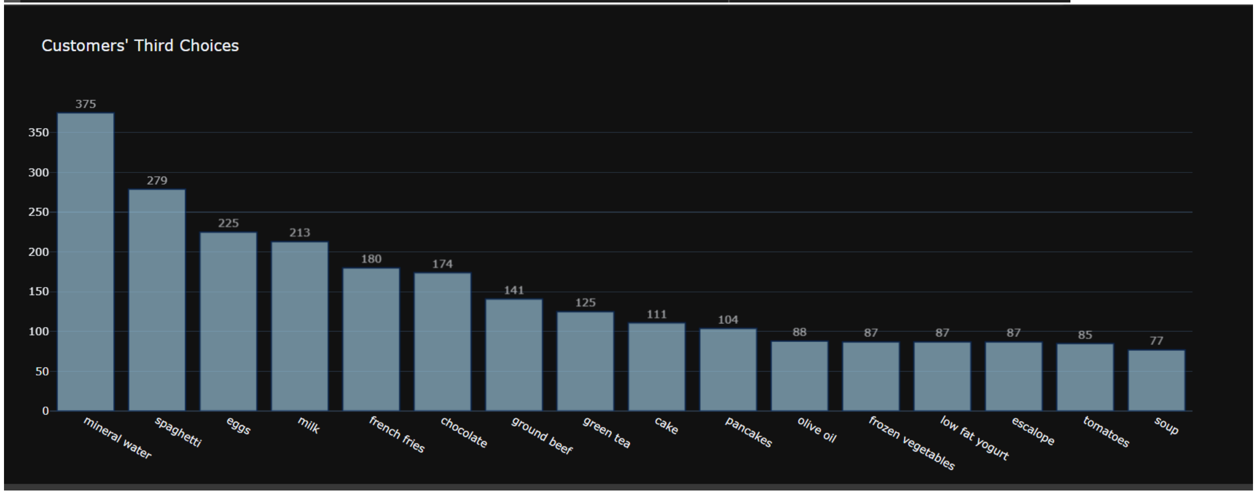
6. ຈົ່ງທຳການສຳຫຼວດຊຸດຂໍ້ມູນດ້ວຍການສະແດງຄ່າຄວາມຖີ່ຂອງຜະລິດຕະພັນໃນການເລືອກຄັ້ງທີ 3 ຂອງລູກຄ້າດ້ວຍ (Customers' Third Choices).



```
# 1. Gather Only Third Choice of Each Transaction into Numpy Array
## For Column "2"
transaction = []
for i in range(0, data.shape[0]):
    transaction.append(data.values[i,2])

transaction = np.array(transaction)

# 2. Transform Them a Pandas DataFrame
df_third = pd.DataFrame(transaction, columns=["items"]) # Transaction Item Name
df_third["incident_count"] = 1 # Put 1 to Each Item For Making Countable Table, Group By Will Be Done Later On

# 3. Delete NaN Items from Dataset
indexNames = df_third[df_third['items'] == "nan" ].index
df_third.drop(indexNames , inplace=True)

# 4. Final Step: Make a New Appropriate Pandas DataFrame for Visualizations
df_table_third = df_third.groupby("items").sum().sort_values("incident_count", ascending=False).reset_index()
df_table_third["food"] = "food"
df_table_third = df_table_third.truncate(before=-1, after=15) # Fist 15 Choice
```



```
fig = go.Figure(data=[go.Bar(x=df_table_third["items"], y=df_table_third["incident_count"],
            hovertext=df_table_third["items"], text=df_table_third["incident_count"], textposition="outside")])

fig.update_traces(marker_color='rgb(158,202,225)', marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5, opacity=0.65)
fig.update_layout(title_text="Customers' Third Choices", template="plotly_dark")
fig.show()
```



7. ຖາມວ່າກ່ອນການກະກຽມຂໍ້ມູນ (data preprocessing) ຊຸດຂໍ້ມູນມີ memory usage ເທົ່າໃດ?

memory usage: 1.1 MB

8. ຖາມວ່າຫຼັງການກະກຽມຂໍ້ມູນ (data preprocessing) ຊຸດຂໍ້ມູນມີ memory usage ເທົ່າໃດ?

memory usage: 886.5 KB

9. ຖາມວ່າຫຼັງການຄັດເລືອກເອົາ 50 ຜະລິດຕະພັນໃນຊຸດຂໍ້ມູນມາດຳເນີນການປະມວນຜົນ, memory usage ເທົ່າໃດ (ຄ່າຂໍ້ມູນເປັນ boolean)?

memory usage: 366.4 KB

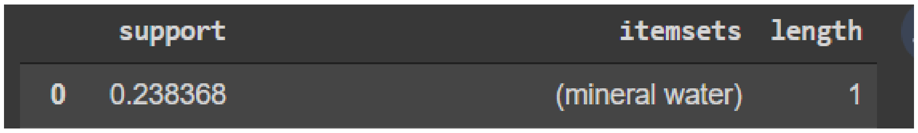
10. ຖາມວ່າຫຼັງການຄັດເລືອກເອົາ 50 ຜະລິດຕະພັນໃນຊຸດຂໍ້ມູນມາດຳເນີນການປະມວນຜົນ, memory usage ເທົ່າໃດ (ຄ່າຂໍ້ມູນເປັນ int64)?

memory usage: 2.9 MB

11. ໃນການວິເຄາະຄວາມສຳພັນຂອງຂໍ້ມູນດ້ວຍ Association Rule Learning ຄັ້ງນີ້. ຖາມວ່າເພິ່ນນຳໃຊ້ Algorithm ຕົວໃດ?

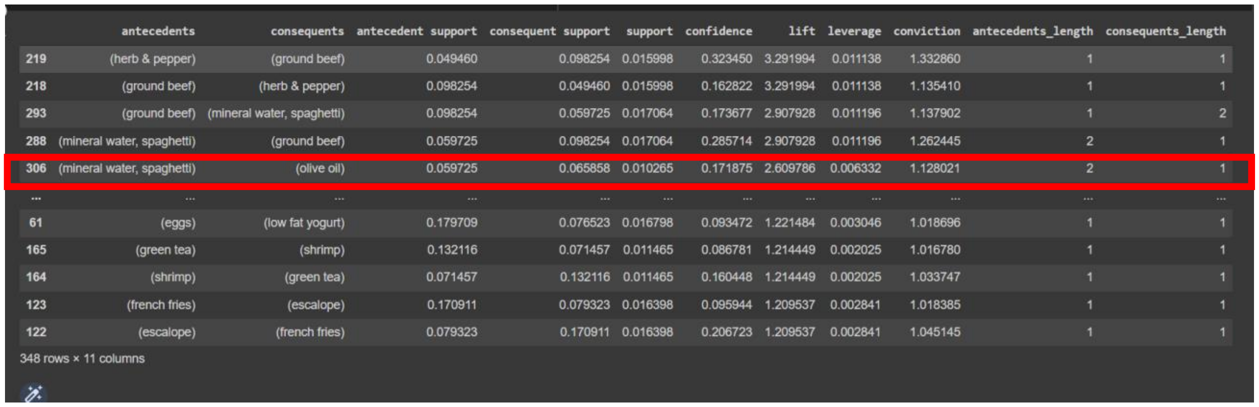
ຕອບ: Apriori Algorithm

12. ຖາມວ່າຜະລິດຕະພັນ mineral water ມີຄ່າ support ເທົ່າໃດ?



	support	itemsets	length
0	0.238368	(mineral water)	1

13. ເມື່ອ antecedents ເປັນ (mineral water, spaghetti) ເວລານັ້ນ consequents ເປັນ olive oil. ຖາມ ວ່າ confidence ແລະ lift ມີຄ່າເທົ່າໃດ?

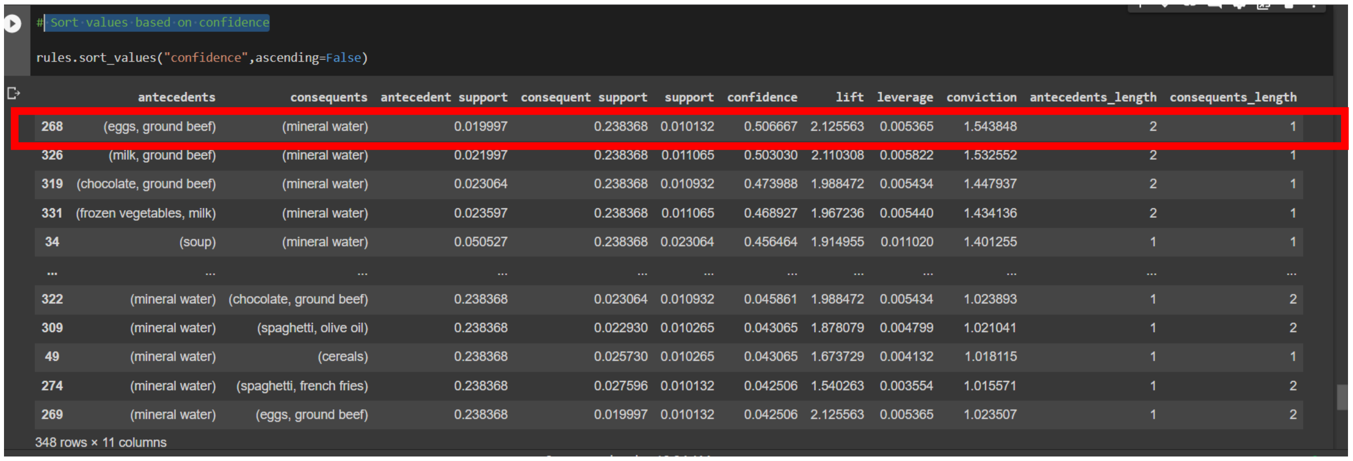


	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents_length	consequents_length
219	(herb & pepper)	(ground beef)	0.049460	0.098254	0.015998	0.323450	3.291994	0.011138	1.332860	1	1
218	(ground beef)	(herb & pepper)	0.098254	0.049460	0.015998	0.162822	3.291994	0.011138	1.135410	1	1
293	(ground beef)	(mineral water, spaghetti)	0.098254	0.059725	0.017064	0.173677	2.907928	0.011196	1.137902	1	2
288	(mineral water, spaghetti)	(ground beef)	0.059725	0.098254	0.017064	0.285714	2.907928	0.011196	1.262445	2	1
306	(mineral water, spaghetti)	(olive oil)	0.059725	0.065858	0.010265	0.171875	2.609786	0.006332	1.128021	2	1
...	...	...	...	...	...	...	...	...	...	...	...
61	(eggs)	(low fat yogurt)	0.179709	0.076523	0.016798	0.093472	1.221484	0.003046	1.018696	1	1
165	(green tea)	(shrimp)	0.132116	0.071457	0.011465	0.086781	1.214449	0.002025	1.016780	1	1
164	(shrimp)	(green tea)	0.071457	0.132116	0.011465	0.160448	1.214449	0.002025	1.033747	1	1
123	(french fries)	(escalope)	0.170911	0.079323	0.016398	0.095944	1.209537	0.002841	1.018385	1	1
122	(escalope)	(french fries)	0.079323	0.170911	0.016398	0.206723	1.209537	0.002841	1.045145	1	1

348 rows × 11 columns

- Confidence 0.171875
- Lift 2.609786

14. ຖາມວ່າຄວາມສຳພັນຂອງກຸ່ມຜະລິດຕະພັນໃດມີຄ່າ confidence ສູງກ່ວາໝູ່?



```python
# Sort values based on confidence
rules.sort_values("confidence",ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	antecedents_length	consequents_length
268	(eggs, ground beef)	(mineral water)	0.019997	0.238368	0.010132	0.506667	2.125563	0.005365	1.543848	2	1
326	(milk, ground beef)	(mineral water)	0.021997	0.238368	0.011065	0.503030	2.110308	0.005822	1.532552	2	1
319	(chocolate, ground beef)	(mineral water)	0.023064	0.238368	0.010932	0.473988	1.988472	0.005434	1.447937	2	1
331	(frozen vegetables, milk)	(mineral water)	0.023597	0.238368	0.011065	0.468927	1.967236	0.005440	1.434136	2	1
34	(soup)	(mineral water)	0.050527	0.238368	0.023064	0.456464	1.914955	0.011020	1.401255	1	1
...	...	...	...	...	...	...	...	...	...	...	...
322	(mineral water)	(chocolate, ground beef)	0.238368	0.023064	0.010932	0.045861	1.988472	0.005434	1.023893	1	2
309	(mineral water)	(spaghetti, olive oil)	0.238368	0.022930	0.010265	0.043065	1.878079	0.004799	1.021041	1	2
49	(mineral water)	(cereals)	0.238368	0.025730	0.010265	0.043065	1.673729	0.004132	1.018115	1	1
274	(mineral water)	(spaghetti, french fries)	0.238368	0.027596	0.010132	0.042506	1.540263	0.003554	1.015571	1	2
269	(mineral water)	(eggs, ground beef)	0.238368	0.019997	0.010132	0.042506	2.125563	0.005365	1.023507	1	2

348 rows × 11 columns