Date-31/10/2023

Team ID-698

Project Title-Market Basket Analysis for Fresh Product Location Improvement

```
import pandas as pd
import numpy as np

#for viz
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

#to avoid warning
import warnings
warnings.filterwarnings('ignore')

#to display all feature if the number increase
pd.set_option('display.max_columns', None)

data=pd.read_excel('/content/Assignment-1_Data.xlsx')
```

```
data.tail()
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
522059	581587	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
522060	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
522061	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
522062	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
522063	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

```
data.head()
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  datetime64[ns]
 4   Price       522064 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 27.9+ MB
```

```
data.shape
```

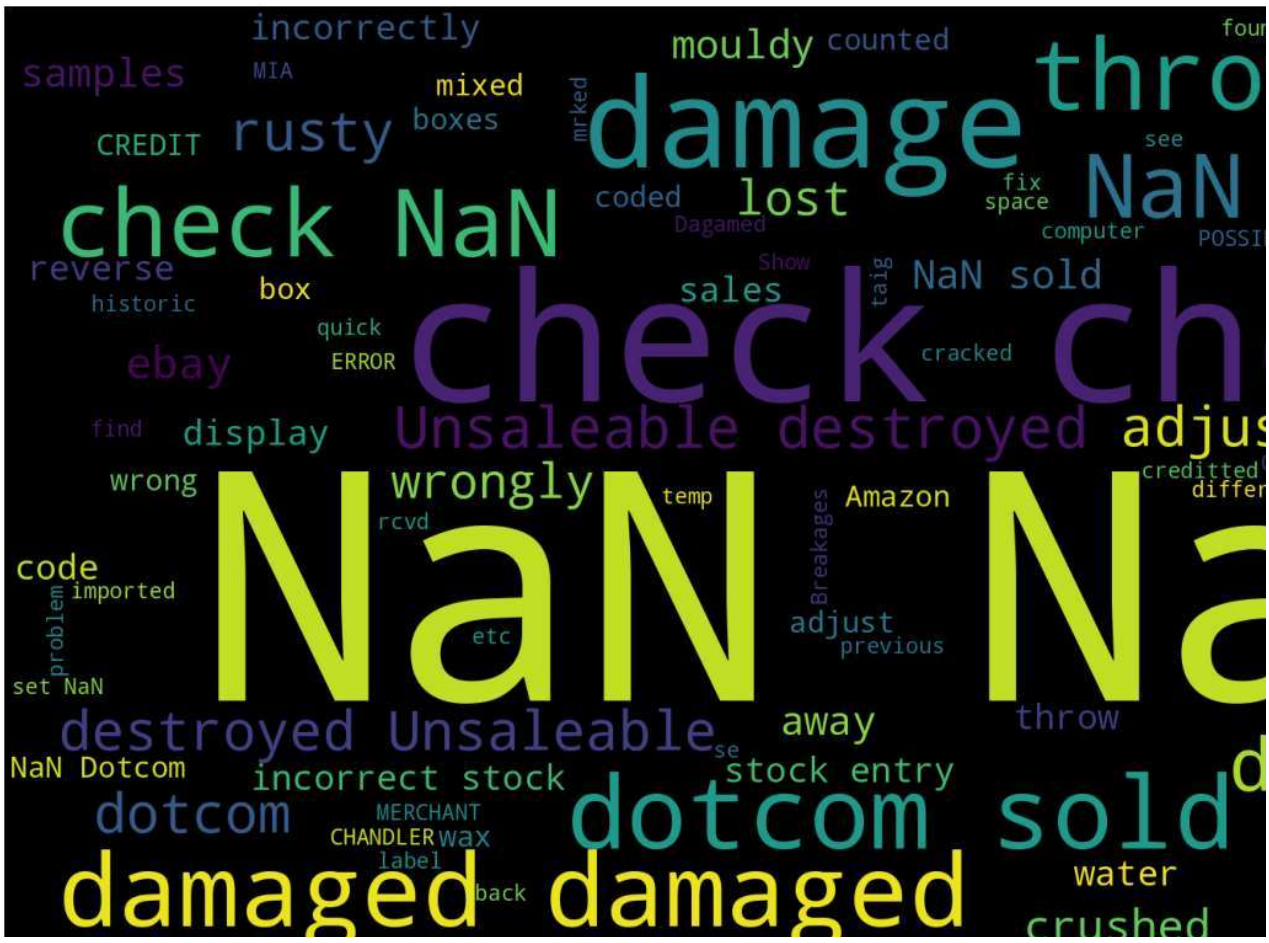
```
(522064, 7)
```

```
data.describe()
```

	Quantity	Price	CustomerID
count	522064.000000	522064.000000	388023.000000
mean	10.090435	3.826801	15316.931710
std	161.110525	41.900599	1721.846964
min	-9600.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13950.000000
50%	3.000000	2.080000	15265.000000
75%	10.000000	4.130000	16837.000000
max	80995.000000	13541.330000	18287.000000

```
from wordcloud import WordCloud, STOPWORDS
stopwords = STOPWORDS
worldcloud= WordCloud(background_color='Black',stopwords=stopwords, height=1000, width =2000)

temp=data[data['Quantity']<0]
body =temp['Itemname'].to_string(index=False)
### Generate word cloud
worldcloud.generate(body)
## Visualize
plt.figure(figsize=(22,10))
plt.imshow(worldcloud)
plt.axis("off")
```

```
(-0.5, 1999.5, 999.5, -0.5)
```



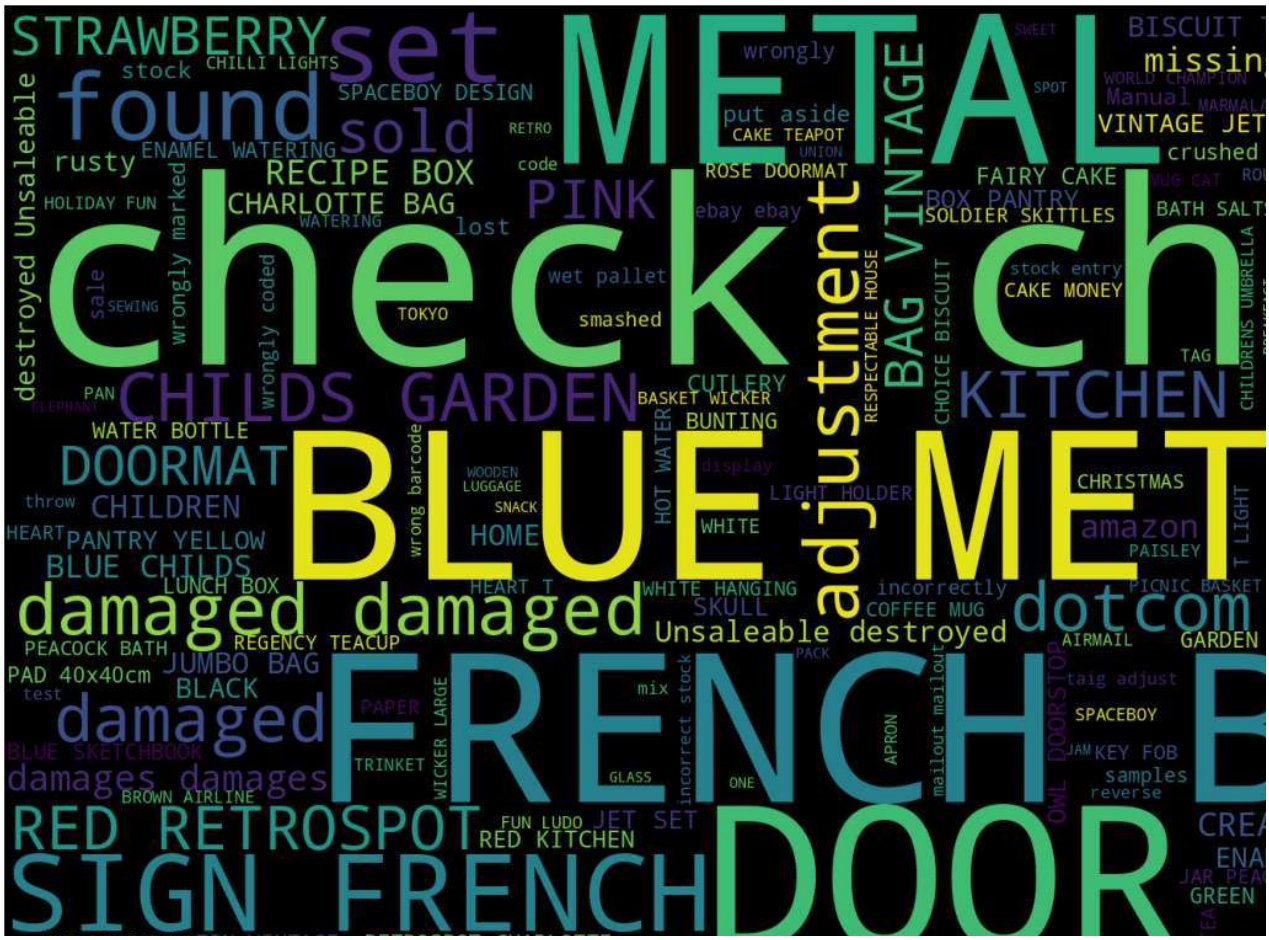
```
data[data['Quantity']<0]
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
2359	536589	NaN	-10	2010-12-01 16:50:00	0.0	NaN	United Kingdom
4289	536764	NaN	-38	2010-12-02 14:42:00	0.0	NaN	United Kingdom
6998	536996	NaN	-20	2010-12-03 15:30:00	0.0	NaN	United Kingdom
6999	536997	NaN	-20	2010-12-03 15:30:00	0.0	NaN	United Kingdom
7000	536998	NaN	-6	2010-12-03 15:30:00	0.0	NaN	United Kingdom
...	...	...	...	...	...	...	...
515634	581210	check	-26	2011-12-07 18:36:00	0.0	NaN	United Kingdom
515636	581212	lost	-1050	2011-12-07 18:38:00	0.0	NaN	United Kingdom
515637	581213	check	-30	2011-12-07 18:38:00	0.0	NaN	United Kingdom
517209	581226	missing	-338	2011-12-08 09:56:00	0.0	NaN	United Kingdom
519172	581422	smashed	-235	2011-12-08 15:24:00	0.0	NaN	United Kingdom

1336 rows × 7 columns

```
#next we can see for price small or equal to 0
temp=data[data['Price']<=0]
body =temp['Itemname'].dropna().to_string(index=False)
### Generate word cloud
worldcloud.generate(body)
## Visualize
plt.figure(figsize=(22,10))
plt.imshow(worldcloud)
plt.axis("off")
```

```
(-0.5, 1999.5, 999.5, -0.5)
```



```
# check for duplicate entries
data.duplicated().sum()
```

```
5286
```

```
# there are 5286 duplicates transcations are present in the dataset Lets remove them
data.drop_duplicates(inplace=True)
```

```
#Let remove the space in that word
data['Itemname'] = data['Itemname'].str.strip()
```

```
#Lets Check for null Values
data.isnull().sum()
```

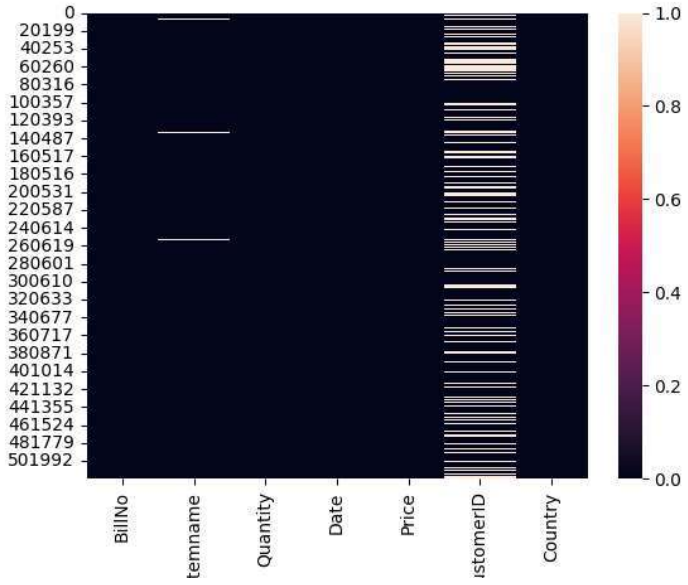
```
BillNo            0
Itemname        1455
Quantity           0
Date               0
Price              0
CustomerID    133967
Country            0
dtype: int64
```

```
data.isnull().mean()*100
```

```
BillNo         0.000000
Itemname       0.281552
Quantity       0.000000
Date           0.000000
Price          0.000000
CustomerID    25.923511
Country        0.000000
dtype: float64
```

```
sns.heatmap(data.isnull())
```

```
<Axes: >
```




```
data['Date']
```

```
0        2010-12-01 08:26:00
1        2010-12-01 08:26:00
2        2010-12-01 08:26:00
3        2010-12-01 08:26:00
4        2010-12-01 08:26:00
                 ...
522059   2011-12-09 12:50:00
522060   2011-12-09 12:50:00
522061   2011-12-09 12:50:00
522062   2011-12-09 12:50:00
522063   2011-12-09 12:50:00
Name: Date, Length: 516778, dtype: datetime64[ns]
```

```
#we can spearate the Data and time to different columns

import datetime as datetime
from datetime import datetime

#datetime.strptime('2013-01-01 09:10:12', '%Y-%m-%d %H:%M:%S')
data['date'] = data['Date'].dt.date
data['hour'] = data['Date'].dt.hour

### Converting invoice date to data time
data['date']= pd.to_datetime(data['date'], infer_datetime_format= True)
data.drop('Date',inplace=True,axis=1)

data.head(3)
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	17850.0	United Kingdom	2010-12-01	8
1	536365	WHITE METAL LANTERN	6	3.39	17850.0	United Kingdom	2010-12-01	8
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2.75	17850.0	United Kingdom	2010-12-01	8

```
data[data['Quantity']<=0]
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
2359	536589	NaN	-10	0.0	NaN	United Kingdom	2010-12-01	16
4289	536764	NaN	-38	0.0	NaN	United Kingdom	2010-12-02	14
6998	536996	NaN	-20	0.0	NaN	United Kingdom	2010-12-03	15
6999	536997	NaN	-20	0.0	NaN	United Kingdom	2010-12-03	15
7000	536998	NaN	-6	0.0	NaN	United Kingdom	2010-12-03	15
...	...	...	...	...	...	...	...	...
515634	581210	check	-26	0.0	NaN	United Kingdom	2011-12-07	18
515636	581212	lost	-1050	0.0	NaN	United Kingdom	2011-12-07	18
515637	581213	check	-30	0.0	NaN	United Kingdom	2011-12-07	18
517209	581226	missing	-338	0.0	NaN	United Kingdom	2011-12-08	9
519172	581422	smashed	-235	0.0	NaN	United Kingdom	2011-12-08	15

1336 rows × 8 columns

```
data[data['Price']<=0]
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
613	536414	NaN	56	0.0	NaN	United Kingdom	2010-12-01	11
1937	536545	NaN	1	0.0	NaN	United Kingdom	2010-12-01	14
1938	536546	NaN	1	0.0	NaN	United Kingdom	2010-12-01	14
1939	536547	NaN	1	0.0	NaN	United Kingdom	2010-12-01	14
1940	536549	NaN	1	0.0	NaN	United Kingdom	2010-12-01	14
...	...	...	...	...	...	...	...	...
517266	581234	NaN	27	0.0	NaN	United Kingdom	2011-12-08	10
518770	581406	POLYESTER FILLER PAD 45x45cm	240	0.0	NaN	United Kingdom	2011-12-08	13
518771	581406	POLYESTER FILLER PAD 40x40cm	300	0.0	NaN	United Kingdom	2011-12-08	13
518820	581408	NaN	20	0.0	NaN	United Kingdom	2011-12-08	14
519172	581422	smashed	-235	0.0	NaN	United Kingdom	2011-12-08	15

```
#remove the rows which has the buyed quality is small or equal to zero
data=data[data['Quantity']>0]

#remove the rows which price is small or equal to zero
data=data[data['Price']>0]
data.shape
```

```
(514270, 8)
```

```
# Lets Check for Quantity
data.sort_values(by='Quantity',ascending=False)
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
520583	581483	PAPER CRAFT , LITTLE BIRDIE	80995	2.08	16446.0	United Kingdom	2011-12-09	9
59999	541431	MEDIUM CERAMIC TOP STORAGE JAR	74215	1.04	12346.0	United Kingdom	2011-01-18	10
405138	573008	WORLD WAR 2 GLIDERS ASSTD DESIGNS	4800	0.21	12901.0	United Kingdom	2011-10-27	12
198929	554868	SMALL POPCORN HOLDER	4300	0.72	13135.0	United Kingdom	2011-05-27	10
94245	544612	EMPIRE DESIGN ROSETTE	3906	0.82	18087.0	United Kingdom	2011-02-22	10
...	...	...	...	...	...	...	...	...
382839	571243	LE GRAND TRAY CHIC SET	1	12.50	14595.0	United Kingdom	2011-10-14	15
382842	571243	REGENCY CAKESTAND 3 TIER	1	12.75	14595.0	United Kingdom	2011-10-14	15
160161	550835	DOORMAT ENGLISH ROSE	1	7.95	15034.0	United Kingdom	2011-04-21	10
160164	550836	CLASSIC METAL BIRDCAGE PLANT HOLDER	1	12.75	14759.0	United Kingdom	2011-04-21	10
356890	569208	GLASS APOTHECARY BOTTLE PERFUME	1	3.95	16693.0	United Kingdom	2011-10-02	11

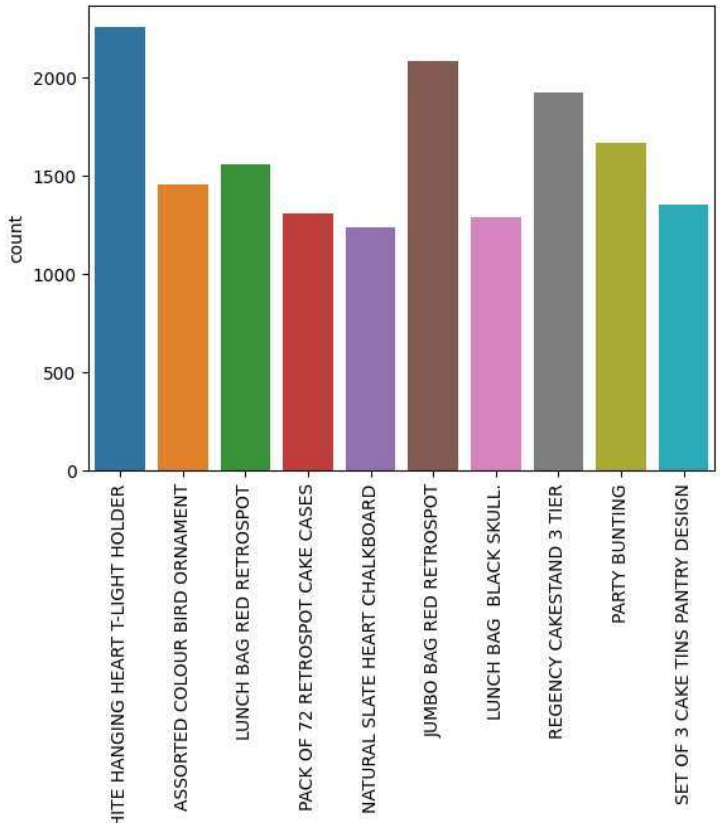
```
#Lets check For Price
data.sort_values(by='Price',ascending=False)
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
14696	537632	AMAZON FEE	1	13541.330	NaN	United Kingdom	2010-12-07	15
288772	A563185	Adjust bad debt	1	11062.060	NaN	United Kingdom	2011-08-12	14
167329	551697	POSTAGE	1	8142.750	16029.0	United Kingdom	2011-05-03	13
286674	562955	DOTCOM POSTAGE	1	4505.170	NaN	United Kingdom	2011-08-11	10
258372	560373	Manual	1	4287.630	NaN	United Kingdom	2011-07-18	12
...	...	...	...	...	...	...	...	...
504147	580513	POPART WOODEN PENCILS ASST	100	0.040	14456.0	United Kingdom	2011-12-04	13
346188	568200	PADS TO MATCH ALL CUSHIONS	1	0.001	16198.0	United Kingdom	2011-09-25	14
347944	568375	Bank Charges	1	0.001	13405.0	United Kingdom	2011-09-26	17
268879	561226	PADS TO MATCH ALL CUSHIONS	1	0.001	15618.0	United Kingdom	2011-07-26	10
151547	550193	PADS TO MATCH ALL CUSHIONS	1	0.001	13952.0	United Kingdom	2011-04-15	9

514270 rows × 8 columns

```
data=data[data['Price']<5000]
data=data[data['Quantity']<5000]

# Get the top 10 item names by count
top_10_items = data['Itemname'].value_counts().nlargest(10).index

# Filter the DataFrame to include only the top 10 item names
df_top_10 = data[data['Itemname'].isin(top_10_items)]

# Create a countplot for the top 10 item names
ax=sns.countplot(data=df_top_10, x='Itemname')
plt.xticks(rotation=90)
```

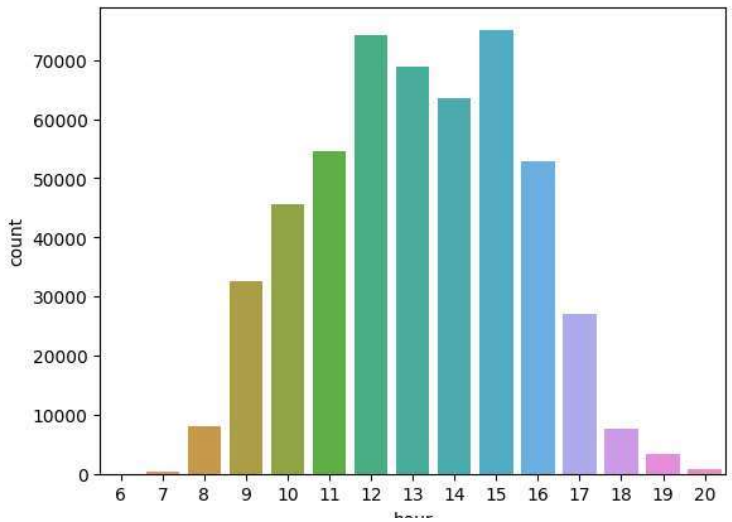
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'WHITE HANGING HEART T-LIGHT HOLDER'),
  Text(1, 0, 'ASSORTED COLOUR BIRD ORNAMENT'),
  Text(2, 0, 'LUNCH BAG RED RETROSPOT'),
  Text(3, 0, 'PACK OF 72 RETROSPOT CAKE CASES'),
  Text(4, 0, 'NATURAL SLATE HEART CHALKBOARD'),
  Text(5, 0, 'JUMBO BAG RED RETROSPOT'),
  Text(6, 0, 'LUNCH BAG  BLACK SKULL.'),
  Text(7, 0, 'REGENCY CAKESTAND 3 TIER'),
  Text(8, 0, 'PARTY BUNTING'),
  Text(9, 0, 'SET OF 3 CAKE TINS PANTRY DESIGN')])
```



#lets do some viz

```
sns.countplot(data=data,x='hour')
```

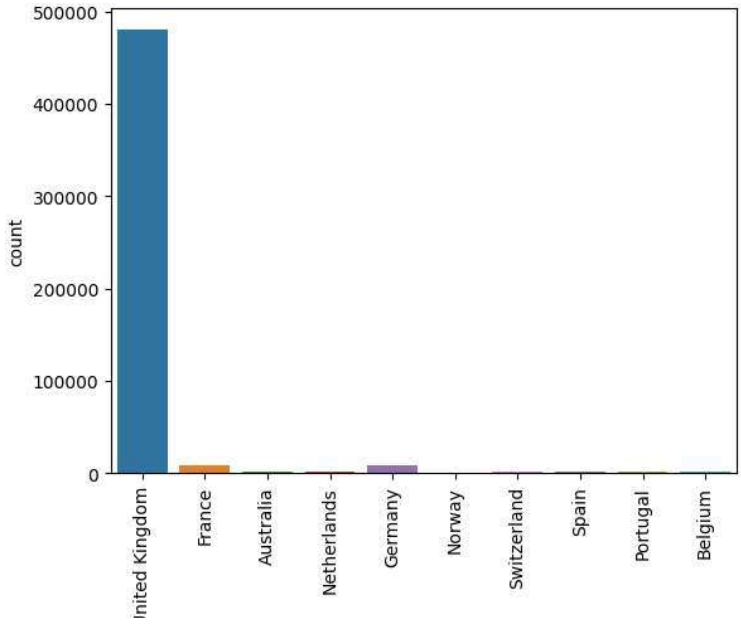
```
<Axes: xlabel='hour', ylabel='count'>
```




```
# Get the top 5 country names by count
top_5_country = data['Country'].value_counts().nlargest(10).index
```

```
# Filter the DataFrame to include only the top 10 item names
df_top_5 = data[data['Country'].isin(top_5_country)]

# Create a countplot for the top 10 item names
ax=sns.countplot(data=df_top_5, x='Country')
plt.xticks(rotation=90)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'United Kingdom'),
  Text(1, 0, 'France'),
  Text(2, 0, 'Australia'),
  Text(3, 0, 'Netherlands'),
  Text(4, 0, 'Germany'),
  Text(5, 0, 'Norway'),
  Text(6, 0, 'Switzerland'),
  Text(7, 0, 'Spain'),
  Text(8, 0, 'Portugal'),
  Text(9, 0, 'Belgium')])
```




```
data.Country.value_counts().head(10)
```

```
United Kingdom    479903
Germany             9025
France              8392
Spain               2479
Netherlands         2359
Belgium             2031
Switzerland         1958
Portugal            1492
Australia           1181
Norway              1071
Name: Country, dtype: int64
```

```
#lets import the models
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

mybasket= (data[data['Country'] =="Germany"]
          .groupby(['BillNo', 'Itemname'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('BillNo'))

mybasket
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
```

Itemname	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE SKULLS	12 PENCILS TALL TUBE WOODLAND	12 PINK HEN+CHICKS IN BASKET	I
BillNo													
536527	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
536840	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
536861	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
536967	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
536983	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
581266	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
581494	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
581570	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
581574	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

```
a=data[data['Country']=='Germany']
a['BillNo'].nunique()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
457
```

```
def my_encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

my_basket = mybasket.applymap(my_encode_units)
my_basket.drop('POSTAGE', inplace=True, axis=1) #Remove "postage" as an item
### Display sample of set
my_basket
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
```

```
data[data['BillNo']==536527]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

	BillNo	Itemname	Quantity	Price	CustomerID	Country	date	hour
1099	536527	SET OF 6 T-LIGHTS SANTA	6	2.95	12662.0	Germany	2010-12-01	13
1100	536527	ROTATING SILVER ANGELS T-LIGHT HLDR	6	2.55	12662.0	Germany	2010-12-01	13
1101	536527	MULTI COLOUR SILVER T-LIGHT HOLDER	12	0.85	12662.0	Germany	2010-12-01	13
1102	536527	5 HOOK HANGER MAGIC TOADSTOOL	12	1.65	12662.0	Germany	2010-12-01	13
1103	536527	3 HOOK HANGER MAGIC GARDEN	12	1.95	12662.0	Germany	2010-12-01	13
1104	536527	5 HOOK HANGER RED MAGIC TOADSTOOL	12	1.65	12662.0	Germany	2010-12-01	13
1105	536527	ASSORTED COLOUR LIZARD SUCTION HOOK	24	0.42	12662.0	Germany	2010-12-01	13
1106	536527	JUMBO BAG WOODLAND ANIMALS	10	1.95	12662.0	Germany	2010-12-01	13
1107	536527	JUMBO BAG OWLS	10	1.95	12662.0	Germany	2010-12-01	13
1108	536527	HOT WATER BOTTLE BABUSHKA	4	4.65	12662.0	Germany	2010-12-01	13
1109	536527	HOMEMADE JAM SCENTED CANDLES	12	1.45	12662.0	Germany	2010-12-01	13
1110	536527	CHILDREN'S CIRCUS PARADE MUG	12	1.65	12662.0	Germany	2010-12-01	13
1111	536527	PACK 3 FIRE ENGINE/CAR PATCHES	12	1.25	12662.0	Germany	2010-12-01	13
1112	536527	PICTURE DOMINOES	12	1.45	12662.0	Germany	2010-12-01	13

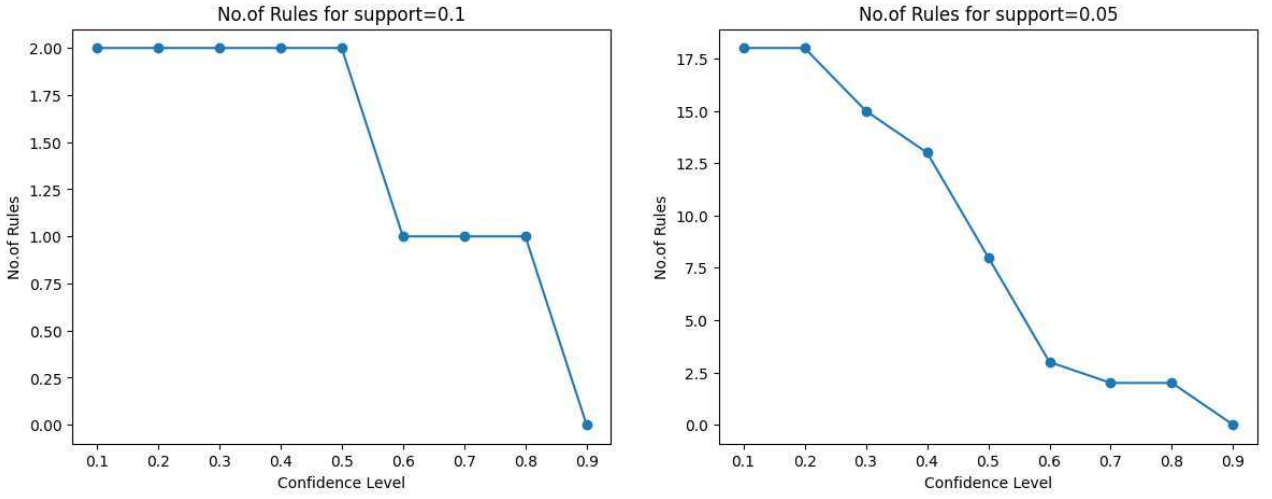
```
support=[0.1, 0.05, 0.01]
confidenceLevels=[0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]

# Empty lists
rules_sup10=[0]*9
rules_sup5=[0]*9
rules_sup1=[0]*9

#rules for support 0.1
my_frequent_itemsets01 = apriori(my_basket, min_support=0.1, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup10[i]=len(association_rules(my_frequent_itemsets01, metric="confidence", min_thresh

#rules for support 0.05
my_frequent_itemsets005 = apriori(my_basket, min_support=0.05, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup5[i]=len(association_rules(my_frequent_itemsets005, metric="confidence", min_thresh

#rules for support 0.01
my_frequent_itemsets001 = apriori(my_basket, min_support=0.01, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup1[i]=len(association_rules(my_frequent_itemsets001, metric="confidence", min_thresh
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool typ
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool typ
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool typ
  warnings.warn(
```

```
# Plot the data
plt.figure(figsize=(22,5))
plt.subplot(1,3,1)
plt.plot(confidenceLevels,rules_sup10, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.1")
```

```
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.subplot(1,3,2)
plt.plot(confidenceLevels,rules_sup5, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.05")
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.subplot(1,3,3)
plt.plot(confidenceLevels,rules_sup1, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.001")
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.savefig("comparision")
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```



Let's analyze the results,

- Support level of 10% = We only identify a few rules with very low confidence levels. This means that there are no relatively frequent associations in our data set. We can't choose this value, the resulting rules are unrepresentative.
- Support level of 5% = We only identify a rule with a confidence of at least 50%. It seems that we have to look for support levels below 5% to obtain a greater number of rules with a reasonable confidence.
- Support level of 1% = We started to get dozens of rules

```
my_frequent_itemsets = apriori(my_basket, min_support=0.07, use_colnames=True)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool typ
  warnings.warn(
```

```
myrules=association_rules(my_frequent_itemsets, metric="lift", min_threshold=1)
```

```
myrules
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_
  and should_run_async(code)
```

```
mybasket= (data[data['Country'] =="France"]
          .groupby(['BillNo', 'Itemname'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('BillNo'))

mybasket

def my_encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

my_basket = mybasket.applymap(my_encode_units)
my_basket.drop('POSTAGE', inplace=True, axis=1) #Remove "postage" as an item
### Display sample of set
my_basket
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

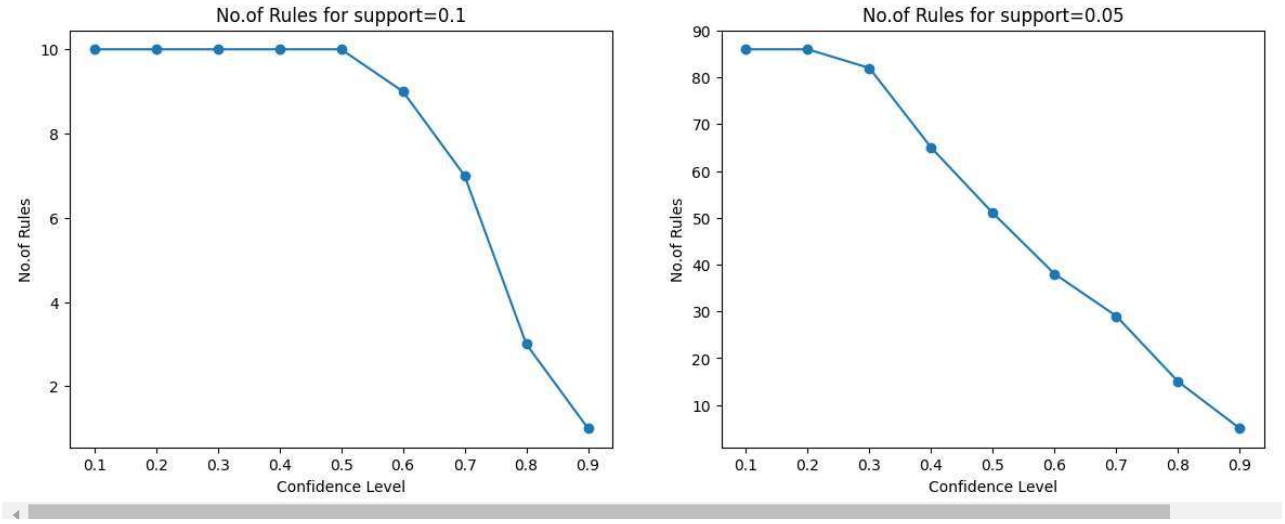
Itemname	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE WOODLAND	15CM CHRISTMAS GLASS BALL 20 LIGHTS	16 PIECE CUTLERY SET PANTRY DESIGN	DIS
BillNo													
536370	0	0	0	0	0	0	0	0	0	0	0	0	
536852	0	0	0	0	0	0	0	0	0	0	0	0	
536974	0	0	0	0	0	0	0	0	0	0	0	0	
537065	0	0	0	0	0	0	0	0	0	0	0	0	
537463	0	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
580986	0	0	0	0	0	0	0	0	0	0	0	0	
581001	0	0	0	0	0	0	0	0	0	0	0	0	
581171	0	0	0	0	0	0	0	0	0	0	0	0	
581279	0	0	0	0	0	0	0	0	0	0	0	0	
581587	0	0	0	0	0	0	0	0	0	0	0	0	

```
support=[0.1, 0.05, 0.01]
confidenceLevels=[0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]

# Empty lists
rules_sup10=[0]*9
rules_sup5=[0]*9
rules_sup1=[0]*9

#rules for support 0.1
my_frequent_itemsets01 = apriori(my_basket, min_support=0.1, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup10[i]=len(association_rules(my_frequent_itemsets01, metric="confidence", min_thresh

#rules for support 0.05
my_frequent_itemsets005 = apriori(my_basket, min_support=0.05, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup5[i]=len(association_rules(my_frequent_itemsets005, metric="confidence", min_thresh
```

```
#rules for support 0.01
my_frequent_itemsets001 = apriori(my_basket, min_support=0.01, use_colnames=True)
for i in range(len(confidenceLevels)):
    rules_sup1[i]=len(association_rules(my_frequent_itemsets001, metric="confidence", min_thresh
```

```
# Plot the data
plt.figure(figsize=(22,5))
plt.subplot(1,3,1)
plt.plot(confidenceLevels,rules_sup10, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.1")
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.subplot(1,3,2)
plt.plot(confidenceLevels,rules_sup5, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.05")
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.subplot(1,3,3)
plt.plot(confidenceLevels,rules_sup1, marker='o', label='Support=0.1')
plt.title("No.of Rules for support=0.001")
plt.xlabel("Confidence Level")
plt.ylabel("No.of Rules")
plt.savefig("comparision")
#support =0.1 means if there are 100 trnascation altest 10 transcation will have that item
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool type
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool type
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool type
  warnings.warn(
```




```
#lets chosse support=0.1
my_frequent_itemsets = apriori(my_basket, min_support=0.10, use_colnames=True)
myrules=association_rules(my_frequent_itemsets, metric="lift", min_threshold=1)
myrules
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-bool type
  warnings.warn(
```

	antecedents	consequents	antecedent support	consequent support	support
0	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.170918	0.168367	0.102041
1	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.168367	0.170918	0.102041
2	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.170918	0.137755	0.104592
3	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.137755	0.170918	0.104592
4	(SET/6 RED SPOTTY PAPER CUPS)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.137755	0.132653	0.102041
5	(SET/20 RED RETROSPOT PAPER NAPKINS)	(SET/6 RED SPOTTY PAPER CUPS)	0.132653	0.137755	0.102041
6	(SET/20 RED RETROSPOT PAPER NAPKINS)	(SET/6 RED SPOTTY PAPER PLATES)	0.132653	0.127551	0.102041
7	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041
8	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449