```
# ! pip install pandas
# ! pip install matplotlib
! pip install mlxtend
```

Show hidden output

```
! pip install pyvis
```

```
Collecting pyvis
  Downloading pyvis-0.3.2-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: ipython>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from pyvis) (7.34.0)
Requirement already satisfied: jinja2>=2.9.6 in /usr/local/lib/python3.11/dist-packages (from pyvis) (3.1.5)
Requirement already satisfied: jsonpickle>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from pyvis) (4.0.2)
Requirement already satisfied: networkx>=1.11 in /usr/local/lib/python3.11/dist-packages (from pyvis) (3.4.2)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (75.1.0)
Collecting jedi>=0.16 (from ipython>=5.3.0->pyvis)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (2.18.0)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (0.2.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=5.3.0->pyvis) (4.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2>=2.9.6->pyvis) (3.0.2)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=5.3.0->pyvis) (
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=5.3.0->pyvis) (0.7
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ip
Downloading pyvis-0.3.2-py3-none-any.whl (756 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 756.0/756.0 kB 18.8 MB/s eta 0:00:00
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 34.8 MB/s eta 0:00:00
Installing collected packages: jedi, pyvis
Successfully installed jedi-0.19.2 pyvis-0.3.2
```

```
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 npro
```

```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-GPXX0VY2EN/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 × 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
```

```
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
<ipython-input-8-7ae6461dde0f>:1: UserWarning: Parsing dates in %d-%m-%Y %H:%M format when dayfirst=False (the default) was specified. P
  df['date_time']=pd.to_datetime(df['date_time'])
```

```
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','Sept
```

```
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	weekday_weekend	time	hour	month	month name	day	weekday	weekday name
0	1	Bread	2016-10-30 09:58:00	morning	weekend	09:58:00	9	10	October	30	6	Sunday
1	2	Scandinavian	2016-10-30 10:05:00	morning	weekend	10:05:00	10	10	October	30	6	Sunday
2	2	Scandinavian	2016-10-30 10:05:00	morning	weekend	10:05:00	10	10	October	30	6	Sunday
3	3	Hot chocolate	2016-10-30 10:07:00	morning	weekend	10:07:00	10	10	October	30	6	Sunday
4	3	Jam	2016-10-30 10:07:00	morning	weekend	10:07:00	10	10	October	30	6	Sunday
...	...	...	...	...	...	...	...	...	...	...	...	...
20502	9682	Coffee	2017-04-09 14:32:00	afternoon	weekend	14:32:00	14	4	April	9	6	Sunday
20503	9682	Tea	2017-04-09 14:32:00	afternoon	weekend	14:32:00	14	4	April	9	6	Sunday

```
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  int32
 7   month            20507 non-null  int32
 8   month name       20507 non-null  object
 9   day              20507 non-null  int32
 10  weekday          20507 non-null  int32
 11  weekday name     20507 non-null  object
dtypes: datetime64[ns](1), int32(4), int64(1), object(6)
memory usage: 1.6+ MB
```

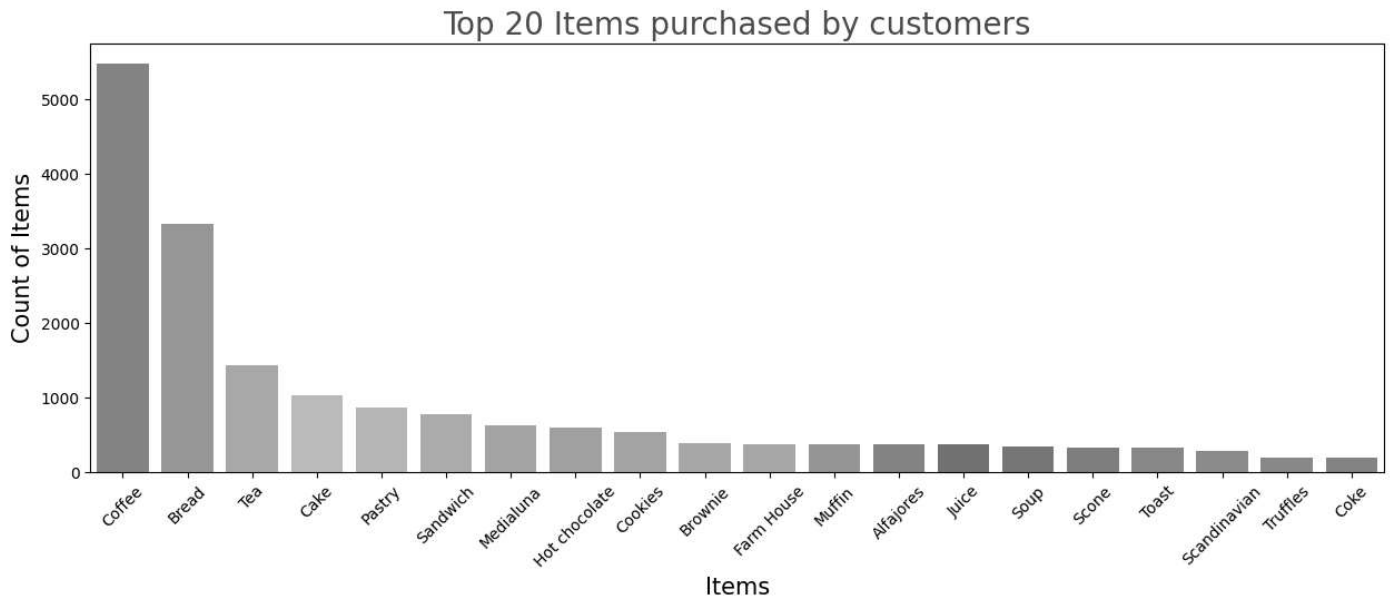
```
popular = df['Item'].value_counts()
(df['Item'].value_counts(normalize=True)*100).head(20)
```

Item	proportion
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

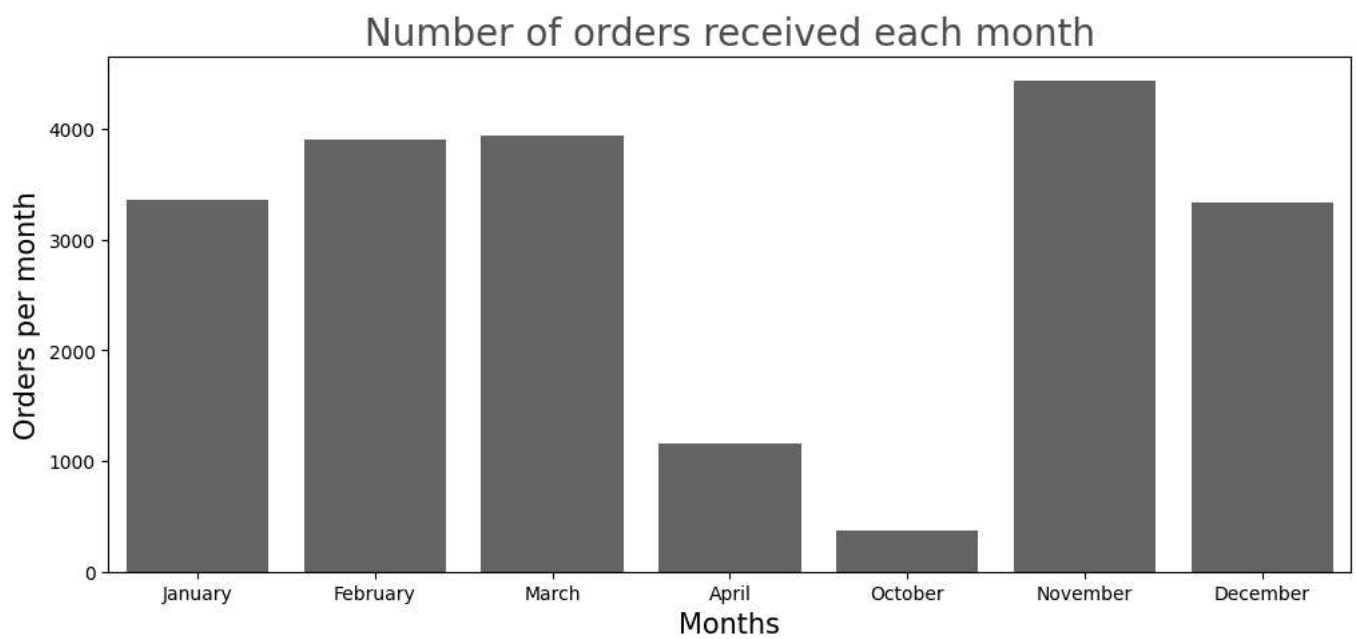
dtype: float64

```
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()
```

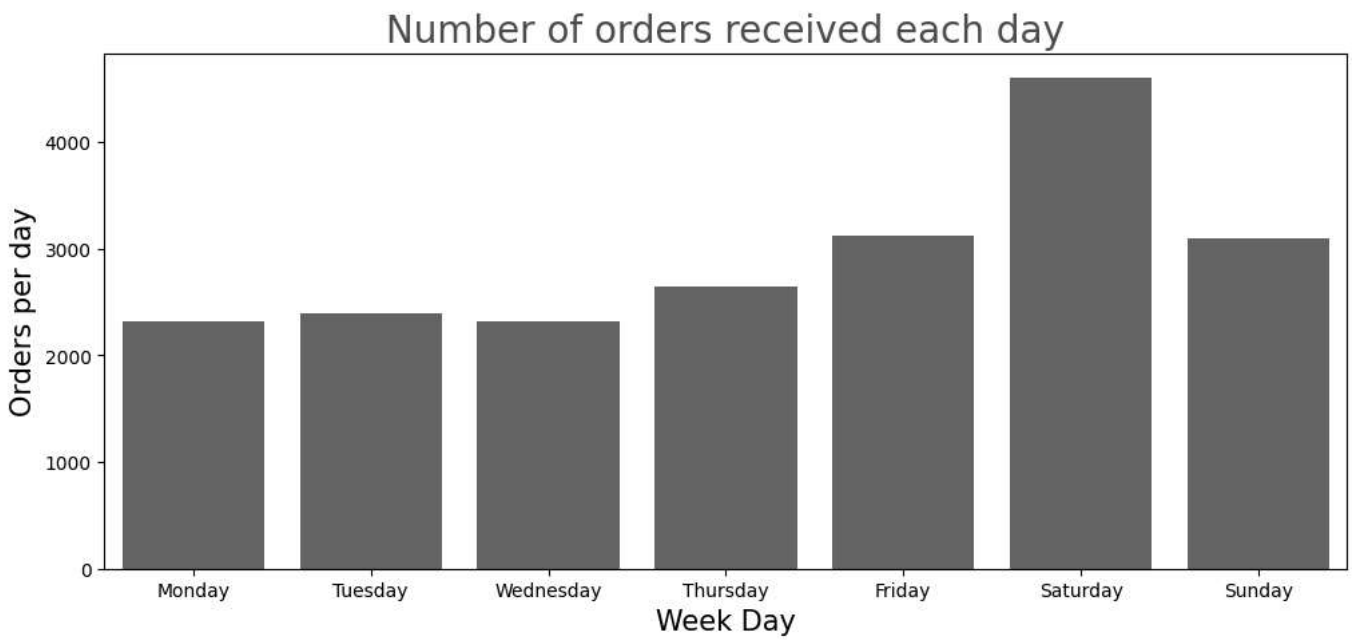
```
<ipython-input-15-62353c6d8660>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legenc

  sns.barplot(x = popular.head(20).index, y = popular.head(20).values, palette = 'hls')
```



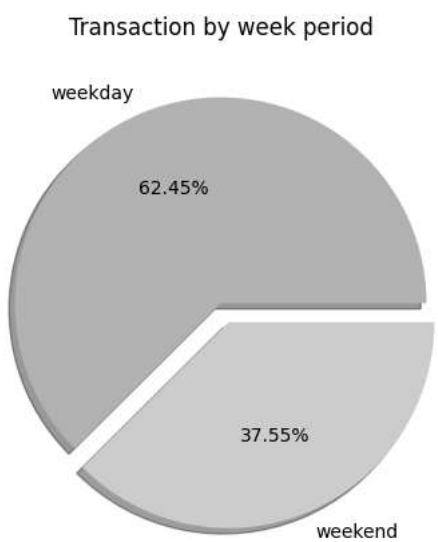

```
monthTran = df.groupby(['month','month name'])['Transaction'].count().reset_index()
plt.figure(figsize=(12,5))
sns.barplot(data = monthTran[['month name', 'Transaction']], x = "month name", y = "Transaction")
plt.xlabel('Months', size = 15)
plt.ylabel('Orders per month', size = 15)
plt.title('Number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



```
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()
```

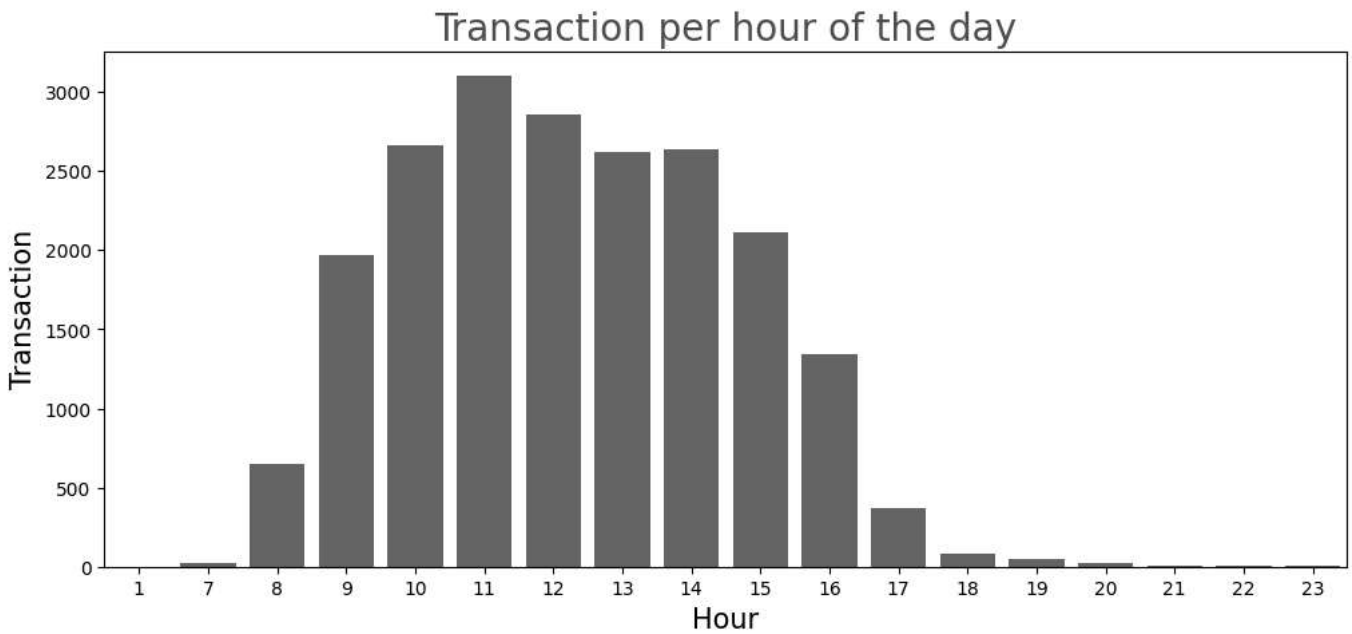



```
 size = df['weekday_weekend'].value_counts()
 labels = size.index.values
colors = ["cyan", "lightblue"]
explode = [0, 0.1]

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()
```

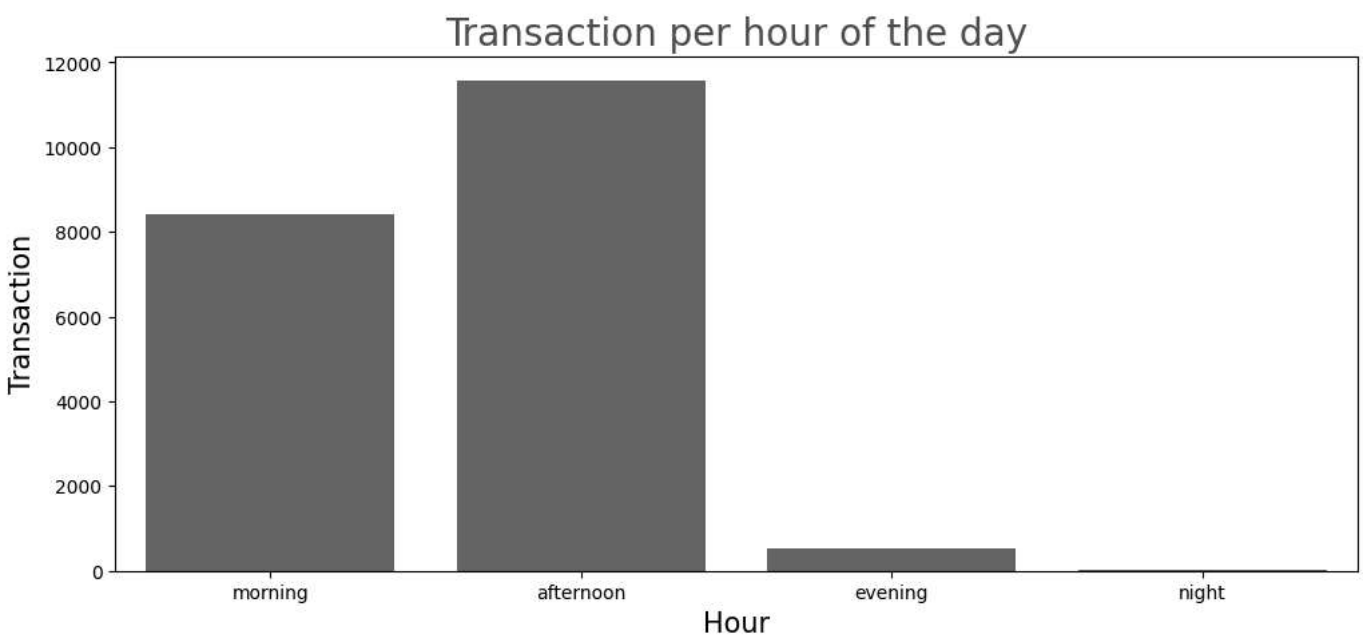


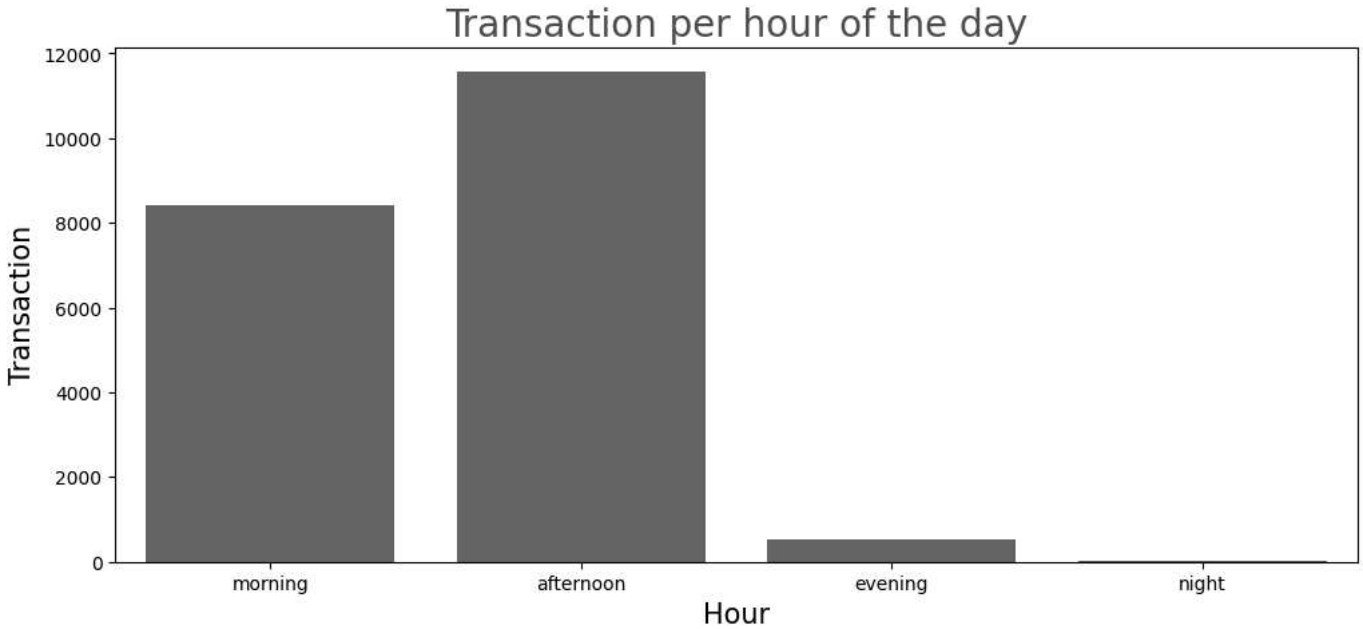

```
coutbyhour=df.groupby('hour')['Transaction'].count().reset_index()
coutbyhour.sort_values('hour',inplace=True)

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()
```




```
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()
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()
```






```
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 × 3 columns

```
# Assuming 'transactions' DataFrame from your previous code holds transaction data
# Create a pivot table to represent the basket
basket = transactions.pivot_table(index='Transaction',
                                  columns='Item',
                                  values='Count',
                                  aggfunc='sum',
                                  fill_value=0)


def encode_units(x):
    # Indent the code within the function
    if(x==0):
        return False
    if(x>0):
        return True

# Now you can use the 'basket' variable, which is defined outside the function
# For example, you can apply the 'encode_units' function to the 'basket' DataFrame
basket_encoded = basket.applymap(encode_units)
```

```
<ipython-input-25-77fe47ff243f>:19: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  basket_encoded = basket.applymap(encode_units)
```

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

transactions[0:10]
```

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

```
# Cell ipython-input-51-c9056906959f
# This line assigns a method object to transactions
# transactions = df.groupby(['Transaction', 'Item'])['Item'].count().reset_index

# Instead, keep the list of transactions from cell ipython-input-46-56c65ffc2649
# transactions should be the list of lists representing the items in each transaction
# Example: transactions = [['Bread']]
```

```
# Use the list of transactions created earlier in the code in cell ipython-input-46-56c65ffc2649
transactions = []
for item in df['Transaction'].unique():
    lst = list(set(df[df['Transaction'] == item]['Item']))
```

```
    transactions.append(lst)
# Now, 'transactions' is the list of lists needed for TransactionEncoder

# Continue with the encoding process
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	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Tru
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9460	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9461	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9462	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9463	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9464	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	

9465 rows × 94 columns

```
# Use the original transactions DataFrame from cell ipython-input-40-f57728cee10d
transactions = df.groupby(['Transaction', 'Item'])['Item'].count().reset_index
```

```
# Use the list of transactions created earlier in the code in cell ipython-input-46-56c65ffc2649
transactions = []
for item in df['Transaction'].unique():
    lst = list(set(df[df['Transaction'] == item]['Item']))
    transactions.append(lst)
# Now, 'transactions' is the list of lists needed for TransactionEncoder

# Continue with the encoding process
te = TransactionEncoder()
encodedData = te.fit(transactions).transform(transactions)
basket_sets = pd.DataFrame(encodedData, columns=te.columns_) # Assign to basket_sets instead of basket_sets_2
basket_sets
```

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Tru
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9460	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9461	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9462	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9463	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
9464	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	

9465 rows × 94 columns

```
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 npro
# ... (rest of your code) ...

# Use the list of transactions created earlier in the code in cell ipython-input-46-56c65ffc2649
transactions = []
for item in df['Transaction'].unique():
    lst = list(set(df[df['Transaction'] == item]['Item']))
    transactions.append(lst)
# Now, 'transactions' is the list of lists needed for TransactionEncoder

# Continue with the encoding process
te = TransactionEncoder()
encodedData = te.fit(transactions).transform(transactions)
basket_sets = pd.DataFrame(encodedData, columns=te.columns_) # Assign to basket_sets instead of basket_sets_2


# Calculate frequent itemsets using apriori BEFORE association_rules
frequentItems = apriori(basket_sets, min_support=0.01, use_colnames=True)  # Adjust min_support as needed


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	support	confidence	lift	representativity	leverage	conviction	zhangs_metr
24	(Toast)	(Coffee)	0.033597	0.478394	0.023666	0.704403	1.472431	1.0	0.007593	1.764582	0.3320
22	(Spanish Brunch)	(Coffee)	0.018172	0.478394	0.010882	0.598837	1.251766	1.0	0.002189	1.300235	0.2048
16	(Medialuna)	(Coffee)	0.061807	0.478394	0.035182	0.569231	1.189878	1.0	0.005614	1.210871	0.1700
18	(Pastry)	(Coffee)	0.086107	0.478394	0.047544	0.552147	1.154168	1.0	0.006351	1.164682	0.1461
1	(Alfajores)	(Coffee)	0.036344	0.478394	0.019651	0.540698	1.130235	1.0	0.002264	1.135648	0.1195
15	(Juice)	(Coffee)	0.038563	0.478394	0.020602	0.534247	1.116750	1.0	0.002154	1.119919	0.1087
19	(Sandwich)	(Coffee)	0.071844	0.478394	0.038246	0.532353	1.112792	1.0	0.003877	1.115384	0.1092
11	(Cake)	(Coffee)	0.103856	0.478394	0.054728	0.526958	1.101515	1.0	0.005044	1.102664	0.1028
20	(Scone)	(Coffee)	0.034548	0.478394	0.018067	0.522936	1.093107	1.0	0.001539	1.093366	0.0882
13	(Cookies)	(Coffee)	0.054411	0.478394	0.028209	0.518447	1.083723	1.0	0.002179	1.083174	0.0817
14	(Hot chocolate)	(Coffee)	0.058320	0.478394	0.029583	0.507246	1.060311	1.0	0.001683	1.058553	0.0604
10	(Brownie)	(Coffee)	0.040042	0.478394	0.019651	0.490765	1.025860	1.0	0.000495	1.024293	0.0262
17	(Muffin)	(Coffee)	0.038457	0.478394	0.018806	0.489011	1.022193	1.0	0.000408	1.020777	0.0225
21	(Soup)	(Coffee)	0.034443	0.478394	0.015848	0.460123	0.961807	1.0	-0.000629	0.966156	-0.0395
26	(Bread, Cake)	(Coffee)	0.023349	0.478394	0.010037	0.429864	0.898557	1.0	-0.001133	0.914880	-0.1036
29	(Cake, Tea)	(Coffee)	0.023772	0.478394	0.010037	0.422222	0.882582	1.0	-0.001335	0.902779	-0.1199
27	(Bread, Pastry)	(Coffee)	0.029160	0.478394	0.011199	0.384058	0.802807	1.0	-0.002751	0.846843	-0.2019
23	(Tea)	(Coffee)	0.142631	0.478394	0.049868	0.349630	0.730840	1.0	-0.018366	0.802014	-0.3004
8	(Pastry)	(Bread)	0.086107	0.327205	0.029160	0.338650	1.034977	1.0	0.000985	1.017305	0.0369
0	(Alfajores)	(Bread)	0.036344	0.327205	0.010354	0.284884	0.870657	1.0	-0.001538	0.940818	-0.1335
4	(Bread)	(Coffee)	0.327205	0.478394	0.090016	0.275105	0.575059	1.0	-0.066517	0.719561	-0.5234
7	(Medialuna)	(Bread)	0.061807	0.327205	0.016904	0.273504	0.835879	1.0	-0.003319	0.926082	-0.1730
2	(Brownie)	(Bread)	0.040042	0.327205	0.010777	0.269129	0.822508	1.0	-0.002326	0.920538	-0.1835
5	(Cookies)	(Bread)	0.054411	0.327205	0.014474	0.266019	0.813004	1.0	-0.003329	0.916638	-0.1956
9	(Sandwich)	(Bread)	0.071844	0.327205	0.017010	0.236765	0.723596	1.0	-0.006498	0.881503	-0.2915
28	(Coffee, Pastry)	(Bread)	0.047544	0.327205	0.011199	0.235556	0.719901	1.0	-0.004357	0.880109	-0.2900
6	(Hot chocolate)	(Bread)	0.058320	0.327205	0.013418	0.230072	0.703144	1.0	-0.005665	0.873841	-0.3095
12	(Cake)	(Tea)	0.103856	0.142631	0.023772	0.228891	1.604781	1.0	0.008959	1.111865	0.4205
3	(Cake)	(Bread)	0.103856	0.327205	0.023349	0.224822	0.687097	1.0	-0.010633	0.867923	-0.3369
30	(Tea, Coffee)	(Cake)	0.049868	0.103856	0.010037	0.201271	1.937977	1.0	0.004858	1.121962	0.5094
25	(Sandwich)	(Tea)	0.071844	0.142631	0.014369	0.200000	1.402222	1.0	0.004122	1.071712	0.3090

```
# Cell ipython-input-65-017c11814289
# Ensure that the cell where 'rules' is calculated (ipython-input-66-017c11814289) has been run before this cell.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori
```

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

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metr
11	(Cake)	(Coffee)	0.103856	0.478394	0.054728	0.526958	1.101515	1.0	0.005044	1.102664	0.1028
12	(Cake)	(Tea)	0.103856	0.142631	0.023772	0.228891	1.604781	1.0	0.008959	1.111865	0.4205
3	(Cake)	(Bread)	0.103856	0.327205	0.023349	0.224822	0.687097	1.0	-0.010633	0.867923	-0.3369

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

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

```
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
```

```
# 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("'})","").replace("',  '",",")
    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")
```

Show hidden output

```
Basket_Network2 = Network(height="1000px", width="1000px", directed=True, notebook=True)
Basket_Network2.repulsion()
Basket_Network_Data2_zip = [] # Replace ##YOUR CODE GOES HERE## with an empty list or appropriate initialization


for i in Basket_Network_Data2_zip:
    FromItem=str(i[0]).replace("frozenset({'","").replace("'})","").replace("',  '",",")
    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")
for i in Basket_Network_Data2_zip:
    FromItem=str(i[0]).replace("frozenset({'","").replace("'})","").replace("',  '",",")
    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
Basket_Network2.html
Basket_Network2.html
```