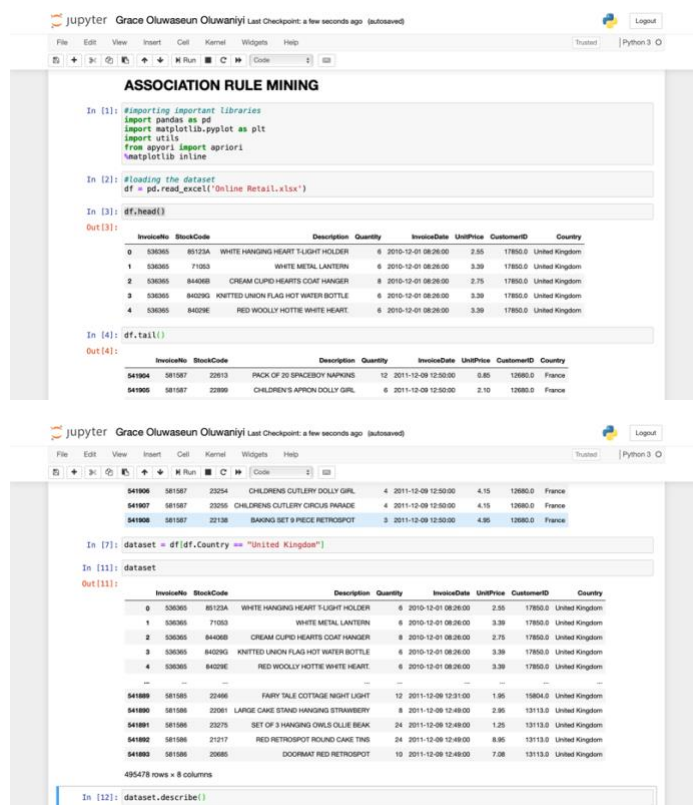


## Datasets

The dataset selected for this study is the online retail dataset on the UCI machine learning repository. The source for this dataset is “Dr Daqing Chen, Director: Public Analytics group. School of Engineering, London South Bank University”.

## Explanation and preparation of datasets

The dataset is a transactional dataset that contains all the transactions that took place in a UK based online retail store. The transactions took place between 01/12/2010 and 09/12/2011. It contains 541910 rows and 8 columns. The rows are for the header and the 541909 purchases that happened within this timeframe. Some purchases were done on the same day and by the same person and as such would have same invoice number and customer ID. The column names are InvoiceNo (Invoice number), StockCode (item code), Description (item name), Quantity (the quantity of the item purchased at the time), InvoiceDate (Date and time of purchase), UnitPrice (price of item per unit), CustomerID (generated customer id), and Country (country where the purchase was made). For this study, we will be selecting the purchases that happened from the United Kingdom because majority of the purchases came from here. The new number of purchases that we would be using to form our rules is 495478.



```
jupyter Grace Oluwaseun Oluwaniyi Last Checkpoint: a few seconds ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: #Importing important libraries
import pandas as pd
import matplotlib.pyplot as plt
import utils
from apyori import apriori
matplotlib inline

In [2]: #Loading the dataset
df = pd.read_excel('Online Retail.xlsx')

In [3]: df.head()
Out[3]:
InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country
0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6 2010-12-01 08:26:00 2.35 17850.0 United Kingdom
1 536365 71053 WHITE METAL LANTERN 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
2 536365 844068 CREAM CUPID HEARTS COAT HANGER 8 2010-12-01 08:26:00 2.75 17850.0 United Kingdom
3 536365 842093 KNITTED UNION FLAG HOT WATER BOTTLE 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
4 536365 842094 RED WOOLLY HOTTIE WHITE HEART 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom

In [4]: df.tail()
Out[4]:
InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country
541904 581587 22813 PACK OF 25 SPACEBOY NAPKINS 12 2011-12-09 12:50:00 6.85 12680.0 France
541905 581587 22899 CHILDREN'S APRON DOLLY GIRL 6 2011-12-09 12:50:00 2.10 12680.0 France

In [5]: df[df['Country'] == 'United Kingdom'].head()
Out[5]:
InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country
0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6 2010-12-01 08:26:00 2.35 17850.0 United Kingdom
1 536365 71053 WHITE METAL LANTERN 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
2 536365 844068 CREAM CUPID HEARTS COAT HANGER 8 2010-12-01 08:26:00 2.75 17850.0 United Kingdom
3 536365 842093 KNITTED UNION FLAG HOT WATER BOTTLE 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
4 536365 842094 RED WOOLLY HOTTIE WHITE HEART 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom

In [6]: df[df['Country'] == 'United Kingdom'].tail()
Out[6]:
InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country
541889 581585 22466 FAIRY TALE COTTAGE NIGHT LIGHT 12 2011-12-09 12:51:00 1.95 13804.0 United Kingdom
541890 581586 22081 LARGE CAKE STAND HANGING STRAWBERRY 8 2011-12-09 12:49:00 2.95 13113.0 United Kingdom
541891 581586 23275 SET OF 3 HANGING OWLS CULLE BREAK 24 2011-12-09 12:49:00 1.25 13113.0 United Kingdom
541892 581586 21217 RED RETROSPOT ROUND CAKE TINS 24 2011-12-09 12:49:00 8.95 13113.0 United Kingdom
541893 581586 20685 DOORMAT RED RETROSPOT 10 2011-12-09 12:49:00 7.08 13113.0 United Kingdom

In [7]: dataset = df[df.Country == "United Kingdom"]

In [11]: dataset
Out[11]:
InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID Country
0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6 2010-12-01 08:26:00 2.35 17850.0 United Kingdom
1 536365 71053 WHITE METAL LANTERN 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
2 536365 844068 CREAM CUPID HEARTS COAT HANGER 8 2010-12-01 08:26:00 2.75 17850.0 United Kingdom
3 536365 842093 KNITTED UNION FLAG HOT WATER BOTTLE 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
4 536365 842094 RED WOOLLY HOTTIE WHITE HEART 6 2010-12-01 08:26:00 3.39 17850.0 United Kingdom
... ..
541889 581585 22466 FAIRY TALE COTTAGE NIGHT LIGHT 12 2011-12-09 12:51:00 1.95 13804.0 United Kingdom
541890 581586 22081 LARGE CAKE STAND HANGING STRAWBERRY 8 2011-12-09 12:49:00 2.95 13113.0 United Kingdom
541891 581586 23275 SET OF 3 HANGING OWLS CULLE BREAK 24 2011-12-09 12:49:00 1.25 13113.0 United Kingdom
541892 581586 21217 RED RETROSPOT ROUND CAKE TINS 24 2011-12-09 12:49:00 8.95 13113.0 United Kingdom
541893 581586 20685 DOORMAT RED RETROSPOT 10 2011-12-09 12:49:00 7.08 13113.0 United Kingdom

495478 rows x 8 columns

In [12]: dataset.describe()
```

Using the describe () feature on our dataset, we could see that we had some missing values and some negative values in the Quantity column. We had 133600 missing values from the CustomerID column and 1454 missing values from the description column. As this data is sensitive in the sense that we need to know what customer got what, we

couldn't deal with the missing values by replacing it with the mean, mode or median. We had to drop the purchases that didn't have a customer ID or/and a description.

```
In [12]: dataset.describe()
```

Out[12]:

	Quantity	UnitPrice	CustomerID
count	495478.000000	495478.000000	361878.000000
mean	8.605486	4.532422	15547.871368
std	227.588756	99.315438	1594.402590
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	14194.000000
50%	3.000000	2.100000	15514.000000
75%	10.000000	4.130000	16931.000000
max	80995.000000	38970.000000	18287.000000

After another look at the dataset, we noticed that some invoice numbers had a c before them to signify that the order was cancelled and that is the reason for the minus for some values in quantity column. We can't use such transactions to form our rules, so they were dropped as well.

```
: dataset.describe()
```

:

	InvoiceNo	Quantity	UnitPrice	CustomerID
count	354345.000000	354345.000000	354345.000000	354345.000000
mean	560672.612341	12.048913	2.963793	15552.436219
std	13168.988616	190.428127	17.862067	1594.546025
min	536365.000000	1.000000	0.000000	12346.000000
25%	549241.000000	2.000000	1.250000	14194.000000
50%	561900.000000	4.000000	1.950000	15522.000000
75%	572295.000000	12.000000	3.750000	16931.000000
max	581586.000000	80995.000000	8142.750000	18287.000000

After dropping based transactions on the discussed conditions above, the final number of transactions used for obtaining the association rules is 354345 as seen in the image above.

## Implementation in Python

Now that our dataset is clean and ready, we will be generating the association rules. We will first look for the most frequent items and then we will generate the rules using one of the frequent pattern algorithms which is the apriori algorithm. Before we begin, we'll see how the apriori algorithm functions.

### Apriori Algorithm

The apriori algorithm is used in the mining of frequent items and generation of association rules over relational datasets. It was proposed by R Agarwal and R Srikant in

1994 (Software Testing Help, 2022). It works with datasets that are transactional. Each transaction forms an itemset. The algorithm calculates the probability that an item will be present in a frequent itemset. If the probability of an item is less than the predefined minimum support, then the said item is not frequent.

The algorithm basically works by identifying the frequent items, then it calculates its probabilities to see if it is above or equal to the minimum support (if it is not, pruning happens), then joins frequent items to form sets of 2, then calculates the probability to see if it is equal to or greater than the minimum support. Then it creates an itemset with 3 items and does the test again. This process continues until we get a group of items with size equal to a predefined size or when the most frequent itemset is built (Java T Point, s.d.). The probability of the group of items must always be greater than or equal to the minimum support. Support, Confidence, and Lift are the three components of the algorithm. We defined these metrics in the introduction.

### Applying Apriori Algorithm on our dataset

To do this, we will be using the apriori class that we imported from the apyori library. When defining the apriori class, we will need to give values to the min\_support and min\_confidence parameters. This will return only rules whose support and confidence is greater than the values given for both parameters.

From the image below, we can see the most frequent items on the 'description' column that were purchased within the time frame. White hanging heart T-light holder topped the list with 1940 purchases.

```
In [18]: dataset['Description'].value_counts()
Out[18]: WHITE HANGING HEART T-LIGHT HOLDER    1940
          JUMBO BAG RED RETROSPOT              1464
          REGENCY CAKESTAND 3 TIER              1426
          ASSORTED COLOUR BIRD ORNAMENT         1333
          PARTY BUNTING                        1308
          ...
          CROCHET LILAC/RED BEAR KEYRING        1
          GLASS CAKE COVER AND PLATE             1
          BLUE PADDED SOFT MOBILE               1
          DIAMANTE NECKLACE BLACK               1
          WHITE ANEMONE ARTIFICIAL FLOWER        1
          Name: Description, Length: 3844, dtype: int64
```

In our dataset, there is a column for quantity which describes the amount of the item that was purchased. Using this column, we will be seeing what item was purchased the most versus the image above.

```
1): #Frequently sold items by quantity
freqsold = dataset.groupby('Description')
freqsoldqua = freqsold['Quantity'].agg(np.sum).sort_values(ascending=False)
freqsoldqua.head(10)

1): Description
PAPER CRAFT , LITTLE BIRDIE      80995
MEDIUM CERAMIC TOP STORAGE JAR  76919
WORLD WAR 2 GLIDERS ASSTD DESIGNS 49182
JUMBO BAG RED RETROSPOT          41981
WHITE HANGING HEART T-LIGHT HOLDER 34648
ASSORTED COLOUR BIRD ORNAMENT    32727
POPCORN HOLDER                  28935
PACK OF 12 LONDON TISSUES        24337
BROCADE RING PURSE              22711
PACK OF 72 RETROSPOT CAKE CASES  22465
          Name: Quantity, dtype: int64
```

We can see that in truth, the item that had the highest number of outgoing stocks is “Paper Craft, Little Birdie” unlike what we had seen before. Afterwards, we grouped our dataset by invoice number. It is better to group it by invoice number instead of customer ID because we can analyse each purchase that way. A customer could have purchased from the store more than once. We then translated the ‘description’ column into a list using the invoice number for grouping. This was done to enable us to use the apriori algorithm as it requires the dataset to be a list.

```
In [28]: #Group by invoice number
groupinvoice = dataset.groupby('InvoiceNo')

In [30]: #Creating a list of the items grouped by the invoice number
retailist = []
for name,group in groupinvoice:
    retailist.append(list(group['Description'].map(str)))
retailist

Out[30]: [['WHITE HANGING HEART T-LIGHT HOLDER',
'WHITE METAL LANTERN',
'CREAM CUPID HEARTS COAT HANGER',
'KNITTED UNION FLAG HOT WATER BOTTLE',
'RED WOOLLY HOTTIE WHITE HEART.',
'SET 7 BABUSHKA NESTING BOXES',
'GLASS STAR FROSTED T-LIGHT HOLDER'],
['HAND WARMER UNION JACK', 'HAND WARMER RED POLKA DOT'],
['ASSORTED COLOUR BIRD ORNAMENT',
'POPPY'S PLAYHOUSE BEDROOM ',
'POPPY'S PLAYHOUSE KITCHEN',
'FELTCRAFT PRINCESS CHARLOTTE DOLL',
'IVORY KNITTED MUG COSY ',
'BOX OF 6 ASSORTED COLOUR TEASPOONS',
'BOX OF VINTAGE JIGSAW BLOCKS ',
'BOX OF VINTAGE ALPHABET BLOCKS',
'HOME BUILDING BLOCK WORD',
'LOVE BUILDING BLOCK WORD',
'RECIPES BOX WITH METAL HEART',
'POPPY'S PLAYHOUSE KITCHEN']]
```

In implementing the apriori algorithm, we had to define the threshold for the support and confidence. We tried out a couple of min\_support and min\_confidence values. Min\_support = 0.01 and min\_confidence = 0.2 gave us a total of 848 rules, min\_support = 0.009 and min\_confidence = 0.2 gave us a total of 1333 rules, and min\_support = 0.01 and min\_confidence = 0.4 gave us a total of 446 rules. We went with the first set of values and the results are shown in the images below.

```
In [22]: #Using the apriori algorithm to create rules
rules = apriori(retailist, min_support = 0.01, min_confidence = 0.2)
associationrules = utils.extract(rules)
rules_retail = pd.DataFrame(associationrules, columns = ['LHS', 'RHS', 'Support', 'Confidence', 'Lift'])
len(rules_retail)

Out[22]: 848

In [23]: #10 rules with the highest support
rules_retail.nlargest(10, "Support")

Out[23]:
```

	LHS	RHS	Support	Confidence	Lift
208	[JUMBO BAG PINK POLKADOT]	[JUMBO BAG RED RETROSPOT]	0.030392	0.623153	7.169917
209	[JUMBO BAG RED RETROSPOT]	[JUMBO BAG PINK POLKADOT]	0.030392	0.349689	7.169917
309	[LUNCH BAG BLACK SKULL]	[LUNCH BAG RED RETROSPOT]	0.029071	0.485944	7.223641
310	[LUNCH BAG RED RETROSPOT]	[LUNCH BAG BLACK SKULL]	0.029071	0.432143	7.223641
105	[GREEN REGENCY TEACUP AND SAUCER]	[ROSES REGENCY TEACUP AND SAUCER]	0.028590	0.777778	19.099148
106	[ROSES REGENCY TEACUP AND SAUCER]	[GREEN REGENCY TEACUP AND SAUCER]	0.028590	0.702065	19.099148
382	[LUNCH BAG PINK POLKADOT]	[LUNCH BAG RED RETROSPOT]	0.028290	0.555425	8.256485
383	[LUNCH BAG RED RETROSPOT]	[LUNCH BAG PINK POLKADOT]	0.028290	0.420536	8.256485
95	[GARDENERS KNEELING PAD CUP OF TEA]	[GARDENERS KNEELING PAD KEEP CALM]	0.027509	0.730463	16.390122
96	[GARDENERS KNEELING PAD KEEP CALM]	[GARDENERS KNEELING PAD CUP OF TEA]	0.027509	0.617251	16.390122

```
In [24]: #10 rules with the highest confidence
rules_retail.nlargest(10, "Confidence")
```

Out[24]:

	LHS	RHS	Support	Confidence	Lift
138	[HERB MARKER THYME]	[HERB MARKER ROSEMARY]	0.010151	0.944134	86.844687
137	[HERB MARKER ROSEMARY]	[HERB MARKER THYME]	0.010151	0.933702	86.844687
821	[WOODEN HEART CHRISTMAS SCANDINAVIAN, WOODEN T...	[WOODEN STAR CHRISTMAS SCANDINAVIAN]	0.010211	0.928962	38.569287
835	[PINK REGENCY TEACUP AND SAUCER, REGENCY CAKES...	[GREEN REGENCY TEACUP AND SAUCER]	0.012133	0.897778	24.423370
534	[PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	[GREEN REGENCY TEACUP AND SAUCER]	0.020482	0.890339	24.221015
832	[PINK REGENCY TEACUP AND SAUCER, GREEN REGENCY...	[ROSES REGENCY TEACUP AND SAUCER]	0.012133	0.878261	21.566615
528	[PINK REGENCY TEACUP AND SAUCER, REGENCY CAKES...	[GREEN REGENCY TEACUP AND SAUCER]	0.013815	0.867925	23.611234
452	[POPPY'S PLAYHOUSE LIVINGROOM]	[POPPY'S PLAYHOUSE KITCHEN]	0.010691	0.851675	49.234483
809	[PINK REGENCY TEACUP AND SAUCER, REGENCY CAKES...	[ROSES REGENCY TEACUP AND SAUCER]	0.013514	0.849057	20.849474
466	[REGENCY TEA PLATE GREEN]	[REGENCY TEA PLATE ROSES]	0.011532	0.845815	52.939750

```
In [25]: #10 rules with the highest lift
rules_retail.nlargest(10, "Lift")
```

Out[25]:

	LHS	RHS	Support	Confidence	Lift
137	[HERB MARKER ROSEMARY]	[HERB MARKER THYME]	0.010151	0.933702	86.844687
138	[HERB MARKER THYME]	[HERB MARKER ROSEMARY]	0.010151	0.944134	86.844687
466	[REGENCY TEA PLATE GREEN]	[REGENCY TEA PLATE ROSES]	0.011532	0.845815	52.939750
467	[REGENCY TEA PLATE ROSES]	[REGENCY TEA PLATE GREEN]	0.011532	0.721805	52.939750
449	[POPPY'S PLAYHOUSE BEDROOM]	[POPPY'S PLAYHOUSE LIVINGROOM]	0.010151	0.650000	51.779187
450	[POPPY'S PLAYHOUSE LIVINGROOM]	[POPPY'S PLAYHOUSE BEDROOM]	0.010151	0.808612	51.779187
484	[SET OF 3 WOODEN STOCKING DECORATION]	[SET OF 3 WOODEN TREE DECORATIONS]	0.010331	0.690763	50.220585
485	[SET OF 3 WOODEN TREE DECORATIONS]	[SET OF 3 WOODEN STOCKING DECORATION]	0.010331	0.751092	50.220585
451	[POPPY'S PLAYHOUSE KITCHEN]	[POPPY'S PLAYHOUSE LIVINGROOM]	0.010691	0.618056	49.234483
452	[POPPY'S PLAYHOUSE LIVINGROOM]	[POPPY'S PLAYHOUSE KITCHEN]	0.010691	0.851675	49.234483

LHS and RHS refers to our If and then situation as earlier stated. LHS is the antecedent and RHS is the consequent. We see that the highest support we can get with these values is 3%. We know that a lift > 1 signifies a strong association between the items. Looking at the values, we see strong associations. We see that most of the items in these rules are complimentary of each other. It is mainly same brand/item but different flavour, colour, or type.

To explore more, we will be creating rules based on the most purchased item as seen earlier on which is the 'White Hanging Heart T-Light Holder'. We decided to go for two items to see what other items can be purchased with the 'White Hanging Heart T-Light Holder'.

```
In [36]: #creating rules based on the most purchased item
```

```
Rules = list(apriori(retailist, min_support = 0.01, min_confidence = 0.3))
associationRules = utils.extract(Rules, 'WHITE HANGING HEART T-LIGHT HOLDER', 2)
utils.inspect(associationRules)

The number of associated rules: 10
LHS: ['CANDLEHOLDER PINK HANGING HEART'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.74, lift: 6.55

LHS: ['HEART OF WICKER LARGE'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.02, confidence: 0.36, lift: 3.20

LHS: ['HOME BUILDING BLOCK WORD'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.30, lift: 2.69

LHS: ['LOVE BUILDING BLOCK WORD'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.32, lift: 2.85

LHS: ['RED HANGING HEART T-LIGHT HOLDER'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.03, confidence: 0.67, lift: 5.89

LHS: ['WOOD 2 DRAWER CABINET WHITE FINISH'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.34, lift: 2.96

LHS: ['WOODEN FRAME ANTIQUE WHITE'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.02, confidence: 0.32, lift: 2.82

LHS: ['WOODEN PICTURE FRAME WHITE FINISH'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.02, confidence: 0.36, lift: 3.16

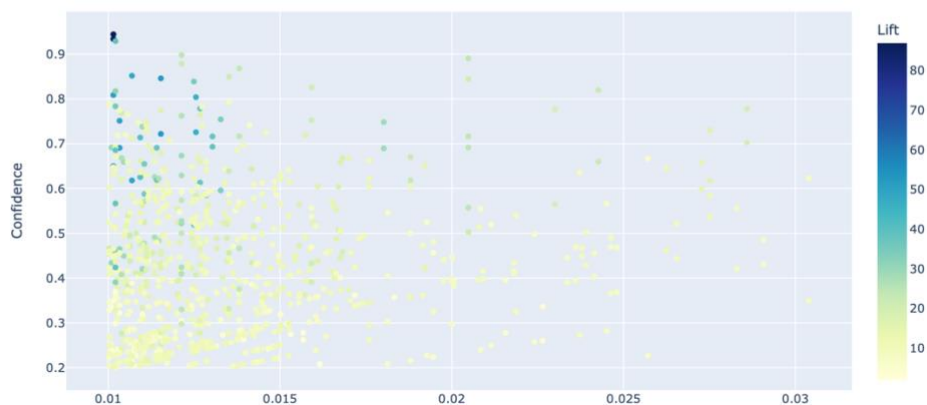
LHS: ['ZINC METAL HEART DECORATION'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.37, lift: 3.27

LHS: ['WOODEN PICTURE FRAME WHITE FINISH', 'WOODEN FRAME ANTIQUE WHITE'] --> RHS: ['WHITE HANGING HEART T-LIGHT HOLDER'], support: 0.01, confidence: 0.39, lift: 3.43
```

## Results analysis and discussion

We created a plot to visually see and understand the 848 rules generated.

```
In [39]: #visualisation of the rules generated
import plotly.express as px
rules_retail = pd.DataFrame(associationrules, columns = ['LHS', 'RHS', 'Support', 'Confidence', 'Lift'])
fig = px.scatter(rules_retail, x = "Support", y = "Confidence", color = "Lift", hover_data=['LHS','RHS'],
                 color_continuous_scale= "ylgnbu")
fig.show()
```



We see that the values for support are within 1% and 3%. With more of the rules having a low support value. Although, we can see that we do have some rules with considerably high values of lift, majority of the rules have lifts with values in the 10 range. The rules are mostly natural, meaning it is no conundrum as to why the items in the rules would be purchased together.

## Conclusions

The frequent itemset mining is very useful and efficient in creating groups of items that are most likely to be gotten together. The apriori algorithm is one that is very efficient in doing so. With the simplicity of the algorithm and its ability to deal with large datasets, we can say that this algorithm performs well. We can also confirm that, we can use apriori algorithm on transactional dataset like the one we used in this study.