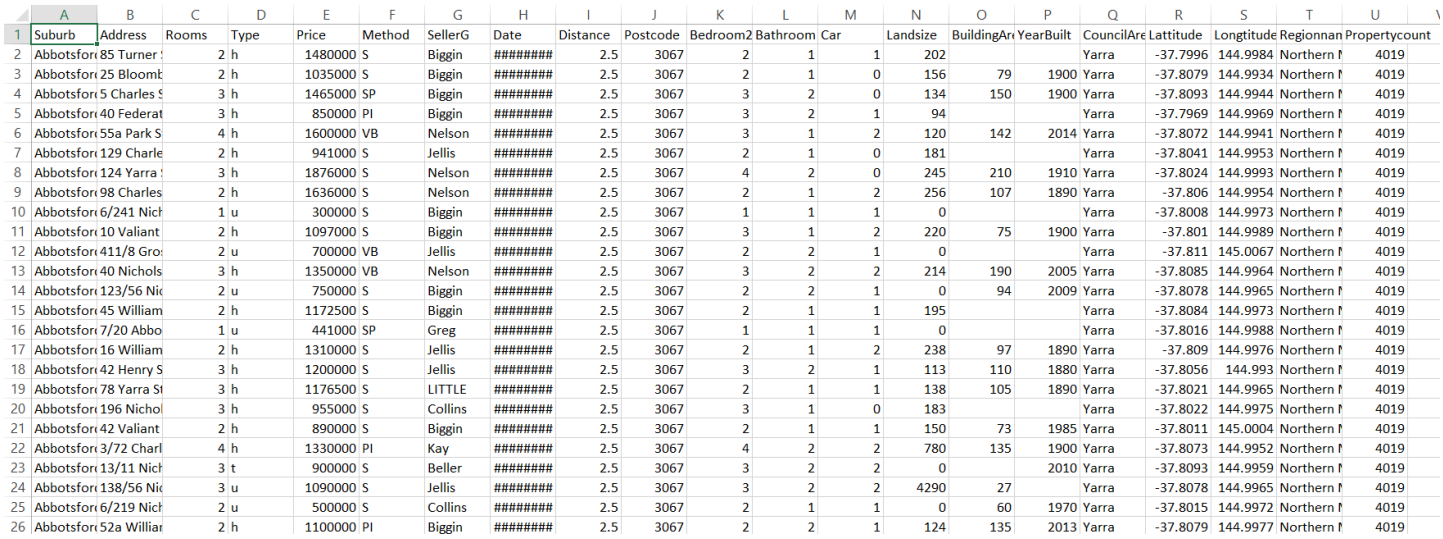
Name: **Varun Sudhir**

Reg No: **21BDS0040**

Data Mining Lab Digital Assignment – 2

Q1) Consider the dataset from the below link



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingAr	YearBuilt	CouncilAre	Lattitude	Longtitude	Regionnan	Propertycount
2	Abbotsfor	85 Turner	2	h	1480000	S	Biggin	########	2.5	3067	2	1	1	202			Yarra	-37.7996	144.9984	Northern	4019
3	Abbotsfor	25 Bloomb	2	h	1035000	S	Biggin	########	2.5	3067	2	1	0	156	79	1900	Yarra	-37.8079	144.9934	Northern	4019
4	Abbotsfor	5 Charles	3	h	1465000	SP	Biggin	########	2.5	3067	3	2	0	134	150	1900	Yarra	-37.8093	144.9944	Northern	4019
5	Abbotsfor	40 Federat	3	h	850000	PI	Biggin	########	2.5	3067	3	2	1	94			Yarra	-37.7969	144.9969	Northern	4019
6	Abbotsfor	55a Park S	4	h	1600000	VB	Nelson	########	2.5	3067	3	1	2	120	142	2014	Yarra	-37.8072	144.9941	Northern	4019
7	Abbotsfor	129 Charle	2	h	941000	S	Jellis	########	2.5	3067	2	1	0	181			Yarra	-37.8041	144.9953	Northern	4019
8	Abbotsfor	124 Yarra	3	h	1876000	S	Nelson	########	2.5	3067	4	2	0	245	210	1910	Yarra	-37.8024	144.9993	Northern	4019
9	Abbotsfor	98 Charles	2	h	1636000	S	Nelson	########	2.5	3067	2	1	2	256	107	1890	Yarra	-37.806	144.9954	Northern	4019
10	Abbotsfor	6/241 Nich	1	u	300000	S	Biggin	########	2.5	3067	1	1	1	0			Yarra	-37.8008	144.9973	Northern	4019
11	Abbotsfor	10 Valiant	2	h	1097000	S	Biggin	########	2.5	3067	3	1	2	220	75	1900	Yarra	-37.801	144.9989	Northern	4019
12	Abbotsfor	411/8 Gro	2	u	700000	VB	Jellis	########	2.5	3067	2	2	1	0			Yarra	-37.811	145.0067	Northern	4019
13	Abbotsfor	40 Nichols	3	h	1350000	VB	Nelson	########	2.5	3067	3	2	2	214	190	2005	Yarra	-37.8085	144.9964	Northern	4019
14	Abbotsfor	123/56 Nic	2	u	750000	S	Biggin	########	2.5	3067	2	2	1	0	94	2009	Yarra	-37.8078	144.9965	Northern	4019
15	Abbotsfor	45 William	2	h	1172500	S	Biggin	########	2.5	3067	2	1	1	195			Yarra	-37.8084	144.9973	Northern	4019
16	Abbotsfor	7/20 Abbo	1	u	441000	SP	Greg	########	2.5	3067	1	1	1	0			Yarra	-37.8016	144.9988	Northern	4019
17	Abbotsfor	16 William	2	h	1310000	S	Jellis	########	2.5	3067	2	1	2	238	97	1890	Yarra	-37.809	144.9976	Northern	4019
18	Abbotsfor	42 Henry S	3	h	1200000	S	Jellis	########	2.5	3067	3	2	1	113	110	1880	Yarra	-37.8056	144.993	Northern	4019
19	Abbotsfor	78 Yarra St	3	h	1176500	S	LITTLE	########	2.5	3067	2	1	1	138	105	1890	Yarra	-37.8021	144.9965	Northern	4019
20	Abbotsfor	196 Nichol	3	h	955000	S	Collins	########	2.5	3067	3	1	0	183			Yarra	-37.8022	144.9975	Northern	4019
21	Abbotsfor	42 Valiant	2	h	890000	S	Biggin	########	2.5	3067	2	1	1	150	73	1985	Yarra	-37.8011	145.0004	Northern	4019
22	Abbotsfor	3/72 Charl	4	h	1330000	PI	Kay	########	2.5	3067	4	2	2	780	135	1900	Yarra	-37.8073	144.9952	Northern	4019
23	Abbotsfor	13/11 Nich	3	t	900000	S	Beller	########	2.5	3067	3	2	2	0		2010	Yarra	-37.8093	144.9959	Northern	4019
24	Abbotsfor	138/56 Nic	3	u	1090000	S	Jellis	########	2.5	3067	3	2	2	4290	27		Yarra	-37.8078	144.9965	Northern	4019
25	Abbotsfor	6/219 Nich	2	u	500000	S	Collins	########	2.5	3067	2	1	1	0	60	1970	Yarra	-37.8015	144.9972	Northern	4019
26	Abbotsfor	52a Williar	2	h	1100000	PI	Biggin	########	2.5	3067	2	2	1	124	135	2013	Yarra	-37.8079	144.9977	Northern	4019

a. Find out how many data is missing in each attribute

b. For all the missing values in ‘car’ attribute, fill the missing value with the mode.

c. For the ‘BuildingArea’ attribute, fill the missing value with the linear interpolation and quadratic interpolation.

d. Fill the ‘yearbuilt’ attribute with forward fill approach

e. Remove all the rows which doesn’t have a councilarea data.

Code:

```
In [8]: # Varun Sudhir
        # Data Mining Lab Digital Assignment -2

        import numpy as np
        import pandas as pd
```

```
In [9]: df = pd.read_csv("missing data.csv")
```

```
# Question 1
# Varun Sudhir 21BDS0040

#A: Find out how many data is missing in each attribute
missing_data = df.isnull().sum()
print("Missing Data Count:\n", missing_data)


#B: For all the missing values in 'car' attribute, fill the missing value with the mode.
missing_values_car = df['Car'].isnull().sum()
print("\nThe number of missing vales in 'Car' attribute is:",missing_values_car)
mode_car = df['Car'].mode()[0]
df['Car'].fillna(mode_car, inplace=True)
missing_values_car = df['Car'].isnull().sum()
print("The number of missing values in 'Car' attribute after replacing with mode is:",missing_values_car)
print("We have removed all the missing values in the column")
print()
print()
```

```
#C: Fill the missing values in the 'BuildingArea' attribute with linear and quadratic interpolation
df['BuildingArea_linear'] = df['BuildingArea'].interpolate(method='linear')
df['BuildingArea_quadratic'] = df['BuildingArea'].interpolate(method='quadratic')
print(df['BuildingArea_linear'])
print(df['BuildingArea_quadratic'])

#D: Fill the missing values in the 'YearBuilt' attribute with forward fill approach
df['YearBuilt'].fillna(method='ffill', inplace=True)
print(df['YearBuilt'])

#E
df.dropna(subset=['CouncilArea'],inplace=True)
df
```

Output:

A)

```
Missing Data Count:
 Suburb              0
Address              0
Rooms                0
Type                 0
Price                0
Method               0
SellerG              0
Date                 0
Distance             0
Postcode             0
Bedroom2             0
Bathroom             0
Car                 62
Landsize             0
BuildingArea      6450
YearBuilt         5375
CouncilArea       1369
Lattitude            0
Longtitude           0
Regionname           0
Propertycount        0
dtype: int64
```

B)

```
The number of missing vales in 'Car' attribute is: 62
The number of missing values in 'Car' attribute after replacing with mode is: 0
We have removed all the missing values in the column
```

C)

```
0          NaN
1         79.0
2        150.0
3        146.0
4        142.0
         ...
13575    146.0
13576    133.0
13577    145.0
13578    157.0
13579    112.0
Name: BuildingArea_linear, Length: 13580, dtype: float64
0               NaN
1         79.000000
2        150.000000
3        148.715056
4        142.000000
            ...
13575    159.855594
13576    133.000000
13577    148.024068
13578    157.000000
13579    112.000000
Name: BuildingArea_quadratic, Length: 13580, dtype: float64
```

D)

```
Name: YearBuilt, Length: 13580, dtype: float64
0        Yarra
1        Yarra
2        Yarra
3        Yarra
4        Yarra
         ...
13575      NaN
13576      NaN
13577      NaN
13578      NaN
13579      NaN
```

E)

Out[31]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Landsize	BuildingArea	YearBuilt	CouncilArea	L
0	Abbotsford	85 Turner St	2	h	1480000	S	Biggin	03-12-2016	2.5	3067	...	202	NaN	NaN	Yarra	-3
1	Abbotsford	25 Bloomburg St	2	h	1035000	S	Biggin	04-02-2016	2.5	3067	...	156	79.0	1900.0	Yarra	-3
2	Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	04-03-2017	2.5	3067	...	134	150.0	1900.0	Yarra	-3
3	Abbotsford	40 Federation La	3	h	850000	PI	Biggin	04-03-2017	2.5	3067	...	94	NaN	1900.0	Yarra	-3
4	Abbotsford	55a Park St	4	h	1600000	VB	Nelson	04-06-2016	2.5	3067	...	120	142.0	2014.0	Yarra	-3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
12208	Williamstown	87 Pasco St	3	h	1285000	S	Jas	29-07-2017	6.8	3016	...	296	NaN	1967.0	Hobsons Bay	-3
12209	Windsor	201/152 Peel St	2	u	560000	PI	hockingstuart	29-07-2017	4.6	3181	...	0	61.6	2012.0	Stonnington	-3
12210	Wollert	60 Saltlake Bvd	3	h	525300	S	Stockdale	29-07-2017	25.5	3750	...	400	NaN	2012.0	Whittlesea	-3
12211	Yarraville	2 Adeney St	2	h	750000	SP	hockingstuart	29-07-2017	6.3	3013	...	269	NaN	2012.0	Maribyrnong	-3
12212	Yarraville	54 Pentland Pde	6	h	2450000	VB	Village	29-07-2017	6.3	3013	...	1087	388.5	1920.0	Maribyrnong	-3

12211 rows × 23 columns

2. Consider the below data set

dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Milk', 'Apple', 'Kidney Beans', 'Eggs'],

['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],

['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

a. Find the frequent itemset from the dataset with support count = 4 and support count = 2.

b. Generate the association rules for support count 2 using the two methods 'confidence' and 'lift'

Code:

In [10]:
```
# Varun Sudhir 21BDS0040
# Question 2

from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

# Define the transactions
transactions = [
    ['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
    ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
    ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']
]

# Transform the transactions into a one-hot encoded DataFrame
encoder = TransactionEncoder()
encoded_array = encoder.fit(transactions).transform(transactions)
transaction_df = pd.DataFrame(encoded_array, columns=encoder.columns_)

# Display the DataFrame
print(transaction_df)
```

Output:

```
   Apple   Corn   Dill   Eggs  Ice cream  Kidney Beans   Milk  Nutmeg  Onion  \
0  False  False  False   True      False          True   True    True   True
1  False  False   True   True      False          True  False    True   True
2   True  False  False   True      False          True   True   False  False
3  False   True  False  False      False          True   True   False  False
4  False   True  False   True       True          True  False   False   True

   Unicorn  Yogurt
0    False    True
1    False    True
2    False   False
3     True    True
4    False   False
```

In [11]:
```python
# Varun Sudhir 21BDS0040
# A. Find the frequent itemsets from the dataset with support count = 4 and support count = 2:

# For support count = 4 (minimum support = 4/5 = 0.8)
frequent_itemsets_4 = apriori(transaction_df, min_support=0.8, use_colnames=True)
print("Frequent itemsets with support count 4:")
print(frequent_itemsets_4)

# For support count = 2 (minimum support = 2/5 = 0.4)
frequent_itemsets_2 = apriori(transaction_df, min_support=0.4, use_colnames=True)
print("\nFrequent itemsets with support count 2:")
print(frequent_itemsets_2)
```

Output:

```
Frequent itemsets with support count 4:
   support              itemsets
0      0.8                 (Eggs)
1      1.0          (Kidney Beans)
2      0.8  (Kidney Beans, Eggs)

Frequent itemsets with support count 2:
    support                                        itemsets
0       0.4                                          (Corn)
1       0.8                                          (Eggs)
2       1.0                                  (Kidney Beans)
3       0.6                                          (Milk)
4       0.4                                       (Nutmeg)
5       0.6                                        (Onion)
6       0.6                                       (Yogurt)
7       0.4                           (Kidney Beans, Corn)
8       0.8                           (Kidney Beans, Eggs)
9       0.4                                   (Milk, Eggs)
10      0.4                                (Nutmeg, Eggs)
11      0.6                                 (Onion, Eggs)
12      0.4                                (Eggs, Yogurt)
13      0.6                           (Kidney Beans, Milk)
14      0.4                         (Nutmeg, Kidney Beans)
15      0.6                          (Onion, Kidney Beans)
16      0.6                         (Kidney Beans, Yogurt)
17      0.4                                  (Milk, Yogurt)
18      0.4                               (Onion, Nutmeg)
19      0.4                              (Nutmeg, Yogurt)
20      0.4                               (Onion, Yogurt)
21      0.4                     (Kidney Beans, Milk, Eggs)
22      0.4                   (Nutmeg, Kidney Beans, Eggs)
23      0.6                    (Onion, Kidney Beans, Eggs)
24      0.4                   (Kidney Beans, Eggs, Yogurt)
25      0.4                         (Onion, Nutmeg, Eggs)
26      0.4                        (Nutmeg, Eggs, Yogurt)
27      0.4                         (Onion, Eggs, Yogurt)

28      0.4                    (Kidney Beans, Milk, Yogurt)
29      0.4                  (Onion, Nutmeg, Kidney Beans)
30      0.4                 (Nutmeg, Kidney Beans, Yogurt)
31      0.4                  (Onion, Kidney Beans, Yogurt)
32      0.4                        (Onion, Nutmeg, Yogurt)
33      0.4           (Onion, Nutmeg, Kidney Beans, Eggs)
34      0.4          (Nutmeg, Kidney Beans, Eggs, Yogurt)
35      0.4           (Onion, Kidney Beans, Eggs, Yogurt)
36      0.4                 (Onion, Nutmeg, Eggs, Yogurt)
37      0.4         (Onion, Nutmeg, Kidney Beans, Yogurt)
38      0.4  (Nutmeg, Onion, Yogurt, Kidney Beans, Eggs)
```

In [12]:
```
# Varun Sudhir 21BDS0040
#B. Generate the association rules for support count 2 using the two methods 'confidence' and 'lift':

# Generate rules using confidence
rules_confidence = association_rules(frequent_itemsets_2, metric="confidence", min_threshold=0)
print("\nAssociation rules using confidence:")
print(rules_confidence[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Generate rules using lift
rules_lift = association_rules(frequent_itemsets_2, metric="lift", min_threshold=0)
print("\nAssociation rules using lift:")
print(rules_lift[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Output:

```
Association rules using confidence:
       antecedents                            consequents  support  \
0    (Kidney Beans)                                 (Corn)      0.4
1           (Corn)                          (Kidney Beans)      0.4
2    (Kidney Beans)                                 (Eggs)      0.8
3           (Eggs)                          (Kidney Beans)      0.8
4           (Milk)                                 (Eggs)      0.4
..             ...                                    ...      ...
195        (Nutmeg)    (Onion, Eggs, Kidney Beans, Yogurt)      0.4
196         (Onion)   (Nutmeg, Eggs, Kidney Beans, Yogurt)      0.4
197        (Yogurt)    (Nutmeg, Onion, Kidney Beans, Eggs)      0.4
198  (Kidney Beans)          (Nutmeg, Onion, Eggs, Yogurt)      0.4
199         (Eggs)  (Nutmeg, Onion, Kidney Beans, Yogurt)      0.4

     confidence      lift
0      0.400000  1.000000
1      1.000000  1.000000
2      0.800000  1.000000
3      1.000000  1.000000
4      0.666667  0.833333
..          ...       ...
195    1.000000  2.500000
196    0.666667  1.666667
197    0.666667  1.666667
198    0.400000  1.000000
199    0.500000  1.250000

Association rules using lift:
       antecedents                            consequents  support  \
0    (Kidney Beans)                                 (Corn)      0.4
1           (Corn)                          (Kidney Beans)      0.4
2    (Kidney Beans)                                 (Eggs)      0.8
3           (Eggs)                          (Kidney Beans)      0.8
4           (Milk)                                 (Eggs)      0.4
..             ...                                    ...      ...
195        (Nutmeg)    (Onion, Eggs, Kidney Beans, Yogurt)      0.4
196         (Onion)   (Nutmeg, Eggs, Kidney Beans, Yogurt)      0.4
197        (Yogurt)    (Nutmeg, Onion, Kidney Beans, Eggs)      0.4
198  (Kidney Beans)          (Nutmeg, Onion, Eggs, Yogurt)      0.4
199         (Eggs)  (Nutmeg, Onion, Kidney Beans, Yogurt)      0.4

     confidence      lift
0      0.400000  1.000000
1      1.000000  1.000000
2      0.800000  1.000000
3      1.000000  1.000000
4      0.666667  0.833333
..          ...       ...
195    1.000000  2.500000
196    0.666667  1.666667
197    0.666667  1.666667
198    0.400000  1.000000
199    0.500000  1.250000

[200 rows x 5 columns]
```

3. Suppose if the dataset has a Nan value

dataset = [['Milk', 'Onion', Nan, 'Kidney Beans', 'Eggs', 'Yogurt'],

['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

['Milk', 'Apple', 'Kidney Beans', 'Eggs'],

['Milk', 'Unicorn', 'Corn', Nan, 'Yogurt'],

['Corn', 'Onion', Nan, 'Kidney Beans', 'Ice cream', 'Eggs']]

How does the frequent itemset change for the support count of 2

Code:

In [ ]:
```python
# Varun Sudhir 21BDS0040
# Question 3

import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.preprocessing import TransactionEncoder

# Dataset with NaN values
dataset = [
    ['Milk', 'Onion', np.nan, 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
    ['Milk', 'Unicorn', 'Corn', np.nan, 'Yogurt'],
    ['Corn', 'Onion', np.nan, 'Kidney Beans', 'Ice cream', 'Eggs']
]
```

In [13]:
```python
# Remove NaN values from each transaction
dataset = [[item for item in transaction if not pd.isna(item)] for transaction in transactions]

# Transform the cleaned dataset into a one-hot encoded DataFrame
encoder = TransactionEncoder()
encoded_array = encoder.fit(dataset).transform(dataset)
transaction_df = pd.DataFrame(encoded_array, columns=encoder.columns_)

# Find frequent itemsets with support count of 2 (minimum support = 2/5 = 0.4)
frequent_itemsets_2 = apriori(transaction_df, min_support=0.4, use_colnames=True)

# Print the frequent itemsets
print("Frequent itemsets with support count 2:")
print(frequent_itemsets_2)
```

Output:

Frequent itemsets with support count 2:

	support	itemsets
0	0.4	(Corn)
1	0.8	(Eggs)
2	1.0	(Kidney Beans)
3	0.6	(Milk)
4	0.4	(Nutmeg)
5	0.6	(Onion)
6	0.6	(Yogurt)
7	0.4	(Kidney Beans, Corn)
8	0.8	(Kidney Beans, Eggs)
9	0.4	(Milk, Eggs)
10	0.4	(Nutmeg, Eggs)
11	0.6	(Onion, Eggs)
12	0.4	(Eggs, Yogurt)
13	0.6	(Kidney Beans, Milk)
14	0.4	(Nutmeg, Kidney Beans)
15	0.6	(Onion, Kidney Beans)
16	0.6	(Kidney Beans, Yogurt)
17	0.4	(Milk, Yogurt)
18	0.4	(Onion, Nutmeg)
19	0.4	(Nutmeg, Yogurt)
20	0.4	(Onion, Yogurt)
21	0.4	(Kidney Beans, Milk, Eggs)
22	0.4	(Nutmeg, Kidney Beans, Eggs)
23	0.6	(Onion, Kidney Beans, Eggs)
24	0.4	(Kidney Beans, Eggs, Yogurt)
25	0.4	(Onion, Nutmeg, Eggs)
26	0.4	(Nutmeg, Eggs, Yogurt)
27	0.4	(Onion, Eggs, Yogurt)
28	0.4	(Kidney Beans, Milk, Yogurt)
29	0.4	(Onion, Nutmeg, Kidney Beans)
30	0.4	(Nutmeg, Kidney Beans, Yogurt)
31	0.4	(Onion, Kidney Beans, Yogurt)
32	0.4	(Onion, Nutmeg, Yogurt)
33	0.4	(Onion, Nutmeg, Kidney Beans, Eggs)
34	0.4	(Nutmeg, Kidney Beans, Eggs, Yogurt)
35	0.4	(Onion, Kidney Beans, Eggs, Yogurt)
36	0.4	(Onion, Nutmeg, Eggs, Yogurt)
37	0.4	(Onion, Nutmeg, Kidney Beans, Yogurt)
38	0.4	(Nutmeg, Onion, Yogurt, Kidney Beans, Eggs)