Introduction

The Bread Basket, a bakery situated in Edinburgh, this is the market basket analysis based on the dataset consisting of 20507 records with 4 columns, which represents over 9000 transactions.

Objectives

- Download a DataSet from *.csv files
- Create new and recalculate values of existing columns
- Transform a DataSet of transactions into a market basket DataSet
- Visualize data with seaborn
- Produce Association rules
- Analyze market basket
- Visualize graph of association rules

Requirements

- [Python]
- [Pandas]
- [SeaBorn]
- [mlxtend]
- [pyvis]

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
from pyvis.network import Network
import datetime as dt
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-GPXX0R8UEN/bread%20basket.csv")
df
```

```
       Transaction           Item         date_time period_day
weekday_weekend
0                1          Bread  30-10-2016 09:58    morning
weekend
1                2   Scandinavian  30-10-2016 10:05    morning
weekend
```

```
2                2   Scandinavian  30-10-2016 10:05    morning
weekend
3                3  Hot chocolate  30-10-2016 10:07    morning
weekend
4                3            Jam  30-10-2016 10:07    morning
weekend
...            ...            ...               ...        ...
...
20502         9682         Coffee  09-04-2017 14:32  afternoon
weekend
20503         9682            Tea  09-04-2017 14:32  afternoon
weekend
20504         9683         Coffee  09-04-2017 14:57  afternoon
weekend
20505         9683         Pastry  09-04-2017 14:57  afternoon
weekend
20506         9684      Smoothies  09-04-2017 15:04  afternoon
weekend

[20507 rows x 5 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20507 entries, 0 to 20506
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Transaction      20507 non-null  int64 
 1   Item             20507 non-null  object
 2   date_time        20507 non-null  object
 3   period_day       20507 non-null  object
 4   weekday_weekend  20507 non-null  object
dtypes: int64(1), object(4)
memory usage: 801.2+ KB
```

1. Transaction: the transaction id which is unique for each order
2. Item: a list of items to be ordered/placed by customer
3. date_time: the date and time of the transaction.
4. period_day: the period of the day when a customer ordered/placed
5. weekday_weekend: is the day is weekend (sat or sun) or a weekday.

Transforming the data type of the date_time column.

```
df['date_time']=pd.to_datetime(df['date_time'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20507 entries, 0 to 20506
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
```

```
---  ------           -------------  -----
 0   Transaction      20507 non-null  int64
 1   Item             20507 non-null  object
 2   date_time        20507 non-null  datetime64[ns]
 3   period_day       20507 non-null  object
 4   weekday_weekend  20507 non-null  object
dtypes: datetime64[ns](1), int64(1), object(3)
memory usage: 801.2+ KB
```

```
df['time']=df['date_time'].dt.time
df['hour']=df['date_time'].dt.hour
```

```
df['month'] = df['date_time'].dt.month
df['month name'] = df['month'].replace([1,2,3,4,5,6,7,8,9,10,11,12],
['January','February','March','April','May','June','July','August','Se
ptember','October','November','December'])
```

```
df['day'] = df['date_time'].dt.day
df['weekday'] = df['date_time'].dt.weekday
df['weekday name'] = df['weekday'].replace([0,1,2,3,4,5,6],
['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday
'])
```

```
df
```

```
       Transaction           Item           date_time period_day  \
0                1          Bread 2016-10-30 09:58:00    morning
1                2   Scandinavian 2016-10-30 10:05:00    morning
2                2   Scandinavian 2016-10-30 10:05:00    morning
3                3  Hot chocolate 2016-10-30 10:07:00    morning
4                3            Jam 2016-10-30 10:07:00    morning
...            ...            ...                 ...        ...
20502         9682         Coffee 2017-09-04 14:32:00  afternoon
20503         9682            Tea 2017-09-04 14:32:00  afternoon
20504         9683         Coffee 2017-09-04 14:57:00  afternoon
20505         9683         Pastry 2017-09-04 14:57:00  afternoon
20506         9684      Smoothies 2017-09-04 15:04:00  afternoon

      weekday_weekend      time  hour  month month name  day  weekday
\
0             weekend  09:58:00     9     10    October   30        6

1             weekend  10:05:00    10     10    October   30        6

2             weekend  10:05:00    10     10    October   30        6

3             weekend  10:07:00    10     10    October   30        6

4             weekend  10:07:00    10     10    October   30        6

...               ...       ...   ...    ...        ...  ...      ...
```

```
20502         weekend  14:32:00    14      9  September    4        0

20503         weekend  14:32:00    14      9  September    4        0

20504         weekend  14:57:00    14      9  September    4        0

20505         weekend  14:57:00    14      9  September    4        0

20506         weekend  15:04:00    15      9  September    4        0
```

```
      weekday name
0           Sunday
1           Sunday
2           Sunday
3           Sunday
4           Sunday
...            ...
20502       Monday
20503       Monday
20504       Monday
20505       Monday
20506       Monday

[20507 rows x 12 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20507 entries, 0 to 20506
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Transaction      20507 non-null  int64
 1   Item             20507 non-null  object
 2   date_time        20507 non-null  datetime64[ns]
 3   period_day       20507 non-null  object
 4   weekday_weekend  20507 non-null  object
 5   time             20507 non-null  object
 6   hour             20507 non-null  int64
 7   month            20507 non-null  int64
 8   month name       20507 non-null  object
 9   day              20507 non-null  int64
 10  weekday          20507 non-null  int64
 11  weekday name     20507 non-null  object
dtypes: datetime64[ns](1), int64(5), object(6)
memory usage: 1.9+ MB
```

11 columns with all necessary information for preliminary visual market basket analysis.

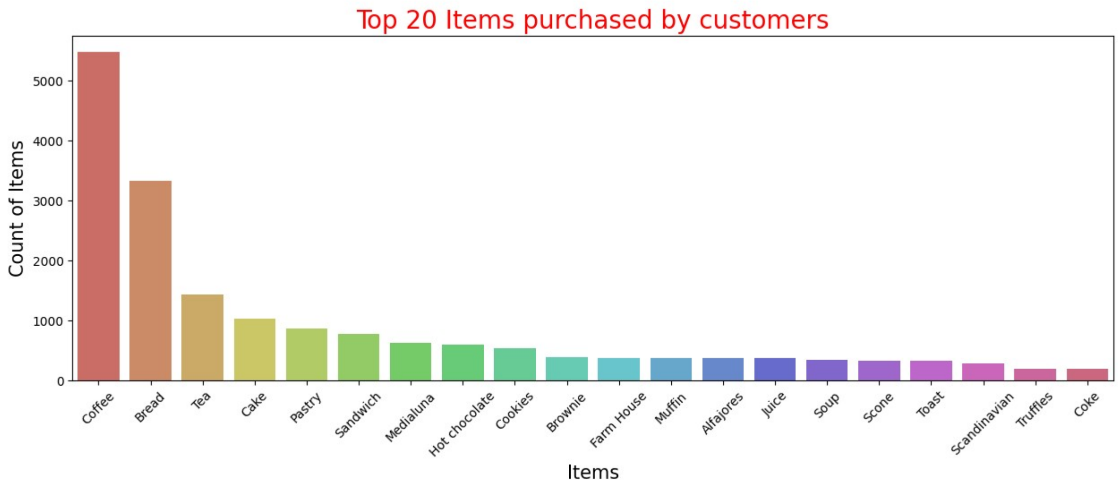
Data Visualizations

Top 20 most popular purchases.

```
popular = df['Item'].value_counts()
(df['Item'].value_counts(normalize=True)*100).head(20)
```

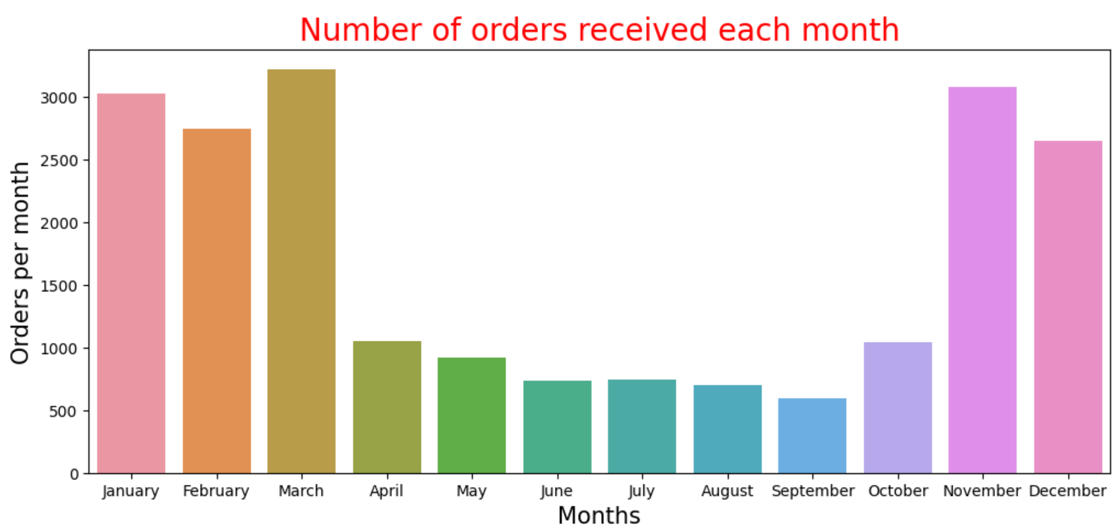
```
Coffee           26.678695
Bread            16.213976
Tea               6.997611
Cake              4.998293
Pastry            4.174184
Sandwich          3.759692
Medialuna         3.003852
Hot chocolate     2.877066
Cookies           2.633247
Brownie           1.848149
Farm House        1.823767
Muffin            1.804262
Alfajores         1.799386
Juice             1.799386
Soup              1.667723
Scone             1.594577
Toast             1.550690
Scandinavian      1.350758
Truffles          0.941142
Coke              0.902131
Name: Item, dtype: float64
```

```
plt.figure(figsize=(15,5))
sns.barplot(x = popular.head(20).index, y = popular.head(20).values,
palette = 'hls')
plt.xlabel('Items', size = 15)
plt.xticks(rotation=45)
plt.ylabel('Count of Items', size = 15)
plt.title('Top 20 Items purchased by customers', color = 'red', size =
20)
plt.show()
```



The dynamics of monthly purchases.

```
monthTran = df.groupby(['month','month name'])
['Transaction'].count().reset_index()
plt.figure(figsize=(12,5))
sns.barplot(data = monthTran[['month name', 'Transaction']], x =
"month name", y = "Transaction")
plt.xlabel('Months', size = 15)
plt.ylabel('Orders per month', size = 15)
plt.title('Number of orders received each month', color = 'red', size
= 20)
plt.show()
```

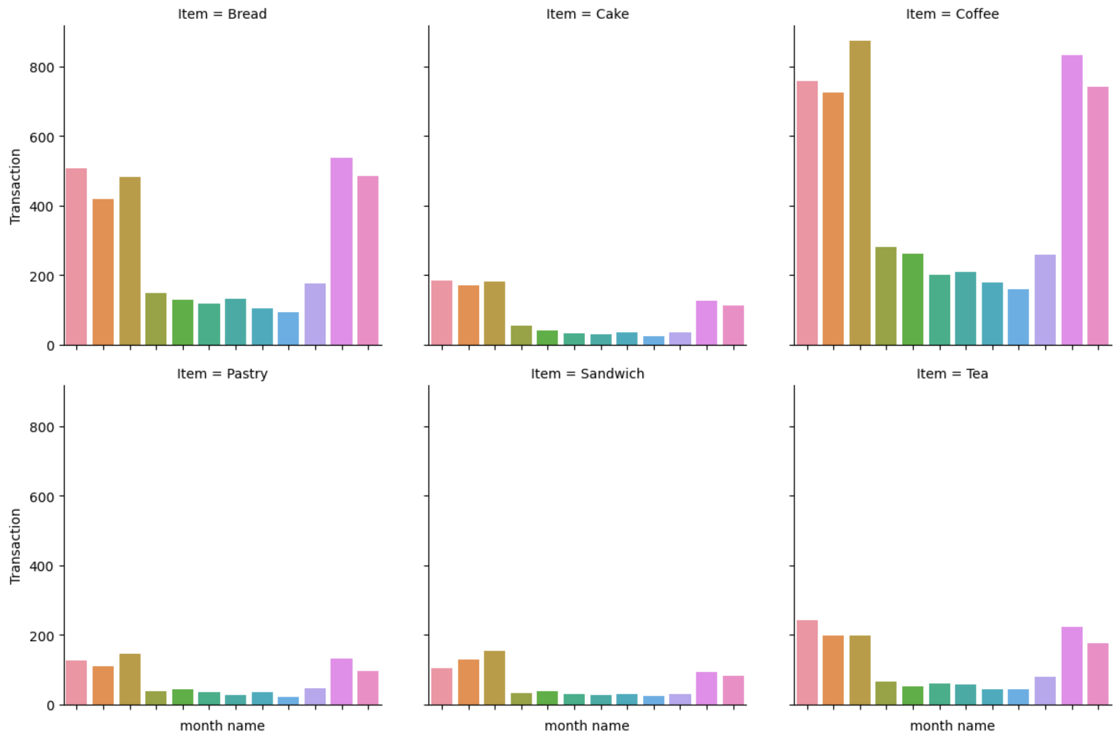


Monthly purchases for the six most popular products.

```
monthTranTransaction =
df[df.Item.isin(popular.head(6).index)].groupby(['month','month
name','Item'])['Transaction'].count().reset_index()

ax = sns.catplot(x="month name", y="Transaction",
```

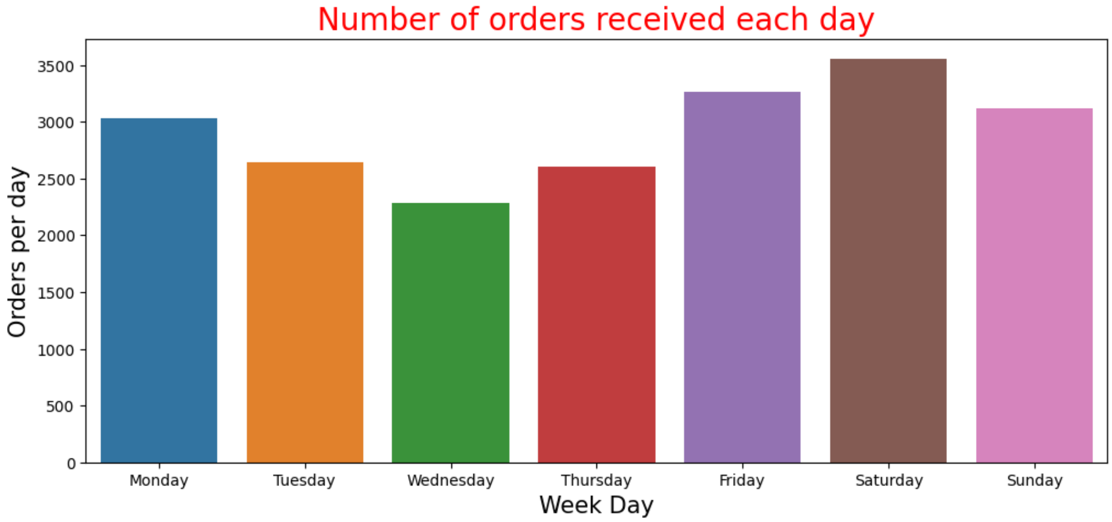
```
                 col="Item",
                data=monthTranTransaction, kind="bar",
                height=4,  col_wrap=3)
ax.set_xticklabels(rotation=45)
```

```
<seaborn.axisgrid.FacetGrid at 0x1dd5ec6cee0>
```



The weekly activity.

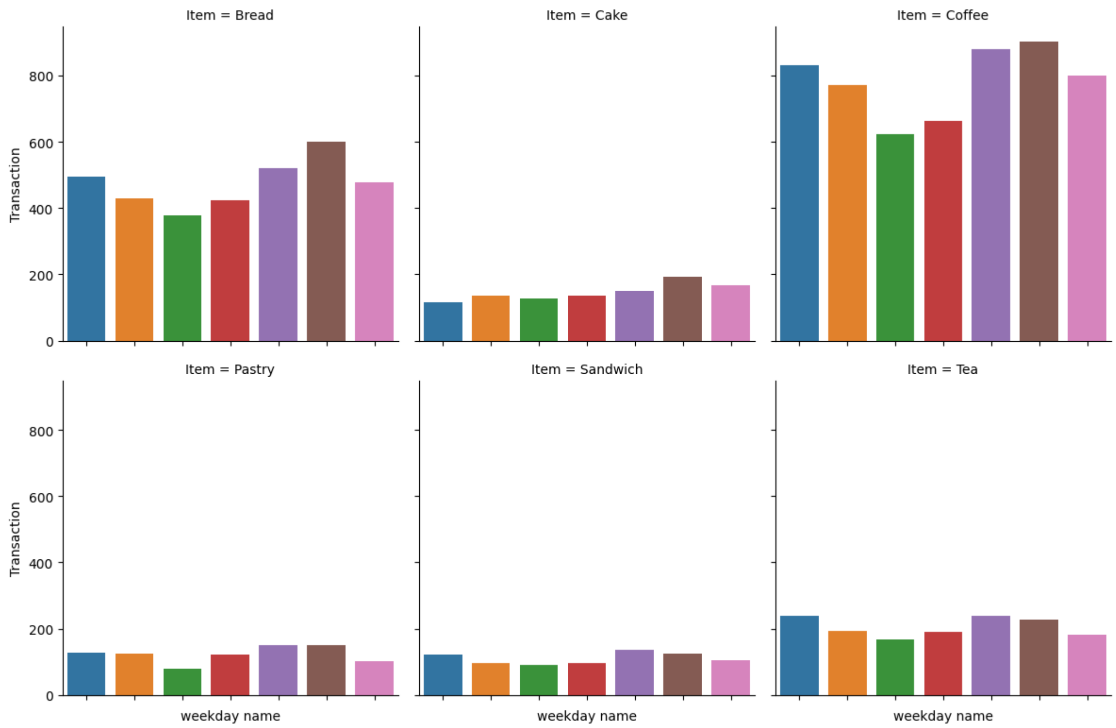
```
weekTran = weekTran = df.groupby(['weekday','weekday name'])
['Transaction'].count().reset_index()

plt.figure(figsize=(12,5))
sns.barplot(data = weekTran[['weekday name', 'Transaction']], x =
"weekday name", y = "Transaction")
plt.xlabel('Week Day', size = 15)
plt.ylabel('Orders per day', size = 15)
plt.title('Number of orders received each day', color = 'red', size =
20)
plt.show()
```



The six most popular products

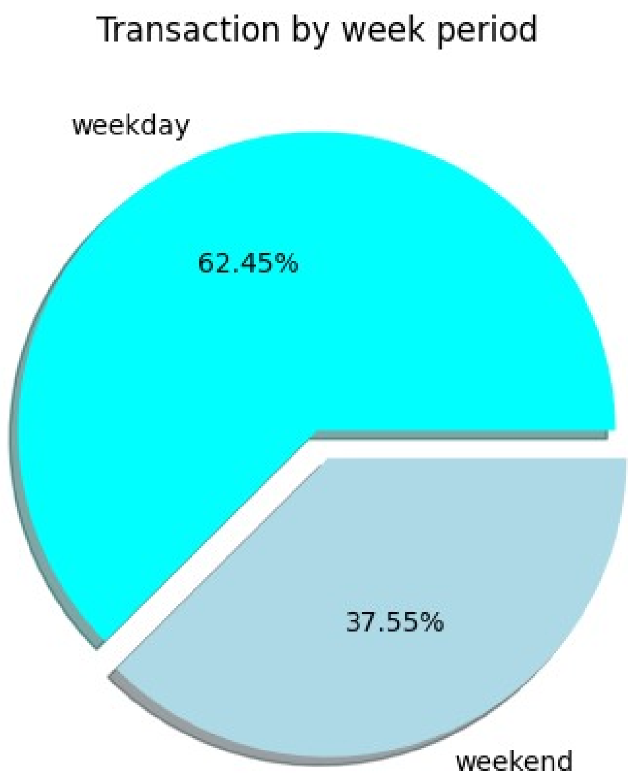
```
weekTran =
df[df.Item.isin(popular.head(6).index)].groupby(['weekday','weekday
name', 'Item'])['Transaction'].count().reset_index()

ax = sns.catplot(x="weekday name", y="Transaction",
                  col="Item",
                 data=weekTran, kind="bar",
                 height=4,  col_wrap=3)
ax.set_xticklabels(rotation=45)
```

```
<seaborn.axisgrid.FacetGrid at 0x1dd748bcb50>
```

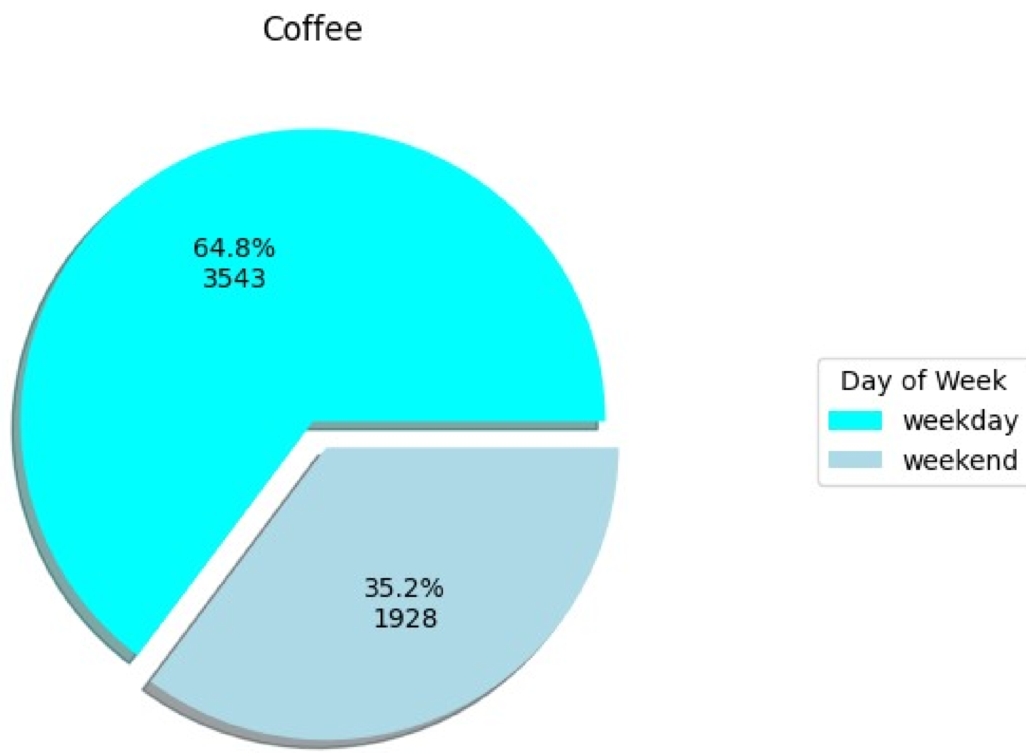


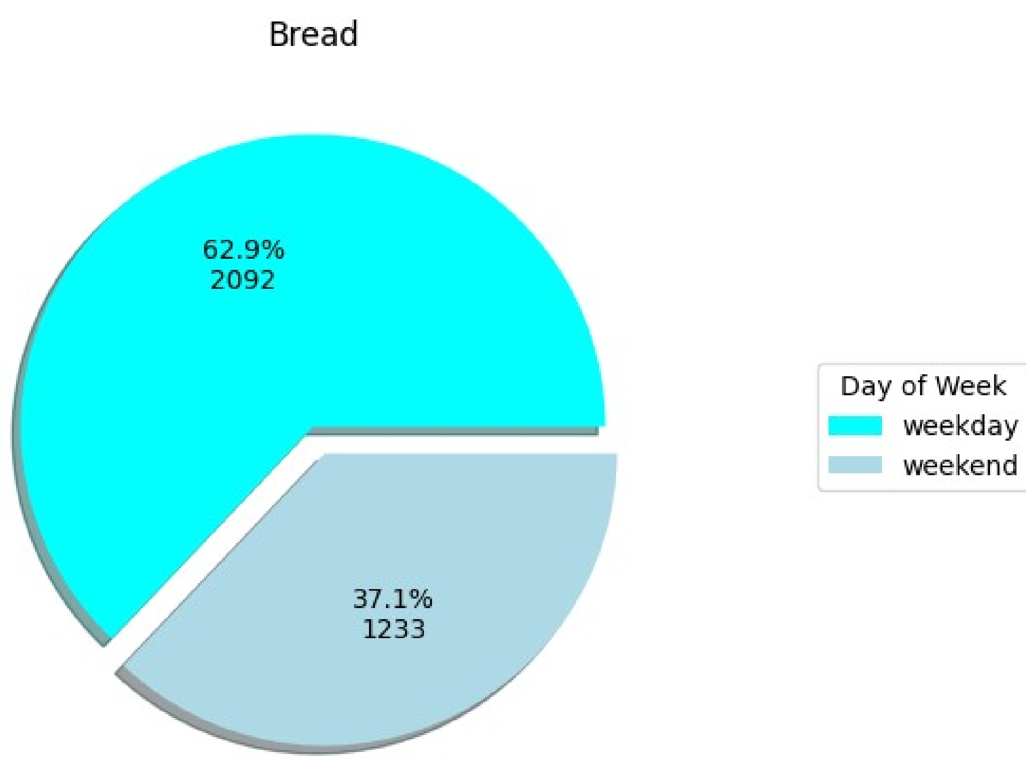
The share of purchases on weekends and weekdays.

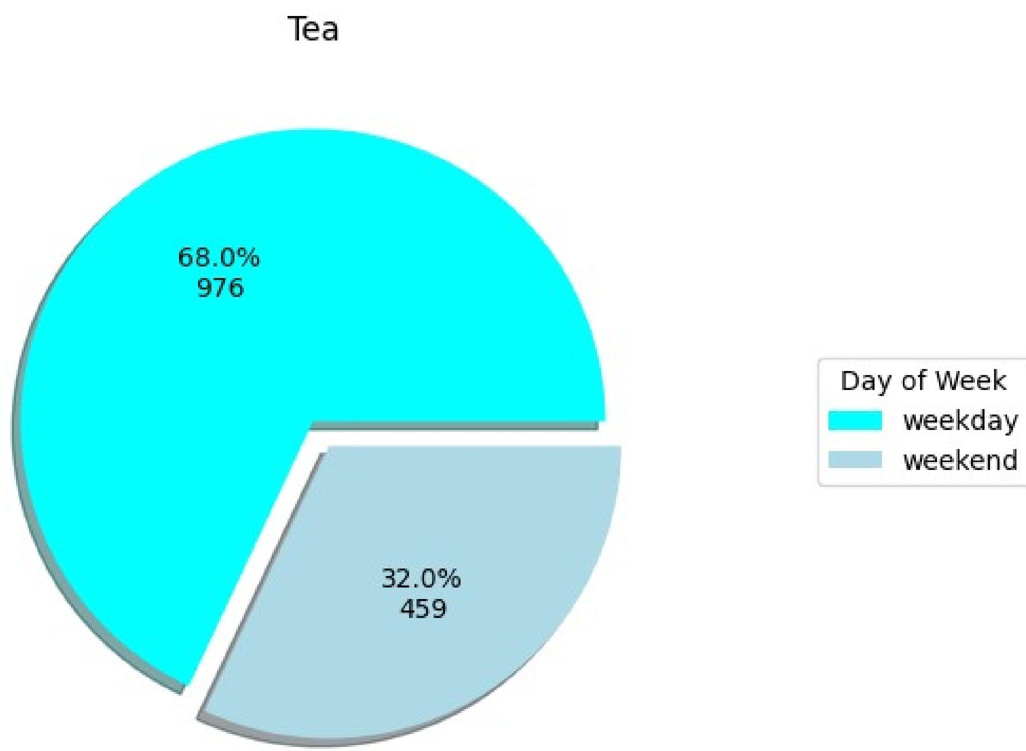
```
size = df['weekday_weekend'].value_counts()
labels = size.index.values
colors = ["cyan", "lightblue"]
explode = [0, 0.1]

plt.figure(figsize=(12,5))
plt.pie(size, labels = labels, colors = colors, explode = explode,
shadow = True, autopct = "%.2f%%")
plt.title('Transaction by week period')
plt.show()
```




```python
def func(pct, allvals):
    absolute = int(np.round(pct/100.*np.sum(allvals)))
    return "{:.1f}%\n{:d}".format(pct, absolute)

size = df[df.Item.isin(popular.head(6).index)]
size = pd.crosstab(size['weekday_weekend'],
                   size['Item'])
# size
labels = size.index.values
colors = ["cyan", "lightblue"]


for e in popular.head(3).index:
    plt.figure(figsize=(12,5))
    dt = size[e]
    explode = [0, 0.1]
    plt.pie(dt, colors = colors, explode = explode, shadow = True,
            autopct=lambda pct: func(pct, dt.values))

    plt.title(e)
    plt.legend(labels = labels, title="Day of Week",
          loc="center right", bbox_to_anchor=(1, 0, 0.5, 1))
    plt.show()
```

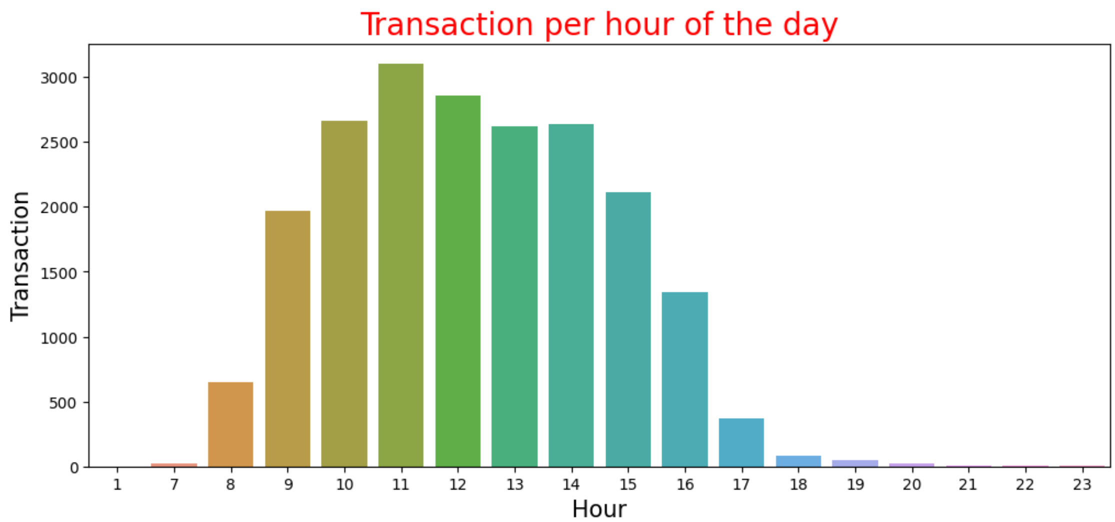

Coffee
64.8%
3543
35.2%
1928
Day of Week
weekday
weekend


Bread
62.9%
2092
37.1%
1233
Day of Week
weekday
weekend



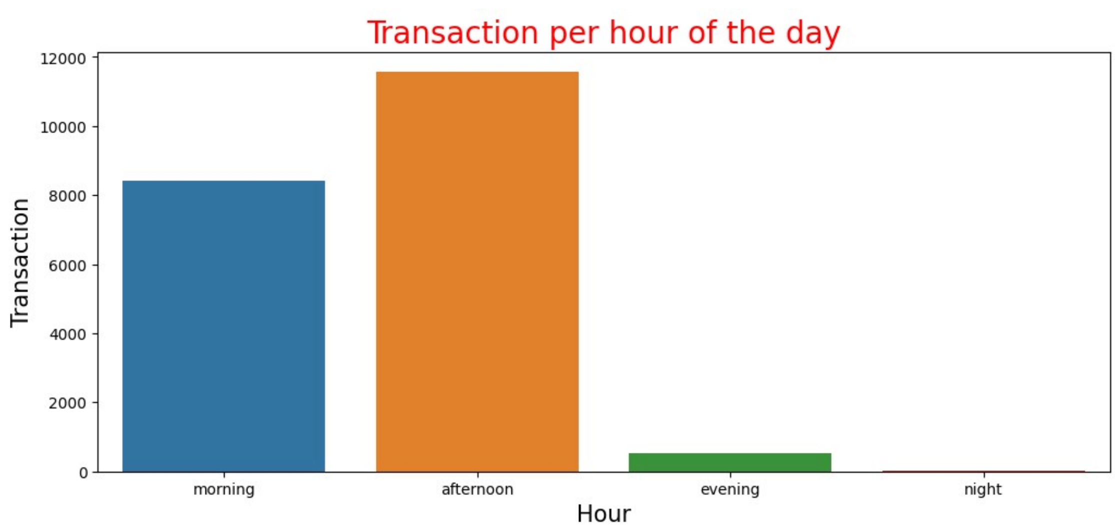
The activity of consumers during the day.

```
coutbyhour=df.groupby('hour')['Transaction'].count().reset_index()
coutbyhour.sort_values('hour',inplace=True)

plt.figure(figsize=(12,5))
sns.barplot(data=coutbyhour, x='hour', y='Transaction')
plt.xlabel('Hour', size = 15)
plt.ylabel('Transaction', size = 15)
plt.title('Transaction per hour of the day', color = 'red', size = 20)
plt.show()
```



The activity of buyers during parts of the day.

```
coutbyweekday=df.groupby('period_day')
['Transaction'].count().reset_index()
coutbyweekday.loc[:,"dayorder"] = [1, 2, 0, 3]
coutbyweekday.sort_values("dayorder",inplace=True)
plt.figure(figsize=(12,5))
sns.barplot(data=coutbyweekday, x='period_day', y='Transaction')
plt.xlabel('Hour', size = 15)
plt.ylabel('Transaction', size = 15)
plt.title('Transaction per hour of the day', color = 'red', size = 20)
plt.show()
```



Association Rules

To create association rules, it's important to establish the connection between purchases. This can be achieved by converting the transaction dataset into a specific table format where the columns represent the types of purchases and the rows represent the transactions. The cells of this table should be boolean values (true or false). There are two commonly used methods to accomplish this.

Using Pivot table

```
transactions = df.groupby(['Transaction', 'Item'])
['Item'].count().reset_index(name ='Count')
transactions

       Transaction            Item  Count
0                1           Bread      1
1                2     Scandinavian      2
2                3         Cookies      1
3                3   Hot chocolate      1
4                3             Jam      1
...            ...             ...    ...
18882         9682    Tacos/Fajita      1
18883         9682             Tea      1
18884         9683          Coffee      1
18885         9683          Pastry      1
18886         9684       Smoothies      1

[18887 rows x 3 columns]

basket = transactions.pivot_table(index='Transaction', columns='Item',
values='Count', aggfunc='sum').fillna(0)
basket

Item         Adjustment  Afternoon with the baker  Alfajores
Argentina Night  \
Transaction

1                   0.0                       0.0        0.0
0.0
2                   0.0                       0.0        0.0
0.0
3                   0.0                       0.0        0.0
0.0
4                   0.0                       0.0        0.0
0.0
5                   0.0                       0.0        0.0
0.0
...                 ...                       ...        ...
...
9680                0.0                       0.0        0.0
0.0
9681                0.0                       0.0        0.0
0.0
9682                0.0                       0.0        0.0
0.0
9683                0.0                       0.0        0.0
```

```
0.0
9684                0.0                        0.0        0.0
0.0

Item          Art Tray  Bacon  Baguette  Bakewell  Bare Popcorn  Basket
...  \
Transaction
...
1                  0.0    0.0       0.0       0.0           0.0     0.0
...
2                  0.0    0.0       0.0       0.0           0.0     0.0
...
3                  0.0    0.0       0.0       0.0           0.0     0.0
...
4                  0.0    0.0       0.0       0.0           0.0     0.0
...
5                  0.0    0.0       0.0       0.0           0.0     0.0
...
...                ...    ...       ...       ...           ...     ...
...
9680               0.0    0.0       0.0       0.0           0.0     0.0
...
9681               0.0    0.0       0.0       0.0           0.0     0.0
...
9682               0.0    0.0       0.0       0.0           0.0     0.0
...
9683               0.0    0.0       0.0       0.0           0.0     0.0
...
9684               0.0    0.0       0.0       0.0           0.0     0.0
...

Item          The BART  The Nomad  Tiffin  Toast  Truffles  Tshirt  \
Transaction
1                  0.0        0.0     0.0    0.0       0.0     0.0
2                  0.0        0.0     0.0    0.0       0.0     0.0
3                  0.0        0.0     0.0    0.0       0.0     0.0
4                  0.0        0.0     0.0    0.0       0.0     0.0
5                  0.0        0.0     0.0    0.0       0.0     0.0
...                ...        ...     ...    ...       ...     ...
9680               0.0        0.0     0.0    0.0       0.0     0.0
9681               0.0        0.0     0.0    0.0       1.0     0.0
9682               0.0        0.0     0.0    0.0       0.0     0.0
9683               0.0        0.0     0.0    0.0       0.0     0.0
9684               0.0        0.0     0.0    0.0       0.0     0.0

Item          Valentine's card  Vegan Feast  Vegan mincepie  Victorian
Sponge
Transaction

1                          0.0          0.0             0.0
```

```
0.0
2                         0.0          0.0             0.0
0.0
3                         0.0          0.0             0.0
0.0
4                         0.0          0.0             0.0
0.0
5                         0.0          0.0             0.0
0.0
...                       ...          ...             ...
...
9680                      0.0          0.0             0.0
0.0
9681                      0.0          0.0             0.0
0.0
9682                      0.0          0.0             0.0
0.0
9683                      0.0          0.0             0.0
0.0
9684                      0.0          0.0             0.0
0.0

[9465 rows x 94 columns]
```

```
def encode_units(x):
    if(x==0):
        return False
    if(x>0):
        return True

basket_sets = basket.applymap(encode_units)
basket_sets
```

```
Item         Adjustment  Afternoon with the baker  Alfajores
Argentina Night  \
Transaction

1                 False                     False      False
False
2                 False                     False      False
False
3                 False                     False      False
False
4                 False                     False      False
False
5                 False                     False      False
False
...                 ...                       ...        ...
...
9680              False                     False      False
```

```
False
9681             False                    False      False
False
9682             False                    False      False
False
9683             False                    False      False
False
9684             False                    False      False
False

Item         Art Tray  Bacon  Baguette  Bakewell  Bare Popcorn  Basket
...  \
Transaction
...
1               False  False     False     False         False   False
...
2               False  False     False     False         False   False
...
3               False  False     False     False         False   False
...
4               False  False     False     False         False   False
...
5               False  False     False     False         False   False
...
...               ...    ...       ...       ...           ...     ...
...
9680            False  False     False     False         False   False
...
9681            False  False     False     False         False   False
...
9682            False  False     False     False         False   False
...
9683            False  False     False     False         False   False
...
9684            False  False     False     False         False   False
...

Item         The BART  The Nomad  Tiffin  Toast  Truffles  Tshirt  \
Transaction
1               False      False   False  False     False   False
2               False      False   False  False     False   False
3               False      False   False  False     False   False
4               False      False   False  False     False   False
5               False      False   False  False     False   False
...               ...        ...     ...    ...       ...     ...
9680            False      False   False  False     False   False
9681            False      False   False  False      True   False
9682            False      False   False  False     False   False
9683            False      False   False  False     False   False
9684            False      False   False  False     False   False
```

```
Item          Valentine's card  Vegan Feast  Vegan mincepie  Victorian
Sponge
Transaction

1                        False        False            False
False
2                        False        False            False
False
3                        False        False            False
False
4                        False        False            False
False
5                        False        False            False
False
...                        ...          ...              ...
...
9680                     False        False            False
False
9681                     False        False            False
False
9682                     False        False            False
False
9683                     False        False            False
False
9684                     False        False            False
False

[9465 rows x 94 columns]
```

Using mlxtend framework

```
transactions=[]
for item in df['Transaction'].unique():
    lst=list(set(df[df['Transaction']==item]['Item']))
    transactions.append(lst)

transactions[0:10]

[['Bread'],
 ['Scandinavian'],
 ['Cookies', 'Jam', 'Hot chocolate'],
 ['Muffin'],
 ['Coffee', 'Pastry', 'Bread'],
 ['Medialuna', 'Pastry', 'Muffin'],
 ['Tea', 'Coffee', 'Medialuna', 'Pastry'],
 ['Bread', 'Pastry'],
 ['Bread', 'Muffin'],
 ['Scandinavian', 'Medialuna']]
```

```
te = TransactionEncoder()
encodedData = te.fit(transactions).transform(transactions)
basket_sets_2 = pd.DataFrame(encodedData, columns=te.columns_)
basket_sets_2
```

```
      Adjustment  Afternoon with the baker  Alfajores  Argentina Night
\
0          False                     False      False            False

1          False                     False      False            False

2          False                     False      False            False

3          False                     False      False            False

4          False                     False      False            False

...          ...                       ...        ...              ...

9460       False                     False      False            False

9461       False                     False      False            False

9462       False                     False      False            False

9463       False                     False      False            False

9464       False                     False      False            False


      Art Tray  Bacon  Baguette  Bakewell  Bare Popcorn
Basket  ...  \
0        False  False     False     False         False   False  ...

1        False  False     False     False         False   False  ...

2        False  False     False     False         False   False  ...

3        False  False     False     False         False   False  ...

4        False  False     False     False         False   False  ...

...        ...    ...       ...       ...           ...     ...  ...

9460     False  False     False     False         False   False  ...

9461     False  False     False     False         False   False  ...
```

```
9462     False  False     False     False         False   False  ...

9463     False  False     False     False         False   False  ...

9464     False  False     False     False         False   False  ...


      The BART  The Nomad  Tiffin  Toast  Truffles  Tshirt
Valentine's card  \
0        False      False   False  False     False   False
False
1        False      False   False  False     False   False
False
2        False      False   False  False     False   False
False
3        False      False   False  False     False   False
False
4        False      False   False  False     False   False
False
...        ...        ...     ...    ...       ...     ...
...
9460     False      False   False  False     False   False
False
9461     False      False   False  False      True   False
False
9462     False      False   False  False     False   False
False
9463     False      False   False  False     False   False
False
9464     False      False   False  False     False   False
False

      Vegan Feast  Vegan mincepie  Victorian Sponge
0           False           False             False
1           False           False             False
2           False           False             False
3           False           False             False
4           False           False             False
...           ...             ...               ...
9460        False           False             False
9461        False           False             False
9462        False           False             False
9463        False           False             False
9464        False           False             False

[9465 rows x 94 columns]
```

```
frequentItems= apriori(basket_sets, use_colnames=True,
min_support=0.01)
frequentItems.sort_values("support", ascending=False)
```

```
     support                 itemsets
6   0.478394                 (Coffee)
2   0.327205                  (Bread)
26  0.142631                    (Tea)
4   0.103856                   (Cake)
34  0.090016          (Bread, Coffee)
..       ...                      ...
11  0.010565    (Hearty & Seasonal)
20  0.010460                  (Salad)
30  0.010354      (Bread, Alfajores)
58  0.010037  (Bread, Cake, Coffee)
60  0.010037    (Tea, Coffee, Cake)

[61 rows x 2 columns]
```

Metrics

```
rules = association_rules(frequentItems, metric="confidence",
min_threshold=0.2)
rules.sort_values('confidence', ascending = False, inplace=True)
rules
```

```
         antecedents consequents  antecedent support  consequent
support  \
24           (Toast)    (Coffee)            0.033597
0.478394
22  (Spanish Brunch)    (Coffee)            0.018172
0.478394
16       (Medialuna)    (Coffee)            0.061807
0.478394
18          (Pastry)    (Coffee)            0.086107
0.478394
1        (Alfajores)    (Coffee)            0.036344
0.478394
15           (Juice)    (Coffee)            0.038563
0.478394
19        (Sandwich)    (Coffee)            0.071844
0.478394
11            (Cake)    (Coffee)            0.103856
0.478394
20           (Scone)    (Coffee)            0.034548
0.478394
13         (Cookies)    (Coffee)            0.054411
0.478394
14    (Hot chocolate)    (Coffee)            0.058320
0.478394
10         (Brownie)    (Coffee)            0.040042
0.478394
17          (Muffin)    (Coffee)            0.038457
0.478394
```

```
21             (Soup)    (Coffee)            0.034443
0.478394
26     (Bread, Cake)    (Coffee)            0.023349
0.478394
30       (Tea, Cake)    (Coffee)            0.023772
0.478394
27   (Bread, Pastry)    (Coffee)            0.029160
0.478394
23              (Tea)    (Coffee)            0.142631
0.478394
8            (Pastry)     (Bread)            0.086107
0.327205
0        (Alfajores)     (Bread)            0.036344
0.327205
4             (Bread)    (Coffee)            0.327205
0.478394
7        (Medialuna)     (Bread)            0.061807
0.327205
2           (Brownie)     (Bread)            0.040042
0.327205
5           (Cookies)     (Bread)            0.054411
0.327205
9          (Sandwich)     (Bread)            0.071844
0.327205
28  (Coffee, Pastry)     (Bread)            0.047544
0.327205
6    (Hot chocolate)     (Bread)            0.058320
0.327205
12             (Cake)       (Tea)            0.103856
0.142631
3              (Cake)     (Bread)            0.103856
0.327205
29     (Tea, Coffee)      (Cake)            0.049868
0.103856
25         (Sandwich)       (Tea)            0.071844
0.142631

     support  confidence      lift  leverage  conviction
24  0.023666    0.704403  1.472431  0.007593    1.764582
22  0.010882    0.598837  1.251766  0.002189    1.300235
16  0.035182    0.569231  1.189878  0.005614    1.210871
18  0.047544    0.552147  1.154168  0.006351    1.164682
1   0.019651    0.540698  1.130235  0.002264    1.135648
15  0.020602    0.534247  1.116750  0.002154    1.119919
19  0.038246    0.532353  1.112792  0.003877    1.115384
11  0.054728    0.526958  1.101515  0.005044    1.102664
20  0.018067    0.522936  1.093107  0.001539    1.093366
13  0.028209    0.518447  1.083723  0.002179    1.083174
14  0.029583    0.507246  1.060311  0.001683    1.058553
10  0.019651    0.490765  1.025860  0.000495    1.024293
```

```
17  0.018806    0.489011  1.022193  0.000408    1.020777
21  0.015848    0.460123  0.961807 -0.000629    0.966156
26  0.010037    0.429864  0.898557 -0.001133    0.914880
30  0.010037    0.422222  0.882582 -0.001335    0.902779
27  0.011199    0.384058  0.802807 -0.002751    0.846843
23  0.049868    0.349630  0.730840 -0.018366    0.802014
8   0.029160    0.338650  1.034977  0.000985    1.017305
0   0.010354    0.284884  0.870657 -0.001538    0.940818
4   0.090016    0.275105  0.575059 -0.066517    0.719561
7   0.016904    0.273504  0.835879 -0.003319    0.926082
2   0.010777    0.269129  0.822508 -0.002326    0.920538
5   0.014474    0.266019  0.813004 -0.003329    0.916638
9   0.017010    0.236765  0.723596 -0.006498    0.881503
28  0.011199    0.235556  0.719901 -0.004357    0.880109
6   0.013418    0.230072  0.703144 -0.005665    0.873841
12  0.023772    0.228891  1.604781  0.008959    1.111865
3   0.023349    0.224822  0.687097 -0.010633    0.867923
29  0.010037    0.201271  1.937977  0.004858    1.121962
25  0.014369    0.200000  1.402222  0.004122    1.071712
```

This data set contains all possible causal relationships.

All rules that have lift>1:

```
rules[rules["lift"]>1].sort_values("support",ascending = False)
```

```
         antecedents consequents  antecedent support  consequent
support  \
11            (Cake)    (Coffee)            0.103856
0.478394
18          (Pastry)    (Coffee)            0.086107
0.478394
19        (Sandwich)    (Coffee)            0.071844
0.478394
16       (Medialuna)    (Coffee)            0.061807
0.478394
14  (Hot chocolate)    (Coffee)            0.058320
0.478394
8           (Pastry)     (Bread)            0.086107
0.327205
13         (Cookies)    (Coffee)            0.054411
0.478394
12            (Cake)       (Tea)            0.103856
0.142631
24           (Toast)    (Coffee)            0.033597
0.478394
15           (Juice)    (Coffee)            0.038563
0.478394
1        (Alfajores)    (Coffee)            0.036344
0.478394
10         (Brownie)    (Coffee)            0.040042
```

```
0.478394
17          (Muffin)    (Coffee)            0.038457
0.478394
20           (Scone)    (Coffee)            0.034548
0.478394
25        (Sandwich)       (Tea)            0.071844
0.142631
22  (Spanish Brunch)    (Coffee)            0.018172
0.478394
29     (Tea, Coffee)      (Cake)            0.049868
0.103856

     support  confidence      lift  leverage  conviction
11  0.054728    0.526958  1.101515  0.005044    1.102664
18  0.047544    0.552147  1.154168  0.006351    1.164682
19  0.038246    0.532353  1.112792  0.003877    1.115384
16  0.035182    0.569231  1.189878  0.005614    1.210871
14  0.029583    0.507246  1.060311  0.001683    1.058553
8   0.029160    0.338650  1.034977  0.000985    1.017305
13  0.028209    0.518447  1.083723  0.002179    1.083174
12  0.023772    0.228891  1.604781  0.008959    1.111865
24  0.023666    0.704403  1.472431  0.007593    1.764582
15  0.020602    0.534247  1.116750  0.002154    1.119919
1   0.019651    0.540698  1.130235  0.002264    1.135648
10  0.019651    0.490765  1.025860  0.000495    1.024293
17  0.018806    0.489011  1.022193  0.000408    1.020777
20  0.018067    0.522936  1.093107  0.001539    1.093366
25  0.014369    0.200000  1.402222  0.004122    1.071712
22  0.010882    0.598837  1.251766  0.002189    1.300235
29  0.010037    0.201271  1.937977  0.004858    1.121962
```

A situation where a customer buys a Cake. Let's predict what else they can buy:

```
rules[rules['antecedents'] == frozenset({'Cake'})]
```

```
   antecedents consequents  antecedent support  consequent support  
support  \
11      (Cake)    (Coffee)            0.103856            0.478394   
0.054728   
12      (Cake)       (Tea)            0.103856            0.142631   
0.023772   
3       (Cake)     (Bread)            0.103856            0.327205   
0.023349   

    confidence      lift  leverage  conviction  
11    0.526958  1.101515  0.005044    1.102664  
12    0.228891  1.604781  0.008959    1.111865  
3     0.224822  0.687097 -0.010633    0.867923  
```

1. Coffee - 47%
2. Tea - 14%

3. Bread - 3%

This can be a recommendation of which product should be placed closer to or farther from the cake on the shelves. Depending on the strategy of the supermarket.

Products which are bought together the most frequently.

```
frequentItems["antecedent_len"] =
frequentItems["itemsets"].apply(lambda x: len(x))
frequentItems[frequentItems["antecedent_len"]>1].sort_values(by=["ante
cedent_len","support"], ascending=False)
```

```
     support                   itemsets  antecedent_len
59  0.011199   (Bread, Pastry, Coffee)               3
58  0.010037     (Bread, Cake, Coffee)               3
60  0.010037       (Tea, Coffee, Cake)               3
34  0.090016           (Bread, Coffee)               2
42  0.054728            (Coffee, Cake)               2
55  0.049868             (Tea, Coffee)               2
50  0.047544          (Coffee, Pastry)               2
51  0.038246        (Coffee, Sandwich)               2
48  0.035182       (Coffee, Medialuna)               2
46  0.029583   (Coffee, Hot chocolate)               2
38  0.029160           (Bread, Pastry)               2
45  0.028209         (Cookies, Coffee)               2
40  0.028104              (Tea, Bread)               2
44  0.023772               (Tea, Cake)               2
56  0.023666           (Coffee, Toast)               2
33  0.023349             (Bread, Cake)               2
47  0.020602           (Coffee, Juice)               2
31  0.019651       (Coffee, Alfajores)               2
41  0.019651         (Coffee, Brownie)               2
49  0.018806          (Coffee, Muffin)               2
52  0.018067           (Coffee, Scone)               2
39  0.017010         (Bread, Sandwich)               2
37  0.016904        (Bread, Medialuna)               2
53  0.015848            (Coffee, Soup)               2
35  0.014474          (Cookies, Bread)               2
57  0.014369           (Tea, Sandwich)               2
36  0.013418    (Bread, Hot chocolate)               2
43  0.011410     (Cake, Hot chocolate)               2
54  0.010882  (Coffee, Spanish Brunch)               2
32  0.010777          (Bread, Brownie)               2
30  0.010354        (Bread, Alfajores)               2
```

```
index_names = rules['consequents'] == frozenset({'Coffee'})
refinedRules = rules[~index_names].sort_values('lift',
ascending=False)
refinedRules.drop(['leverage','conviction'], axis=1, inplace=True)
refinedRules = refinedRules.reset_index()
refinedRules
```

```
    index       antecedents consequents  antecedent support  \
0      29     (Tea, Coffee)      (Cake)            0.049868
1      12            (Cake)       (Tea)            0.103856
2      25        (Sandwich)       (Tea)            0.071844
3       8          (Pastry)     (Bread)            0.086107
4       0       (Alfajores)     (Bread)            0.036344
5       7       (Medialuna)     (Bread)            0.061807
6       2         (Brownie)     (Bread)            0.040042
7       5         (Cookies)     (Bread)            0.054411
8       9        (Sandwich)     (Bread)            0.071844
9      28  (Coffee, Pastry)     (Bread)            0.047544
10      6   (Hot chocolate)     (Bread)            0.058320
11      3            (Cake)     (Bread)            0.103856

    consequent support   support  confidence      lift
0             0.103856  0.010037    0.201271  1.937977
1             0.142631  0.023772    0.228891  1.604781
2             0.142631  0.014369    0.200000  1.402222
3             0.327205  0.029160    0.338650  1.034977
4             0.327205  0.010354    0.284884  0.870657
5             0.327205  0.016904    0.273504  0.835879
6             0.327205  0.010777    0.269129  0.822508
7             0.327205  0.014474    0.266019  0.813004
8             0.327205  0.017010    0.236765  0.723596
9             0.327205  0.011199    0.235556  0.719901
10            0.327205  0.013418    0.230072  0.703144
11            0.327205  0.023349    0.224822  0.687097
```

Visualization of Association Rules

```
Basket_Network = Network(height="1000px", width="1000px",
directed=True, notebook=True)
```

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues
displaying graphics on chrome/safari. Use cdn_resources='in_line' or
cdn_resources='remote' if you have issues viewing graphics in a
notebook.
```

```
Basket_Network.force_atlas_2based()
Basket_Network.barnes_hut()
Basket_Network.hrepulsion()
Basket_Network.repulsion()

Basket_Network_Data_zip=zip(rules["antecedents"],
                            rules["consequents"],
                            rules["antecedent support"],
                            rules["consequent support"],
                            rules["confidence"])

for i in Basket_Network_Data_zip:
```

```
FromItem=str(i[0]).replace("frozenset({'","").replace("'})","").replac
e("', '",",")

ToItem=str(i[1]).replace("frozenset({'","").replace("'})","").replace(
"', '",",")
    FromWeight=i[2]
    ToWeight=i[3]
    EdgeWeight=i[4]

    Basket_Network.add_node(n_id=FromItem, shape="dot",
value=FromWeight,
                            title=FromItem + "<br>Support: " +
str(FromWeight))
    Basket_Network.add_node(n_id=ToItem, shape="dot", value=ToWeight,
                           title=ToItem + "<br>Support: " +
str(ToWeight))
    Basket_Network.add_edge(source=FromItem, to=ToItem,
value=EdgeWeight, arrowStrikethrough=False,
                            title=FromItem + " --> " + ToItem +
"<br>Confidence:" + str(EdgeWeight))

Basket_Network.set_edge_smooth(smooth_type="continuous")
Basket_Network.toggle_hide_edges_on_drag(True)

Basket_Network.save_graph("Basket_Network1.html")
Basket_Network.show("Basket_Network1.html")
```

Basket_Network1.html

<IPython.lib.display.IFrame at 0x1dd76abf940>

```
Basket_Network2 = Network(height="1000px", width="1000px",
directed=True, notebook=True)
Basket_Network2.repulsion()
Basket_Network_Data2_zip=zip(refinedRules["antecedents"],
                            refinedRules["consequents"],
                            refinedRules["antecedent support"],
                            refinedRules["consequent support"],
                            refinedRules["confidence"])


for i in Basket_Network_Data2_zip:

FromItem=str(i[0]).replace("frozenset({'","").replace("'})","").replac
e("', '",",")

ToItem=str(i[1]).replace("frozenset({'","").replace("'})","").replace(
"', '",",")
    FromWeight=i[2]
    ToWeight=i[3]
    EdgeWeight=i[4]
```

```
    Basket_Network2.add_node(n_id=FromItem, shape="dot",
value=FromWeight,
                            title=FromItem + "<br>Support: " +
str(FromWeight))
    Basket_Network2.add_node(n_id=ToItem, shape="dot", value=ToWeight,
                           title=ToItem + "<br>Support: " +
str(ToWeight))
    Basket_Network2.add_edge(source=FromItem, to=ToItem,
value=EdgeWeight, arrowStrikethrough=False,
                            title=FromItem + " --> " + ToItem +
"<br>Confidence:" + str(EdgeWeight))

Basket_Network2.set_edge_smooth(smooth_type="continuous")
Basket_Network2.toggle_hide_edges_on_drag(True)
Basket_Network2.save_graph("Basket_Network2.html")
Basket_Network2.show("Basket_Network2.html")
```

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues
displaying graphics on chrome/safari. Use cdn_resources='in_line' or
cdn_resources='remote' if you have issues viewing graphics in a
notebook.
Basket_Network2.html
```

```
<IPython.lib.display.IFrame at 0x1dd74c4bd00>
```