analysis

August 7, 2018

1 MAG-O data analysis test

Stephen Checkley. August 2018.

1.1 Task 0.1 - set up the Python environment

```
In [63]: import math
         from IPython import display
         from matplotlib import cm
         from matplotlib import gridspec
         from matplotlib import pyplot as plt
         import seaborn as sns
         import numpy as np
         import pandas as pd
         import missingno as msno
         from sklearn import metrics
         from sklearn.preprocessing import scale, StandardScaler, normalize
         from sklearn import decomposition
         from sklearn.decomposition import PCA
         from sklearn.cluster import KMeans
         from sklearn.metrics import silhouette_samples, silhouette_score
         import pickle
         from collections import Counter

         pd.options.display.max_rows = 100
         pd.options.display.max_columns = 100
         pd.options.display.float_format = '{:.1f}'.format

         %matplotlib inline
```

1.2 task 1 - data analysis

1.2.1 Data import and cleaning

```
In [64]: data = pd.read_csv('./data.csv',encoding='ISO-8859-1')
```

```
In [65]: data.shape
```

```
Out[65]: (541909, 8)
```

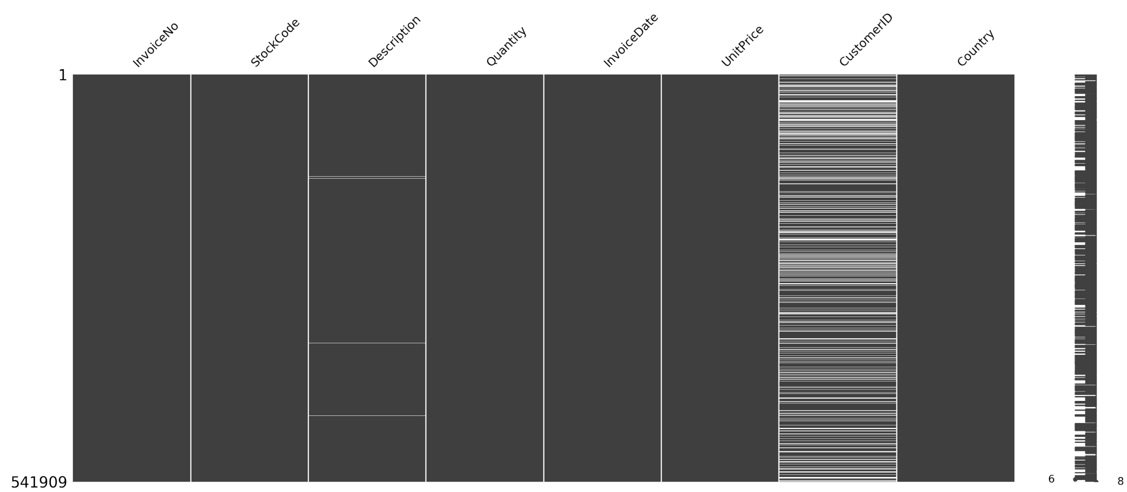
The data consists of 8 columns and 541909 rows.

```
In [66]: data.head(5)
```

```
Out[66]:   InvoiceNo StockCode                          Description  Quantity  \
         0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
         1    536365     71053                  WHITE METAL LANTERN         6
         2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
         3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
         4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

               InvoiceDate  UnitPrice  CustomerID         Country
         0  12/1/2010 8:26        2.5     17850.0  United Kingdom
         1  12/1/2010 8:26        3.4     17850.0  United Kingdom
         2  12/1/2010 8:26        2.8     17850.0  United Kingdom
         3  12/1/2010 8:26        3.4     17850.0  United Kingdom
         4  12/1/2010 8:26        3.4     17850.0  United Kingdom
```

A cursory check for missing data:

```
In [67]: msno.matrix(data);
```



The dataset is missing some Description and customer ID data entries.

```
In [68]: null_data = data[data.isnull().any(axis=1)]
         null_data.shape
```

```
Out[68]: (135080, 8)
```

```
In [69]: null_data_frac = null_data.shape[0]/data.shape[0]*100
         null_data_frac
```

Out[69]: 24.926694334288598

25% of the data contains missing values, which the figure above indicates is mostly nan values in CustomerID

1.3 Exploratory data analysis

```
In [70]: data = data.sort_values('Quantity', ascending=False)
         data.head(11)
```

```
Out[70]:         InvoiceNo StockCode                          Description  Quantity  \
         540421     581483     23843          PAPER CRAFT , LITTLE BIRDIE     80995
         61619      541431     23166       MEDIUM CERAMIC TOP STORAGE JAR     74215
         502122     578841     84826       ASSTD DESIGN 3D PAPER STICKERS     12540
         74614      542504     37413                                  NaN      5568
         421632     573008     84077    WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
         206121     554868     22197                 SMALL POPCORN HOLDER      4300
         220843     556231    85123A                                    ?      4000
         97432      544612     22053                EMPIRE DESIGN ROSETTE      3906
         270885     560599     18007  ESSENTIAL BALM 3.5g TIN IN ENVELOPE      3186
         160546     550461     21108   FAIRY CAKE FLANNEL ASSORTED COLOUR      3114
         52711      540815     21108   FAIRY CAKE FLANNEL ASSORTED COLOUR      3114

                       InvoiceDate  UnitPrice  CustomerID         Country
         540421     12/9/2011 9:15        2.1     16446.0  United Kingdom
         61619     1/18/2011 10:01        1.0     12346.0  United Kingdom
         502122   11/25/2011 15:57        0.0     13256.0  United Kingdom
         74614     1/28/2011 12:03        0.0         nan  United Kingdom
         421632   10/27/2011 12:26        0.2     12901.0  United Kingdom
         206121    5/27/2011 10:52        0.7     13135.0  United Kingdom
         220843     6/9/2011 15:04        0.0         nan  United Kingdom
         97432     2/22/2011 10:43        0.8     18087.0  United Kingdom
         270885    7/19/2011 17:04        0.1     14609.0  United Kingdom
         160546    4/18/2011 13:20        2.1     15749.0  United Kingdom
         52711     1/11/2011 12:55        2.1     15749.0  United Kingdom
```

The top 10 most popular items are sold in/to the UK. Apparently paper craft little birdie is very popular, along with medium ceramic top storage jar. I will remove the items with 'NaN' descriptor. These entries associated with nan CustomerID entries and 131 lower case descriptions which describe problems with the orders and no details of the item ordered. In addition, the data for United Kingdom contains negative values associated with negative UnitPrice values. Removing the rows containing NaN values therefore cleans several issues that complicate this analysis in the absence of the data owner.

```
In [71]: data = data.dropna()
```

```
In [72]: data.shape # 541909 - 406829 = dropped 135,080 entries
```

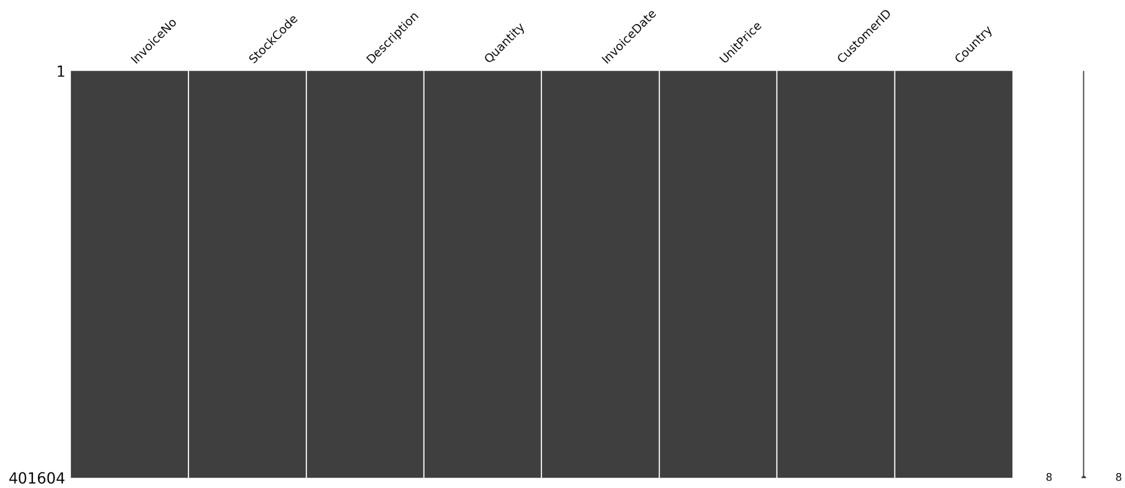
Out[72]: (406829, 8)

```
In [73]: # in addition I will check for duplicate entries and remove those
         data.drop_duplicates(inplace = True)
```

```
In [74]: data.shape
```

```
Out[74]: (401604, 8)
```

```
In [75]: msno.matrix(data);
```



We now have no missing values and have removed assumed erroneous results and duplicates.

1.4 Task 2 - Further exploration and trend analysis

How many unique descriptors are there in "Descriptions"?

```
In [76]: # general clean up of Description column to remove any heading or trailing white spac
         data['Description'] = data['Description'].str.strip()
         descriptors = pd.unique(data['Description'].values.ravel()).tolist()
         print('There are',len(descriptors), 'unique descriptors in the Descriptions column.')
```

```
There are 3885 unique descriptors in the Descriptions column.
```

```
In [77]: descriptors2 = []
         for i in range(0,len(descriptors),1):
             text = str(descriptors[i])
             if text == text.upper():
                 descriptors2.append(text)

         print('There are', len(descriptors2), 'all caps descriptors.')
```

```
There are 3862 all caps descriptors.
```

```
In [78]: # from inspection of the baskets below I identified 3 problem descriptors that need t
         additional_problem_list = ['damages/credits from ASOS.','incorrectly credited C550456
```

```
In [79]: # remove additional erroneous rows
         data = data[~data['Description'].isin(additional_problem_list)]
```

```
In [80]: data.shape
```

```
Out[80]: (401604, 8)
```

```
In [81]: InvoiceNo = pd.unique(data['InvoiceNo'].values.ravel()).tolist()
         print('There are', len(InvoiceNo), 'unique invoice numbers in the dataset.')
```

```
There are 22190 unique invoice numbers in the dataset.
```

1.4.1 Group by country

```
In [82]: by_country = data.groupby('Country')
         by_country.describe()
```

```
Out[82]:                     CustomerID                                              \
                                 count    mean    std     min     25%     50%
         Country
         Australia              1258.0 12464.7  438.0 12386.0 12415.0 12415.0
         Austria                 401.0 12521.5  216.5 12358.0 12360.0 12374.0
         Bahrain                  17.0 12354.5    0.9 12353.0 12355.0 12355.0
         Belgium                2069.0 12430.3  110.0 12361.0 12383.0 12407.0
         Brazil                   32.0 12769.0    0.0 12769.0 12769.0 12769.0
         Canada                  151.0 17321.1  521.5 15388.0 17444.0 17444.0
         Channel Islands         757.0 14888.1  142.8 14442.0 14930.0 14936.0
         Cyprus                  611.0 12405.4  200.6 12359.0 12359.0 12370.0
         Czech Republic           30.0 12781.0    0.0 12781.0 12781.0 12781.0
         Denmark                 389.0 12536.6  421.9 12367.0 12406.0 12412.0
         EIRE                   7475.0 14748.7  314.5 14016.0 14911.0 14911.0
         European Community       61.0 15108.0    0.0 15108.0 15108.0 15108.0
         Finland                 695.0 12517.0  122.4 12348.0 12428.0 12428.0
         France                 8475.0 12677.5  275.4 12413.0 12571.0 12674.0
         Germany                9480.0 12645.8  307.9 12426.0 12480.0 12592.0
         Greece                  146.0 13757.4 1749.6 12478.0 12717.0 12717.0
         Iceland                 182.0 12347.0    0.0 12347.0 12347.0 12347.0
         Israel                  247.0 12659.6   57.6 12512.0 12653.0 12688.0
         Italy                   803.0 12648.4  437.4 12349.0 12578.0 12584.0
         Japan                   358.0 12757.8   13.6 12753.0 12753.0 12753.0
         Lebanon                  45.0 12764.0    0.0 12764.0 12764.0 12764.0
         Lithuania                35.0 15332.0    0.0 15332.0 15332.0 15332.0
         Malta                   127.0 16996.0 1127.5 15480.0 15480.0 17828.0
         Netherlands            2371.0 14420.3  609.5 12759.0 14646.0 14646.0
         Norway                 1086.0 12438.0   76.7 12350.0 12432.0 12433.0
```

Poland	341.0	12733.1	94.9	12576.0	12576.0	12779.0
Portugal	1471.0	12746.4	97.3	12356.0	12757.0	12766.0
RSA	58.0	12446.0	0.0	12446.0	12446.0	12446.0
Saudi Arabia	10.0	12565.0	0.0	12565.0	12565.0	12565.0
Singapore	229.0	12744.0	0.0	12744.0	12744.0	12744.0
Spain	2528.0	12906.1	1272.4	12354.0	12484.0	12540.0
Sweden	461.0	14701.4	2379.8	12483.0	12638.0	12697.0
Switzerland	1877.0	12667.0	460.8	12357.0	12378.0	12451.0
USA	291.0	12618.9	38.5	12558.0	12607.0	12607.0
United Arab Emirates	68.0	14984.6	2546.1	12739.0	12739.0	12739.0
United Kingdom	356728.0	15543.8	1594.3	12346.0	14191.0	15513.0
Unspecified	241.0	13733.7	1520.9	12363.0	12743.0	12743.0

			Quantity					\
	75%	max	count	mean	std	min	25%	50%
Country								
Australia	12415.0	16321.0	1258.0	66.5	97.7	-120.0	6.0	24.0
Austria	12818.0	12865.0	401.0	12.0	21.7	-48.0	6.0	9.0
Bahrain	12355.0	12355.0	17.0	15.3	25.0	2.0	6.0	6.0
Belgium	12431.0	12876.0	2069.0	11.2	13.6	-12.0	4.0	10.0
Brazil	12769.0	12769.0	32.0	11.1	8.5	2.0	3.0	10.0
Canada	17444.0	17844.0	151.0	18.3	46.7	1.0	6.0	12.0
Channel Islands	14936.0	14937.0	757.0	12.5	22.6	-2.0	4.0	10.0
Cyprus	12391.0	13809.0	611.0	10.3	23.4	-33.0	2.0	5.0
Czech Republic	12781.0	12781.0	30.0	19.7	22.8	-24.0	12.0	24.0
Denmark	12429.0	13919.0	389.0	21.0	27.4	-25.0	12.0	12.0
EIRE	14911.0	14911.0	7475.0	18.2	42.0	-288.0	4.0	10.0
European Community	15108.0	15108.0	61.0	8.1	6.5	-2.0	3.0	6.0
Finland	12631.0	12704.0	695.0	15.3	21.0	-27.0	6.0	10.0
France	12689.0	14277.0	8475.0	13.0	21.5	-250.0	5.0	10.0
Germany	12662.0	14335.0	9480.0	12.4	17.9	-288.0	5.0	10.0
Greece	14439.0	17508.0	146.0	10.7	7.7	-1.0	5.2	10.0
Iceland	12347.0	12347.0	182.0	13.5	18.9	2.0	6.0	12.0
Israel	12688.0	12688.0	247.0	16.1	16.7	-32.0	4.0	12.0
Italy	12610.0	14912.0	803.0	10.0	13.6	-12.0	4.0	6.0
Japan	12754.0	12812.0	358.0	70.4	177.2	-624.0	4.0	36.0
Lebanon	12764.0	12764.0	45.0	8.6	4.3	2.0	6.0	8.0
Lithuania	15332.0	15332.0	35.0	18.6	10.1	6.0	12.0	16.0
Malta	17828.0	17828.0	127.0	7.4	8.1	-4.0	3.0	6.0
Netherlands	14646.0	14646.0	2371.0	84.4	111.4	-480.0	16.0	72.0
Norway	12438.0	12752.0	1086.0	17.7	22.6	-12.0	6.0	12.0
Poland	12779.0	12816.0	341.0	10.7	10.2	-6.0	4.0	10.0
Portugal	12782.5	12811.0	1471.0	10.9	11.9	-12.0	4.0	10.0
RSA	12446.0	12446.0	58.0	6.1	3.3	1.0	3.0	6.0
Saudi Arabia	12565.0	12565.0	10.0	7.5	5.7	-5.0	6.0	9.0
Singapore	12744.0	12744.0	229.0	22.9	27.7	-1.0	8.0	12.0
Spain	12550.0	17097.0	2528.0	10.6	24.2	-288.0	3.0	6.0
Sweden	17404.0	17404.0	461.0	77.3	129.0	-240.0	8.0	20.0

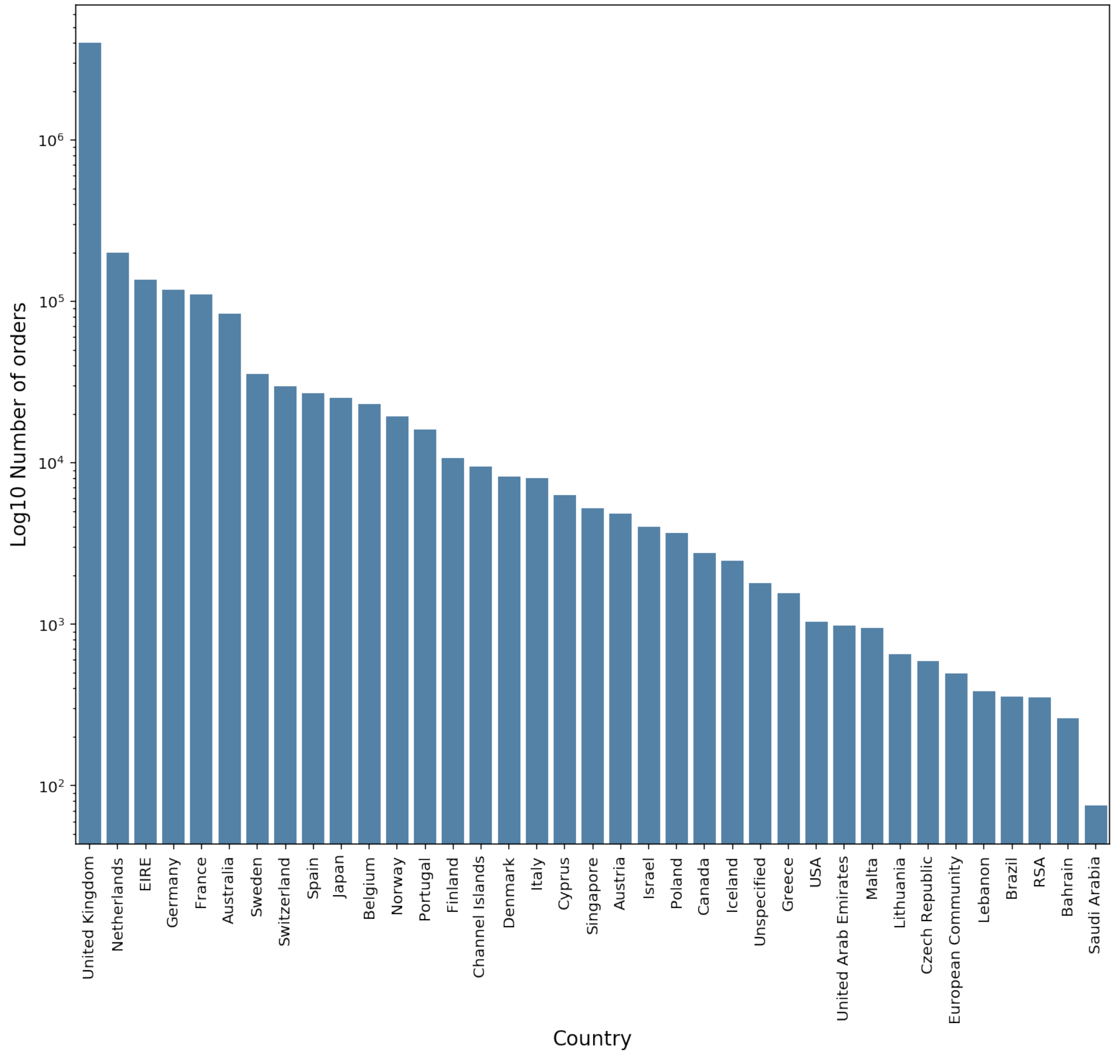
Switzerland	12458.0	13520.0	1877.0	15.9	19.3	-120.0	6.0	12.0
USA	12607.0	12733.0	291.0	3.6	16.5	-36.0	-10.0	5.0
United Arab Emirates	17829.0	17829.0	68.0	14.4	12.5	1.0	6.0	12.0
United Kingdom	16931.0	18287.0	356728.0	11.2	265.0	-80995.0	2.0	4.0
Unspecified	14265.0	16320.0	241.0	7.4	8.9	1.0	1.0	2.0

			UnitPrice							\
	75%	max	count	mean	std	min	25%	50%	75%	
Country										
Australia	96.0	1152.0	1258.0	3.2	12.5	0.0	1.2	1.8	3.8	
Austria	12.0	288.0	401.0	4.2	7.4	0.1	1.2	1.9	4.2	
Bahrain	8.0	96.0	17.0	4.6	3.7	1.2	1.6	3.0	5.0	
Belgium	12.0	272.0	2069.0	3.6	4.2	0.1	1.2	1.9	4.2	
Brazil	18.0	24.0	32.0	4.5	2.8	0.8	2.0	3.3	6.8	
Canada	20.0	504.0	151.0	6.0	44.7	0.1	0.8	1.6	3.0	
Channel Islands	12.0	407.0	757.0	4.9	15.6	0.2	1.4	2.5	6.2	
Cyprus	12.0	288.0	611.0	6.4	22.6	0.1	1.2	3.0	5.0	
Czech Republic	24.0	72.0	30.0	2.9	7.1	0.3	0.8	1.4	2.4	
Denmark	24.0	256.0	389.0	3.3	4.0	0.2	1.2	1.9	3.8	
EIRE	12.0	1440.0	7475.0	5.1	41.8	0.0	1.2	2.1	4.2	
European Community	12.0	24.0	61.0	4.8	4.4	0.6	1.4	3.4	6.8	
Finland	12.0	144.0	695.0	5.4	13.6	0.1	0.8	2.1	4.5	
France	12.0	912.0	8475.0	5.1	80.3	0.0	1.2	1.8	3.8	
Germany	12.0	600.0	9480.0	4.0	16.6	0.0	1.2	1.9	3.8	
Greece	12.0	48.0	146.0	4.9	8.5	0.1	1.2	2.1	5.5	
Iceland	12.0	240.0	182.0	2.6	2.3	0.2	1.2	2.0	3.8	
Israel	24.0	100.0	247.0	3.7	9.4	0.1	0.8	1.6	3.8	
Italy	12.0	200.0	803.0	4.8	11.8	0.1	1.6	2.5	5.0	
Japan	72.0	2040.0	358.0	2.3	3.1	0.2	0.8	1.6	2.5	
Lebanon	12.0	24.0	45.0	5.4	4.1	0.6	2.5	4.0	8.0	
Lithuania	24.0	48.0	35.0	2.8	1.4	1.2	1.6	2.5	3.8	
Malta	12.0	48.0	127.0	5.2	9.4	0.2	1.4	3.0	5.0	
Netherlands	100.0	2400.0	2371.0	2.7	6.3	0.0	0.8	1.4	2.5	
Norway	24.0	240.0	1086.0	6.0	30.6	0.0	1.2	2.1	5.0	
Poland	12.0	72.0	341.0	4.2	5.9	0.2	1.2	2.1	5.0	
Portugal	12.0	120.0	1471.0	8.8	72.5	0.1	1.2	1.6	3.0	
RSA	9.5	12.0	58.0	4.3	3.7	0.0	1.7	3.0	5.0	
Saudi Arabia	12.0	12.0	10.0	2.4	1.4	0.4	1.6	2.3	3.0	
Singapore	24.0	288.0	229.0	109.6	515.3	0.2	1.2	2.1	4.2	
Spain	12.0	360.0	2528.0	5.0	41.0	0.0	1.2	2.1	4.2	
Sweden	96.0	768.0	461.0	3.9	8.3	0.2	0.8	1.6	3.0	
Switzerland	24.0	288.0	1877.0	3.5	5.5	0.0	1.2	1.8	3.8	
USA	12.0	72.0	291.0	2.2	2.3	0.4	0.8	1.4	3.0	
United Arab Emirates	12.0	72.0	68.0	3.4	5.3	0.3	1.1	1.7	3.3	
United Kingdom	12.0	80995.0	356728.0	3.3	71.2	0.0	1.2	1.9	3.8	
Unspecified	12.0	36.0	241.0	3.2	3.4	0.2	1.2	2.1	4.2	

```
                          max
Country
Australia               350.0
Austria                  40.0
Bahrain                  12.8
Belgium                  40.0
Brazil                   10.9
Canada                  550.9
Channel Islands         293.0
Cyprus                  320.7
Czech Republic           40.0
Denmark                  18.0
EIRE                   1687.2
European Community       18.0
Finland                 275.6
France                 4161.1
Germany                 599.5
Greece                   50.0
Iceland                  12.8
Israel                  125.0
Italy                   300.0
Japan                    45.6
Lebanon                  14.9
Lithuania                 6.0
Malta                    65.0
Netherlands             206.4
Norway                  700.0
Poland                   40.0
Portugal               1242.0
RSA                      14.9
Saudi Arabia              5.5
Singapore              3949.3
Spain                  1715.8
Sweden                   40.0
Switzerland              40.0
USA                      16.9
United Arab Emirates     37.5
United Kingdom        38970.0
Unspecified              16.9
```

In [83]:
```
grouped  = data.groupby(['Country']).sum()['Quantity'].sort_values(ascending=False)

f, ax = plt.subplots(figsize=(12, 10))
plt.xticks(rotation='vertical')
sns.barplot(grouped.index, grouped.values, color='steelblue')
f.get_axes()[0].set_yscale('log') #I'm using a log scale just for visualisation as th
plt.ylabel('Log10 Number of orders', fontsize=13)
plt.xlabel('Country', fontsize=13)
```

```
plt.show()
```



The United Kingdom purchases the majority of products by almost 2 orders of magnitude.

In [84]:
```
# group by customer ID and invoice number to create a basket per customer
temp = data.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['InvoiceDate'].cou
products_per_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})
products_per_basket[:10].sort_values('Number of products', ascending=False)
```

Out[84]:

	CustomerID	InvoiceNo	Number of products
7	12347.0	573511	47
2	12347.0	537626	31
3	12347.0	542237	29
4	12347.0	549222	24
6	12347.0	562032	22
5	12347.0	556201	18

```
9       12348.0    539318                    17
8       12347.0    581180                    11
0       12346.0    541431                     1
1       12346.0   C541433                     1
```

There are InvoiceNo entries beginning with the character C.

```
In [85]: print('There are:',data['InvoiceNo'].str.contains("C").sum(), 'orders marked C, which
```

```
There are: 8872 orders marked C, which contribute 2.209141343213713 % of the dataset.
```

```
In [86]: # select all the cancelled orders
         cancelled_orders = data[data.InvoiceNo.str.contains("C")]
         cancelled_orders.head(10)
```

```
Out[86]:        InvoiceNo StockCode                          Description  Quantity  \
         268308   C560408         M                               Manual        -1
         186013   C552841     22838       3 TIER CAKE TIN RED AND CREAM        -1
         169480   C551175     22325             MOBILE VINTAGE HEARTS        -1
         429996   C573575      CRUK                     CRUK Commission        -1
         281674   C561591     22768          FAMILY PHOTO FRAME CORNICE        -1
         268312   C560409    84078A      SET/4 WHITE RETRO STORAGE CUBES        -1
         355585   C567947     23234       BISCUIT TIN VINTAGE CHRISTMAS        -1
         355584   C567947     21201  TROPICAL  HONEYCOMB PAPER GARLAND        -1
         96677    C544577         M                               Manual        -1
         45144    C540250     21928     JUMBO BAG SCANDINAVIAN PAISLEY        -1

                     InvoiceDate  UnitPrice  CustomerID         Country
         268308   7/18/2011 14:24      550.6     13564.0  United Kingdom
         186013   5/11/2011 14:28       14.9     15827.0  United Kingdom
         169480   4/26/2011 17:17        5.0     14329.0  United Kingdom
         429996  10/31/2011 14:09      606.0     14096.0  United Kingdom
         281674   7/28/2011 11:17        9.9     15708.0  United Kingdom
         268312   7/18/2011 14:24       40.0     16717.0  United Kingdom
         355585    9/23/2011 8:00        2.9     17663.0  United Kingdom
         355584    9/23/2011 8:00        2.5     17663.0  United Kingdom
         96677     2/21/2011 14:02     320.7     12365.0          Cyprus
         45144      1/5/2011 16:02       1.6     17511.0  United Kingdom
```

InvoiceNo containing a "C" character correspond with a negative Quantity value, therefore I will assume these are cancelled orders. For the purposes of this report I am going to remove cancelled orders from consideration.

```
In [87]: data = data[~data['InvoiceNo'].str.contains("C")]
```

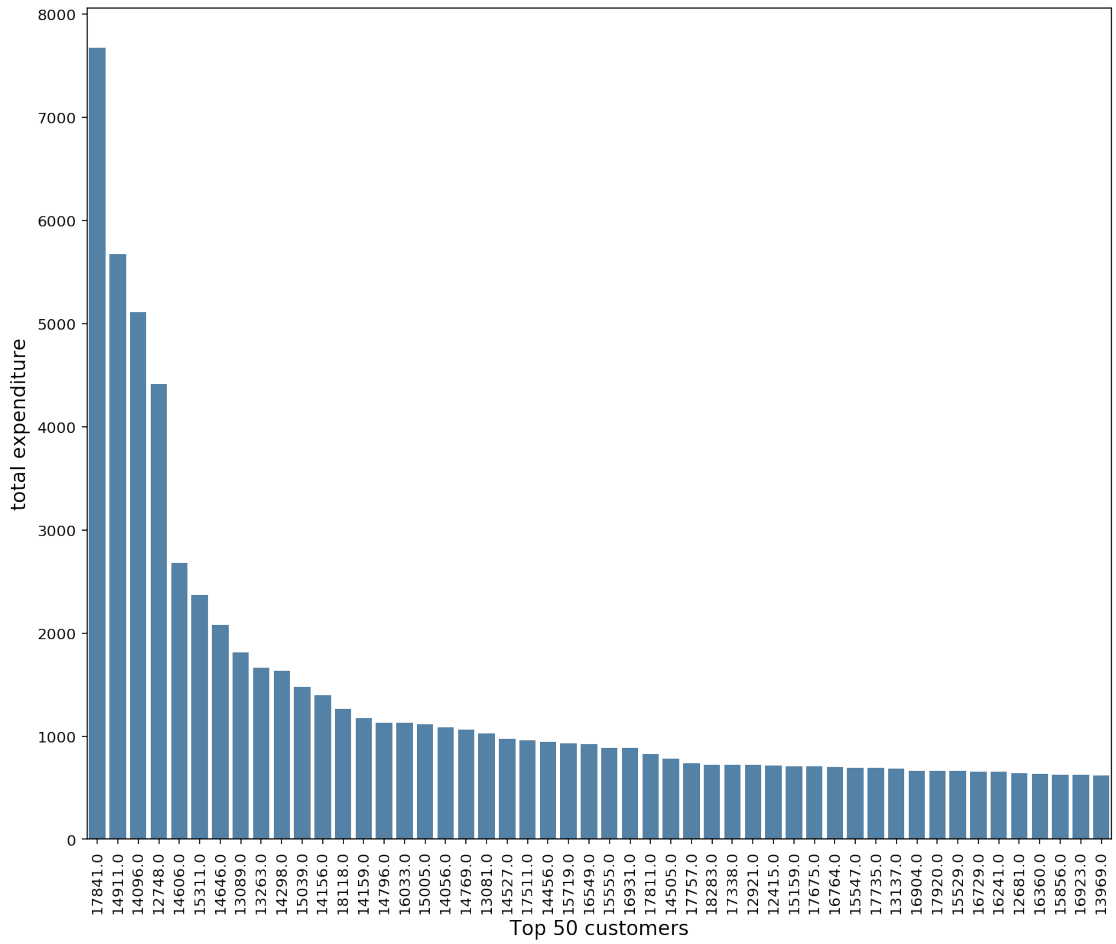
```
In [88]: # group by customer ID to create a rank buyers by how many products they buy in total
         temp = data.groupby(by=['CustomerID'], as_index=False)['InvoiceDate'].count()
         products_per_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})
         top_baskets = products_per_basket.sort_values('Number of products', ascending=False)
         top_baskets.head(10)
```

Out[88]:

	CustomerID	Number of products
4011	17841.0	7676
1880	14911.0	5672
1290	14096.0	5111
326	12748.0	4413
1662	14606.0	2677
2177	15311.0	2366
1690	14646.0	2080
562	13089.0	1814
691	13263.0	1667
1435	14298.0	1637

In [89]:
```
grouped = top_baskets[:50]
grouped.reset_index(level=0, inplace=True)
grouped.sort_values('Number of products', ascending=False, inplace=True)

f, ax = plt.subplots(figsize=(12, 10))
plt.xticks(rotation='vertical')
sns.barplot(x=grouped['CustomerID'], y=grouped['Number of products'],order=grouped['Cu
plt.ylabel('total expenditure', fontsize=13)
plt.xlabel('Top 50 customers', fontsize=13)
plt.show()
```

```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```



1.4.2 Investigation of which items are contained in the top CustomerID basket

In [ ]:
```
top_basket = data[data['CustomerID'] == top_baskets['CustomerID'].iloc[0]]
#top_basket #uncomment to view basket contents
```

Note - this investigation was used with apriori modeling detailed in the Addendum section.

1.4.3 Investigation of StockCode

There are some non-integer values in StockCodes which correspond with order descriptions that are not items.

In [30]:
```
mask = (~data['StockCode'].str.contains('[0-9]'))
odd_stock_codes = data.loc[mask] #filter out any stock codes that are numeric to leav
odd_stock_codes.head()
```

Out[30]:
```
        InvoiceNo StockCode Description  Quantity       InvoiceDate  UnitPrice  \
490502    578060         M      Manual      1600  11/22/2011 15:22        0.2
```

```
452218    575328          M      Manual      1200   11/9/2011 13:48         0.2
437235    574277          M      Manual       832   11/3/2011 14:42         0.2
526018    580646          M      Manual       800   12/5/2011 13:13         0.2
414138    572344          M      Manual       456  10/24/2011 10:43         1.5

        CustomerID         Country
490502     17857.0  United Kingdom
452218     17857.0  United Kingdom
437235     17857.0  United Kingdom
526018     17857.0  United Kingdom
414138     14607.0  United Kingdom
```

```
In [31]: odd_stock_codes['StockCode'].unique(), print('total number of these short stock code e
```

```
total number of these short stock code entries: 1416
```

```
Out[31]: (array(['M', 'POST', 'DOT', 'BANK CHARGES', 'PADS'], dtype=object), None)
```

As the number of non-standard stock codes is small, for the purposes of this report they will be deleted from the dataset.

```
In [32]: data = data.loc[~mask]
```

During the data cleaning process NaN, duplicate entries, cancelled invoices, and miscellaneous stock codes have been removed.

1.5 Investigation of Invoice Date

```
In [33]: timestamp_list = list(data.InvoiceDate)

         Timeframe = pd.DataFrame(pd.to_datetime(timestamp_list), columns=['time'])
```

```
In [34]: data['time'] = Timeframe['time'].values
```

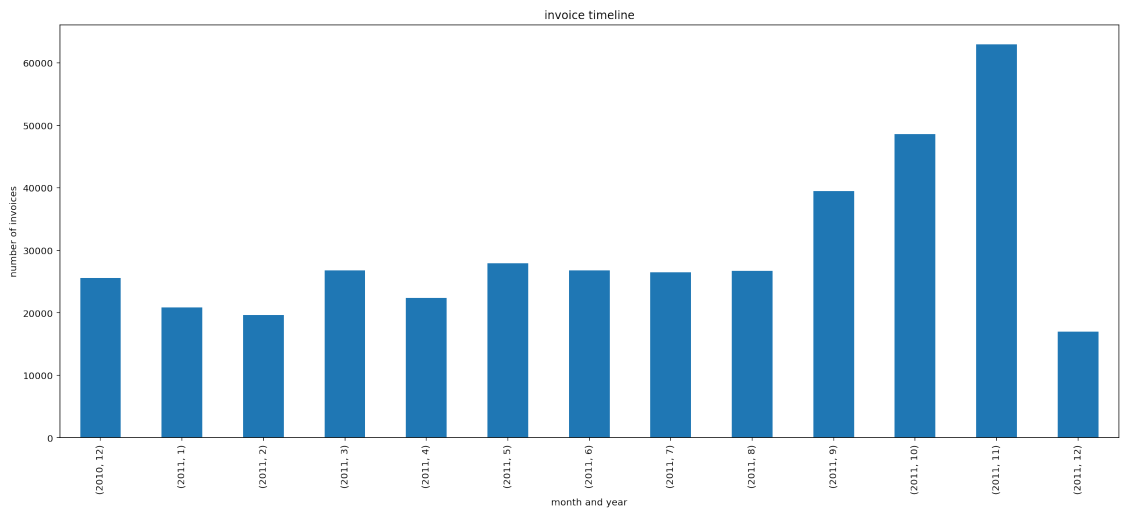
```
In [35]: data.head()
```

```
Out[35]:        InvoiceNo StockCode                          Description  Quantity  \
         540421    581483     23843          PAPER CRAFT , LITTLE BIRDIE     80995
         61619     541431     23166       MEDIUM CERAMIC TOP STORAGE JAR     74215
         502122    578841     84826       ASSTD DESIGN 3D PAPER STICKERS     12540
         421632    573008     84077    WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
         206121    554868     22197                 SMALL POPCORN HOLDER      4300

                      InvoiceDate  UnitPrice  CustomerID         Country  \
         540421    12/9/2011 9:15        2.1     16446.0  United Kingdom
         61619    1/18/2011 10:01        1.0     12346.0  United Kingdom
         502122  11/25/2011 15:57        0.0     13256.0  United Kingdom
         421632  10/27/2011 12:26        0.2     12901.0  United Kingdom
```

```
206121   5/27/2011 10:52        0.7     13135.0   United Kingdom

                           time
540421 2011-12-09 09:15:00
61619  2011-01-18 10:01:00
502122 2011-11-25 15:57:00
421632 2011-10-27 12:26:00
206121 2011-05-27 10:52:00
```

In [36]:
```
plot_dims = (20, 8)

plot = Timeframe.groupby((Timeframe['time'].dt.year, Timeframe['time'].dt.month.renam

plot.set(xlabel='month and year', ylabel='number of invoices',title="invoice timeline
plt.xticks(rotation=90)
plt.show()
```

```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning
  This is separate from the ipykernel package so we can avoid doing imports until
```



Invoice numbers increase in September and October and peak in November, possibly attributed to Christmas shopping.

2 Task 3 - Feature engineering

In [37]:
```
data.head()
```

Out[37]:
```
         InvoiceNo StockCode                      Description  Quantity  \
540421     581483     23843         PAPER CRAFT , LITTLE BIRDIE     80995
61619      541431     23166     MEDIUM CERAMIC TOP STORAGE JAR     74215
```

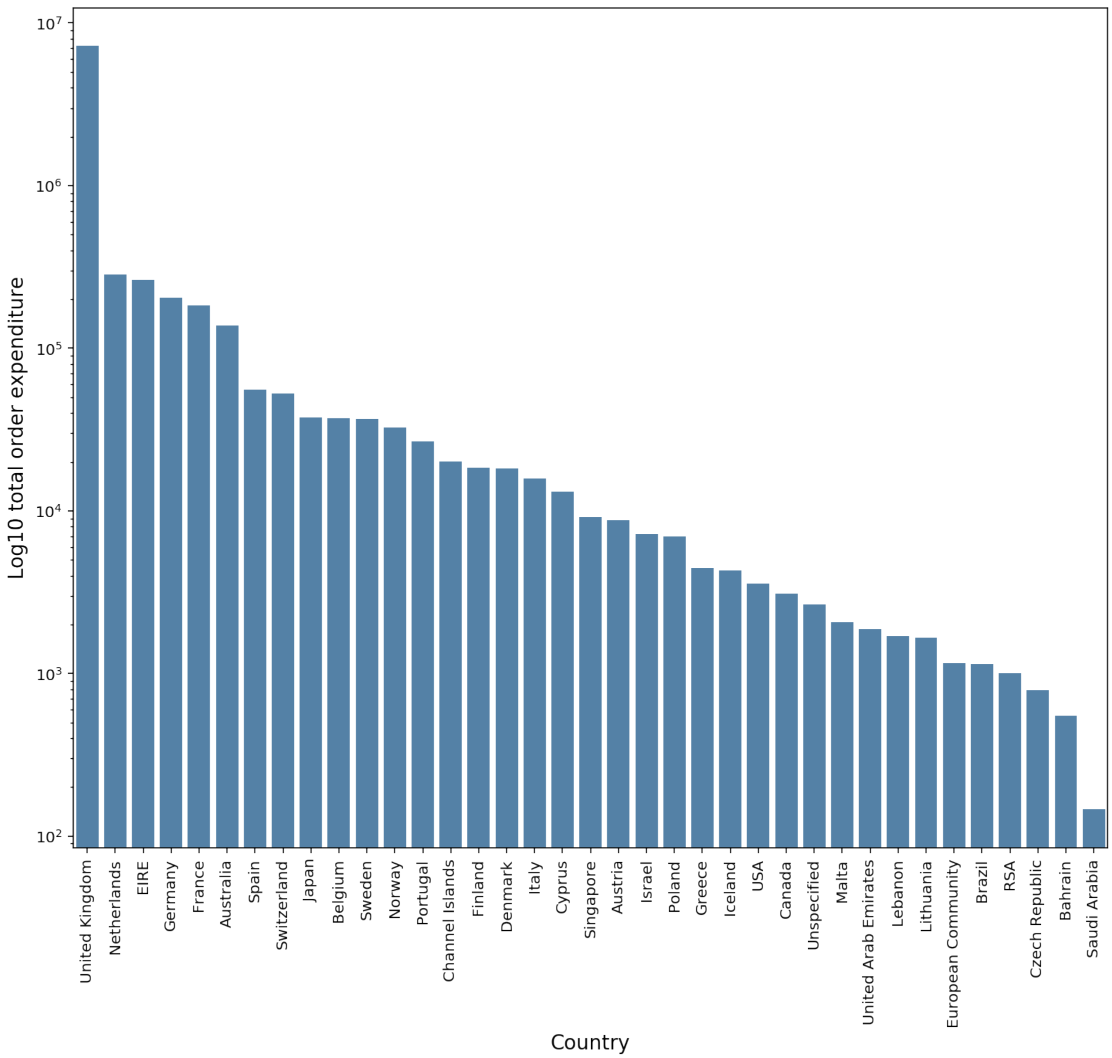
```
502122    578841     84826     ASSTD DESIGN 3D PAPER STICKERS     12540
421632    573008     84077  WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
206121    554868     22197                 SMALL POPCORN HOLDER      4300

              InvoiceDate  UnitPrice  CustomerID         Country  \
540421    12/9/2011 9:15        2.1     16446.0  United Kingdom
61619    1/18/2011 10:01        1.0     12346.0  United Kingdom
502122  11/25/2011 15:57        0.0     13256.0  United Kingdom
421632  10/27/2011 12:26        0.2     12901.0  United Kingdom
206121   5/27/2011 10:52        0.7     13135.0  United Kingdom

                      time
540421 2011-12-09 09:15:00
61619  2011-01-18 10:01:00
502122 2011-11-25 15:57:00
421632 2011-10-27 12:26:00
206121 2011-05-27 10:52:00
```

2.0.1 Investigating the total amount spent per customers

```
In [38]: total_spend = data['Quantity'] * data['UnitPrice']

In [39]: data = data.assign(total_spend=total_spend.values)

In [40]: data.head()

Out[40]:        InvoiceNo StockCode                        Description  Quantity  \
540421    581483     23843        PAPER CRAFT , LITTLE BIRDIE     80995
61619     541431     23166     MEDIUM CERAMIC TOP STORAGE JAR     74215
502122    578841     84826     ASSTD DESIGN 3D PAPER STICKERS     12540
421632    573008     84077  WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
206121    554868     22197                 SMALL POPCORN HOLDER      4300

              InvoiceDate  UnitPrice  CustomerID         Country  \
540421    12/9/2011 9:15        2.1     16446.0  United Kingdom
61619    1/18/2011 10:01        1.0     12346.0  United Kingdom
502122  11/25/2011 15:57        0.0     13256.0  United Kingdom
421632  10/27/2011 12:26        0.2     12901.0  United Kingdom
206121   5/27/2011 10:52        0.7     13135.0  United Kingdom

                      time  total_spend
540421 2011-12-09 09:15:00     168469.6
61619  2011-01-18 10:01:00      77183.6
502122 2011-11-25 15:57:00          0.0
421632 2011-10-27 12:26:00       1008.0
206121 2011-05-27 10:52:00       3096.0
```

2.0.2 Grouped per country

```
In [41]: grouped  = data.groupby(['Country']).sum()['total_spend'].sort_values(ascending=False

         f, ax = plt.subplots(figsize=(12, 10))
         plt.xticks(rotation='vertical')
         sns.barplot(grouped.index, grouped.values, color='steelblue')
         f.get_axes()[0].set_yscale('log') #I'm using a log scale just for visualisation as th
         plt.ylabel('Log10 total order expenditure', fontsize=13)
         plt.xlabel('Country', fontsize=13)
         plt.show()
```

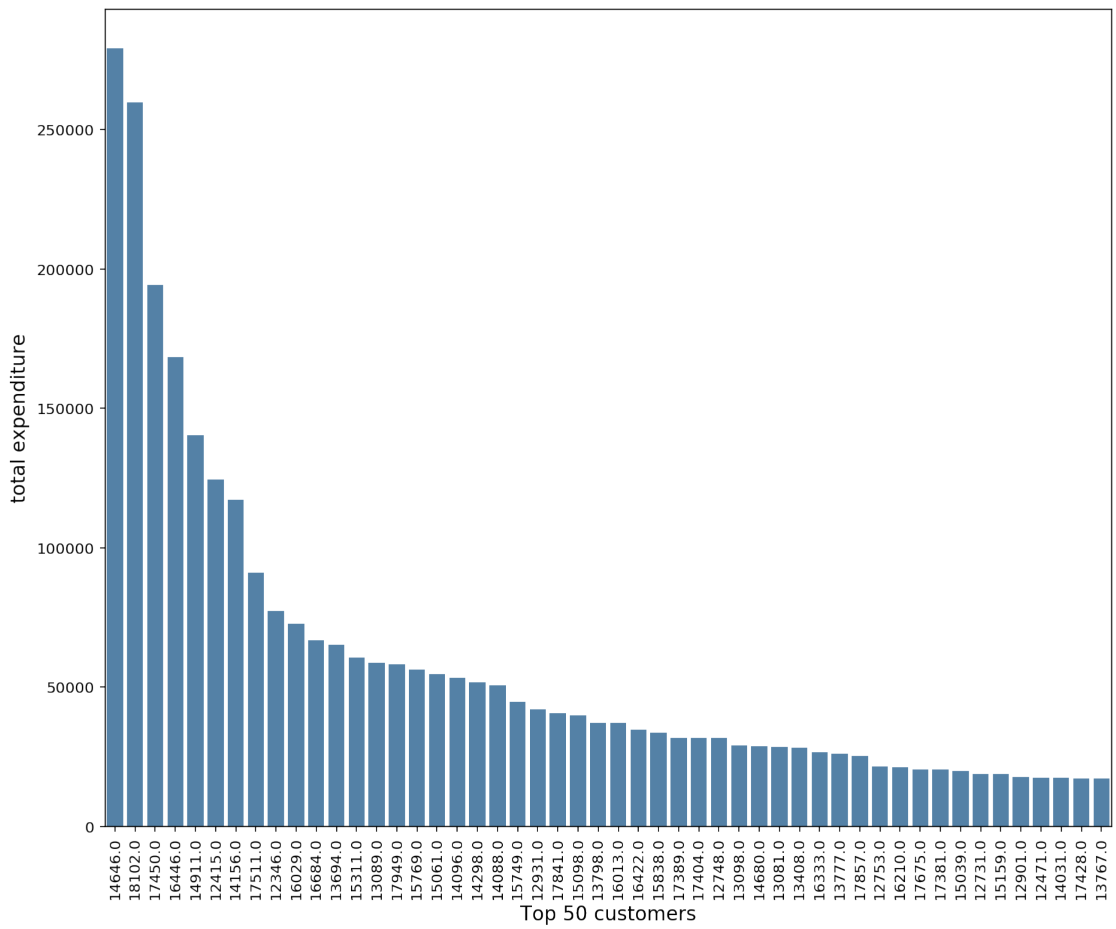


Customers from the United Kingdom spend the most money in addition to placing the most orders.

2.0.3 Grouped per customer

- Identify the top purchasers

```
In [42]: grouped  = data.groupby(['CustomerID']).sum()['total_spend'].sort_values(ascending=Fal
         grouped_top = pd.DataFrame(grouped.head(50))
         grouped_top.reset_index(level=0, inplace=True)
         grouped_top.sort_values('CustomerID', ascending=False)

         f, ax = plt.subplots(figsize=(12, 10))
         plt.xticks(rotation='vertical')
         sns.barplot(x=grouped_top['CustomerID'], y=grouped_top['total_spend'],order=grouped_to
         plt.ylabel('total expenditure', fontsize=13)
         plt.xlabel('Top 50 customers', fontsize=13)
         plt.show()
```



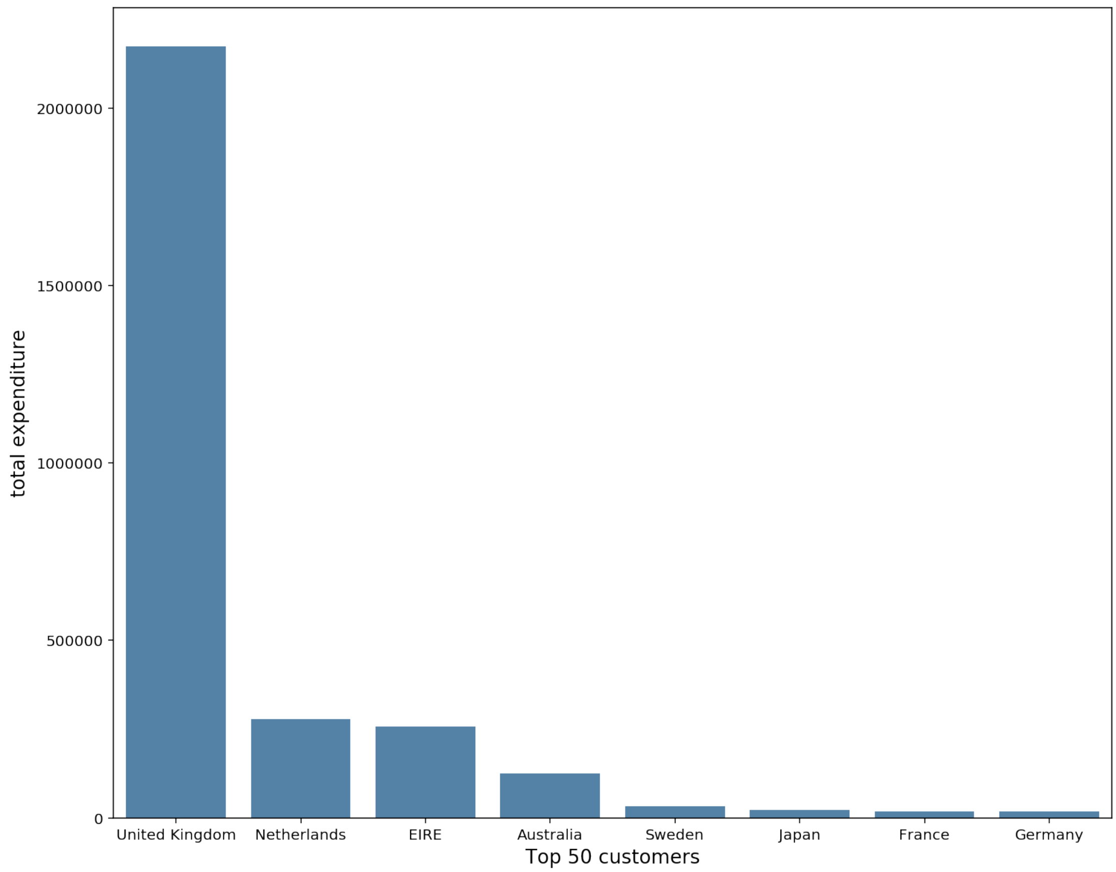
2.0.4 Calculate the top 50 largest baskets, in terms of total spend

```
In [43]: top_50 = data[data['CustomerID'].isin(grouped_top['CustomerID'])]
         top_50.head()
```

```
Out[43]:         InvoiceNo StockCode                          Description  Quantity  \
        540421    581483     23843        PAPER CRAFT , LITTLE BIRDIE     80995
        61619     541431     23166      MEDIUM CERAMIC TOP STORAGE JAR     74215
        421632    573008     84077   WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
        160546    550461     21108  FAIRY CAKE FLANNEL ASSORTED COLOUR      3114
        52711     540815     21108  FAIRY CAKE FLANNEL ASSORTED COLOUR      3114

                     InvoiceDate  UnitPrice  CustomerID         Country  \
        540421    12/9/2011 9:15        2.1     16446.0  United Kingdom
        61619    1/18/2011 10:01        1.0     12346.0  United Kingdom
        421632  10/27/2011 12:26        0.2     12901.0  United Kingdom
        160546   4/18/2011 13:20        2.1     15749.0  United Kingdom
        52711    1/11/2011 12:55        2.1     15749.0  United Kingdom

                              time  total_spend
        540421 2011-12-09 09:15:00     168469.6
        61619  2011-01-18 10:01:00      77183.6
        421632 2011-10-27 12:26:00       1008.0
        160546 2011-04-18 13:20:00       6539.4
        52711  2011-01-11 12:55:00       6539.4
```

2.0.5 Locate the country of origin of the top 50 biggest spenders

```
In [44]: top50_tmp  = top_50.groupby(['Country']).sum()['total_spend'].sort_values(ascending=Fa

         top50_tmp = pd.DataFrame(top50_tmp)
         top50_tmp.reset_index(level=0, inplace=True)
         top50_tmp.sort_values('Country', ascending=False)

         f, ax = plt.subplots(figsize=(12, 10))
         plt.xticks(rotation='horizontal')
         sns.barplot(x=top50_tmp['Country'], y=top50_tmp['total_spend'],order=top50_tmp['Countr
         plt.ylabel('total expenditure', fontsize=13)
         plt.xlabel('Top 50 customers', fontsize=13)
         plt.show()
```



See the Addendum section for further use of this data for Apriori modeling.

2.1 Classify customers based on spend

Collate all the purchases made during a single order to calculate the total order value:

```
In [45]: temp = data.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['total_spend'].sur
         basket_price = temp.rename(columns = {'total_spend':'Basket value'})
```

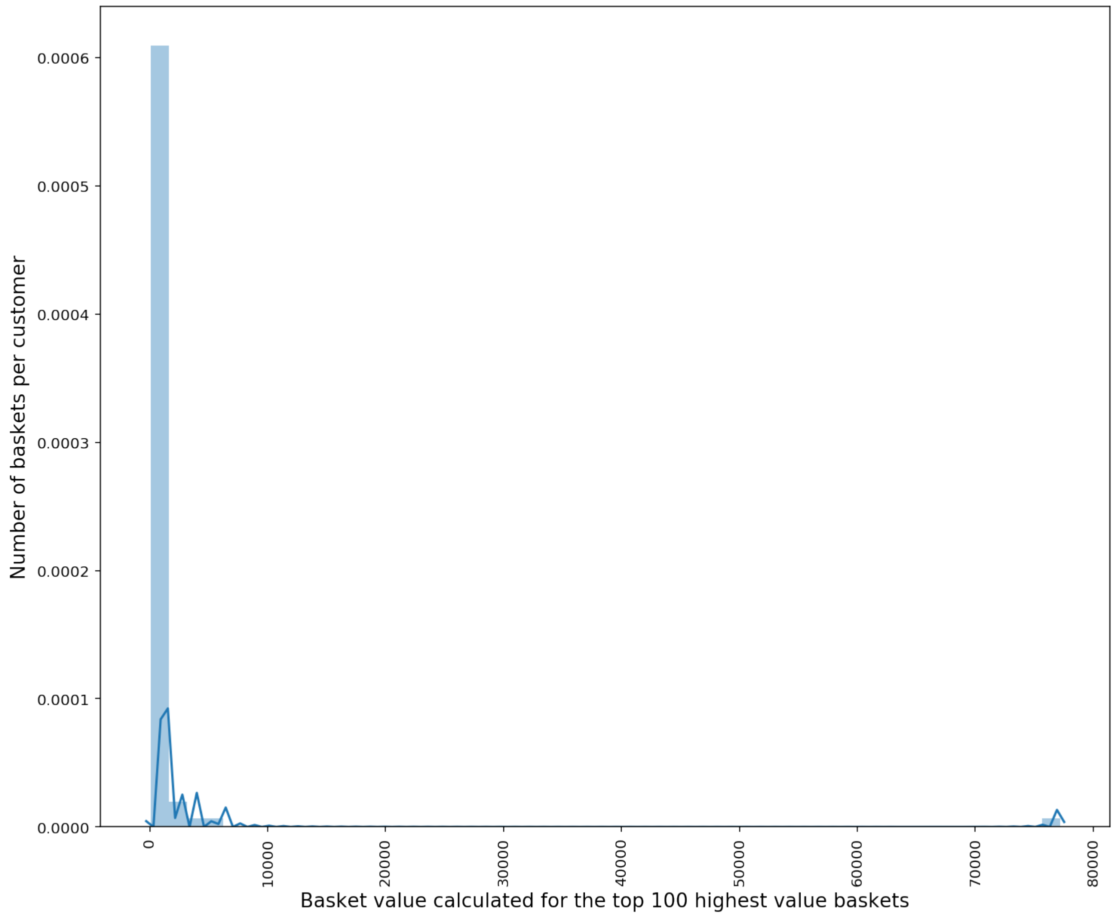
```
In [46]: # top 10 baskets
         basket_price.head(10)
```

```
Out[46]:    CustomerID InvoiceNo  Basket value
         0     12346.0    541431       77183.6
         1     12347.0    537626         711.8
         2     12347.0    542237         475.4
         3     12347.0    549222         636.2
         4     12347.0    556201         382.5
         5     12347.0    562032         584.9
         6     12347.0    573511        1294.3
         7     12347.0    581180         224.8
```

```
8      12348.0    539318         652.8
9      12348.0    541998         187.4
```

In [47]:
```
tmp = basket_price
tmp = pd.DataFrame(tmp)
#tmp.reset_index(level=0, inplace=True)
tmp.sort_values('Basket value', ascending=False)

f, ax = plt.subplots(figsize=(12, 10))
plt.xticks(rotation='vertical')
sns.distplot(tmp['Basket value'][:100])
plt.ylabel('Number of baskets per customer', fontsize=13)
plt.xlabel('Basket value calculated for the top 100 highest value baskets', fontsize=1
plt.show()
```

```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarni
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



The distribution of basket is somewhat bimodal. This histogram of basket values indicates a large number of low total value baskets and a small number of individual orders totaling high value baskets. This observation can be used to bin customers into those spending small amounts, medium amounts, and high value baskets (**note** the bimodal distribution above may cause an imbalance problem for machine learning):

```
In [48]: spend_label = []
         for i in range(0,len(data),1):
             if data['total_spend'].iloc[i] < 5000:
                 spend_label.append(1)
             elif data['total_spend'].iloc[i] >50000:
                 spend_label.append(3)
             else:
                 spend_label.append(2)
```

```
In [49]: data['spend_label'] = spend_label
```

```
In [50]: data.head()
```

```
Out[50]:        InvoiceNo StockCode                         Description  Quantity  \
         540421    581483     23843         PAPER CRAFT , LITTLE BIRDIE     80995
         61619     541431     23166      MEDIUM CERAMIC TOP STORAGE JAR     74215
         502122    578841     84826      ASSTD DESIGN 3D PAPER STICKERS     12540
         421632    573008     84077   WORLD WAR 2 GLIDERS ASSTD DESIGNS      4800
         206121    554868     22197                SMALL POPCORN HOLDER      4300

                      InvoiceDate  UnitPrice  CustomerID         Country  \
         540421    12/9/2011 9:15        2.1     16446.0  United Kingdom
         61619    1/18/2011 10:01        1.0     12346.0  United Kingdom
         502122  11/25/2011 15:57        0.0     13256.0  United Kingdom
         421632  10/27/2011 12:26        0.2     12901.0  United Kingdom
         206121   5/27/2011 10:52        0.7     13135.0  United Kingdom

                                time  total_spend  spend_label
         540421 2011-12-09 09:15:00     168469.6            3
         61619  2011-01-18 10:01:00      77183.6            3
         502122 2011-11-25 15:57:00          0.0            1
         421632 2011-10-27 12:26:00       1008.0            1
         206121 2011-05-27 10:52:00       3096.0            1
```

```
In [51]: # pickle the cleaned dataset
         pickle.dump(data, open( "clean_data.pkl", "wb" ))
```

2.2 Clustering

2.2.1 Group by customerID

- group by CustomerID, together with sum or number of items (quantity) and the unit price

In [52]:
```
data_grouped = data.groupby('CustomerID')
data_cluster=pd.DataFrame(columns=['Quantity','UnitPrice', 'total_spend', 'country',
count=0
```

In [56]:
```
#data_grouped.head(5)
```

In [54]:
```
for k,v in (data_grouped):
    data_cluster.loc[count] = [(v['Quantity'].sum()), v['UnitPrice'].sum(), v['total_s
    count+=1

# Applying K-Means Clustering Algorithm to quantity, and total spend
X = data_cluster.iloc[:, [0, 2]].values
```

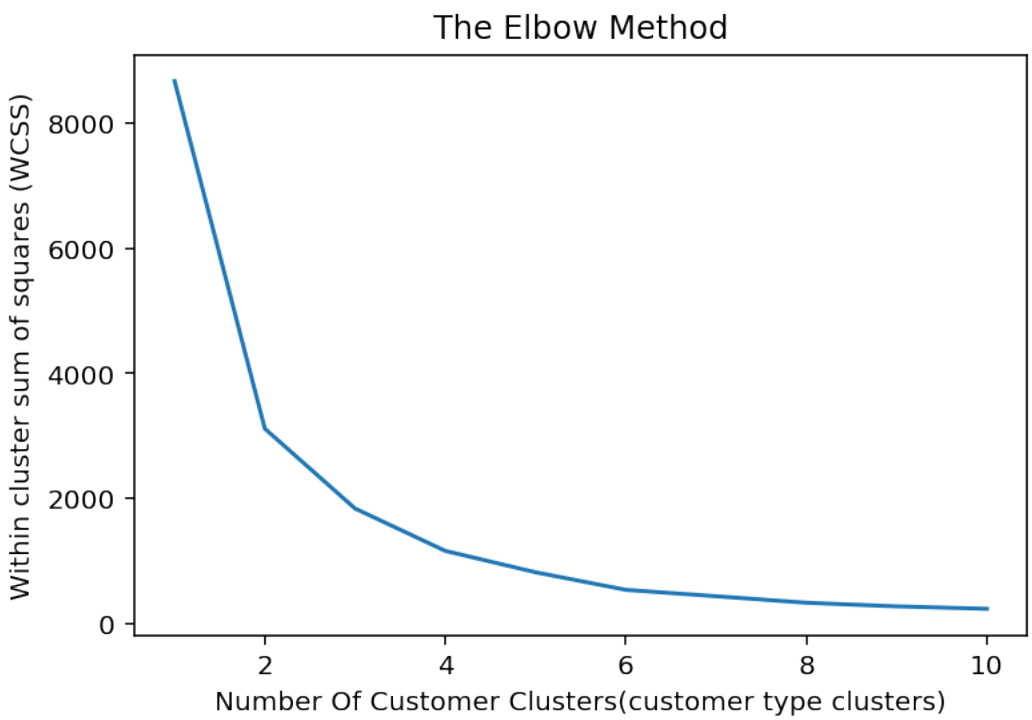
In [55]:
```
data_cluster.head()
```

Out[55]:
```
  Quantity  UnitPrice  total_spend  \
0    74215        1.0      77183.6
1     2458      481.2       4310.0
2     2332       18.7       1437.2
3      630      305.1       1457.5
4      196       25.3        294.4

                                             country  CustomerID
0  61619    United Kingdom
Name: Country, dtype: ...     12346.0
1  148290    Iceland
428974    Iceland
148303    ...      12347.0
2  70051     Finland
70052     Finland
70054     ...      12348.0
3  485568    Italy
485569    Italy
485554    Ital...     12349.0
4  80327    Norway
80339    Norway
80338    Norwa...     12350.0
```

In [56]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X= sc_X.fit_transform(X)
#Using the Elbow method to find the optimum number of clusters
from sklearn.cluster import KMeans
wcss = [] #Within cluster sum of squers(Inertia)

#n_clusters is no.of clusters given by this method,
#k-means++ is an random initialization methods for centriods to avoid random intializ
#max_iter is max no of iterations defined when k-means is running
```

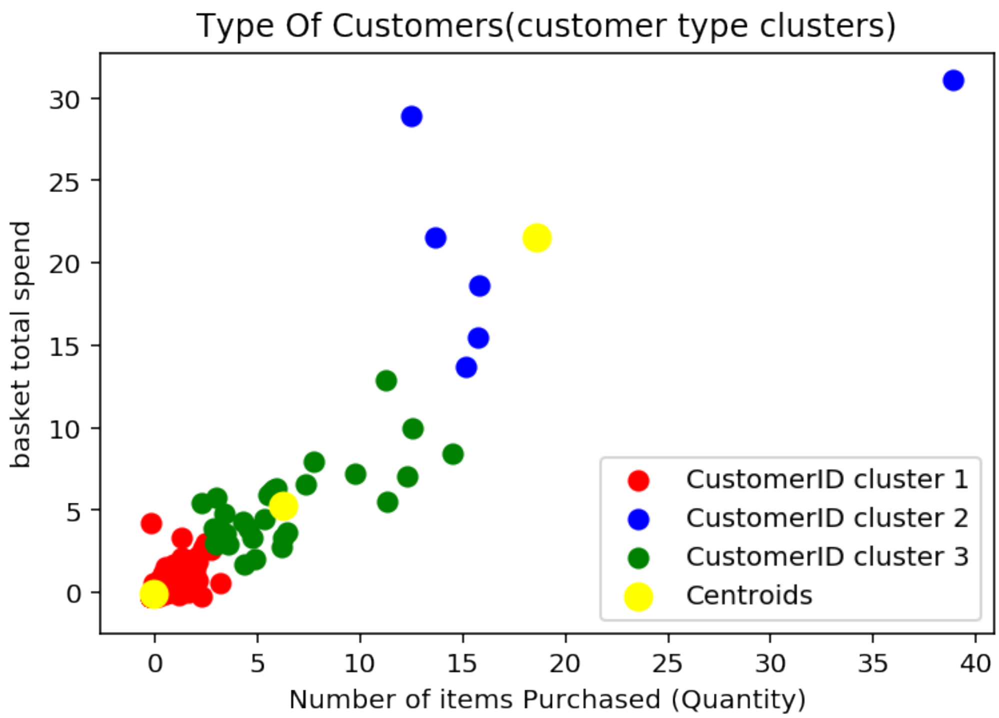
```
#n_init is no of times k-means will run with different initial centroids

for i in range(1,11): #From 2-10 doing multiple random initializations can make a hug
    kmeans = KMeans(n_clusters = i, init ='k-means++',max_iter=300,n_init=10)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11) , wcss)
plt.title('The Elbow Method')
plt.xlabel('Number Of Customer Clusters(customer type clusters)')
plt.ylabel('Within cluster sum of squares (WCSS)')
plt.show()
```

```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:475: DataCo
  warnings.warn(msg, DataConversionWarning)
/Users/scheckley/miniconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:475: DataCo
  warnings.warn(msg, DataConversionWarning)
```



In [57]:
```
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 3, init = 'k-means++')
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 50, c = 'red', label = 'Cus
```

```
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 50, c = 'blue', label = 'Cus
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 50, c = 'green', label = 'Cu
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c =
plt.title('Type Of Customers(customer type clusters)')
plt.xlabel('Number of items Purchased (Quantity)')
plt.ylabel('basket total spend')
plt.legend()
plt.show()
```



Clustering appears to separate the customers based on numbers of items and total spend, which would be expected

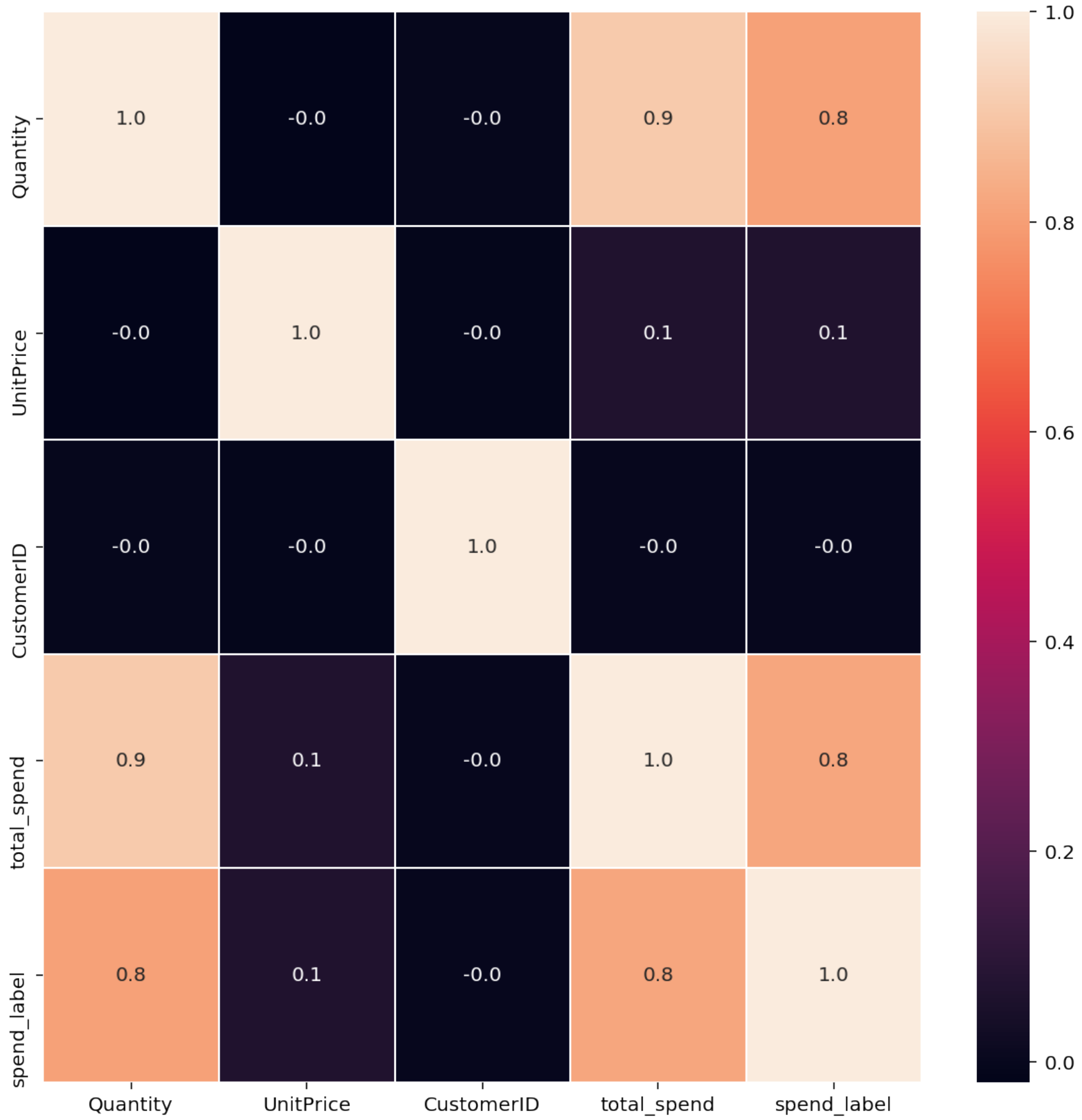
2.3 Task 4 - Modelling

2.3.1 Machine learning data preparation

```
In [3]: data = pickle.load(open( "clean_data.pkl", "rb" ))
```

A cursory examination of correlation to identify potentially problematic variables from the model training dataset.

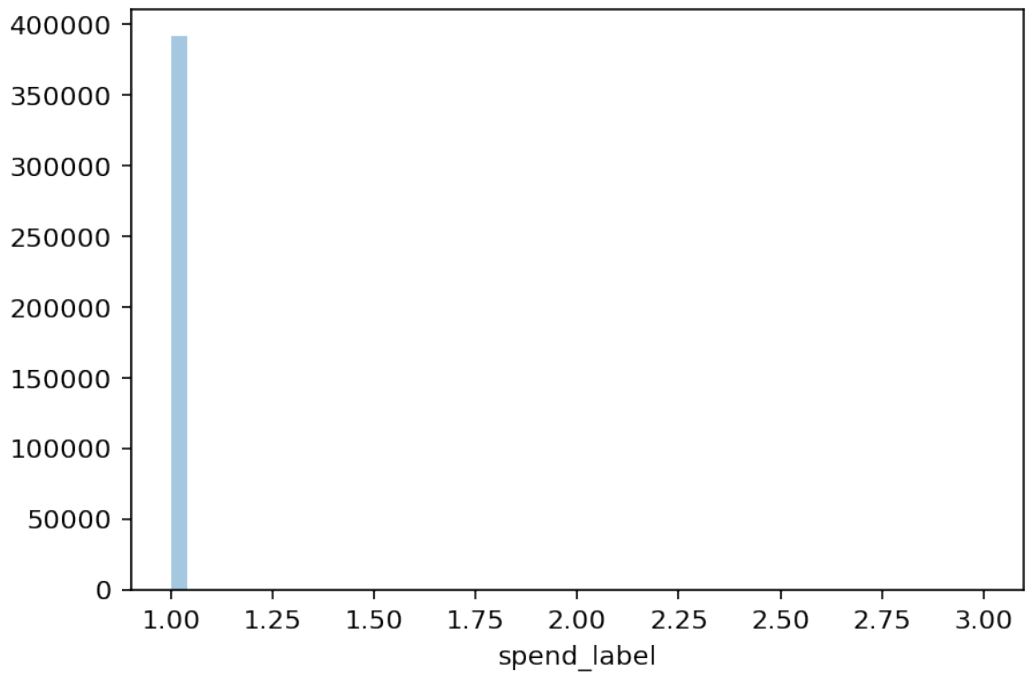
```
In [11]: # heat map to look for correlation
         f,ax = plt.subplots(figsize=(10, 10))
         sns.heatmap(data.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax);
```



Unsurprisingly, total spend correlates with quantity and the spend_label. Potentially, quantity or total spend may have to be removed for training.

```
In [12]: sns.distplot(data['spend_label'],kde = False)
         plt.show()
```

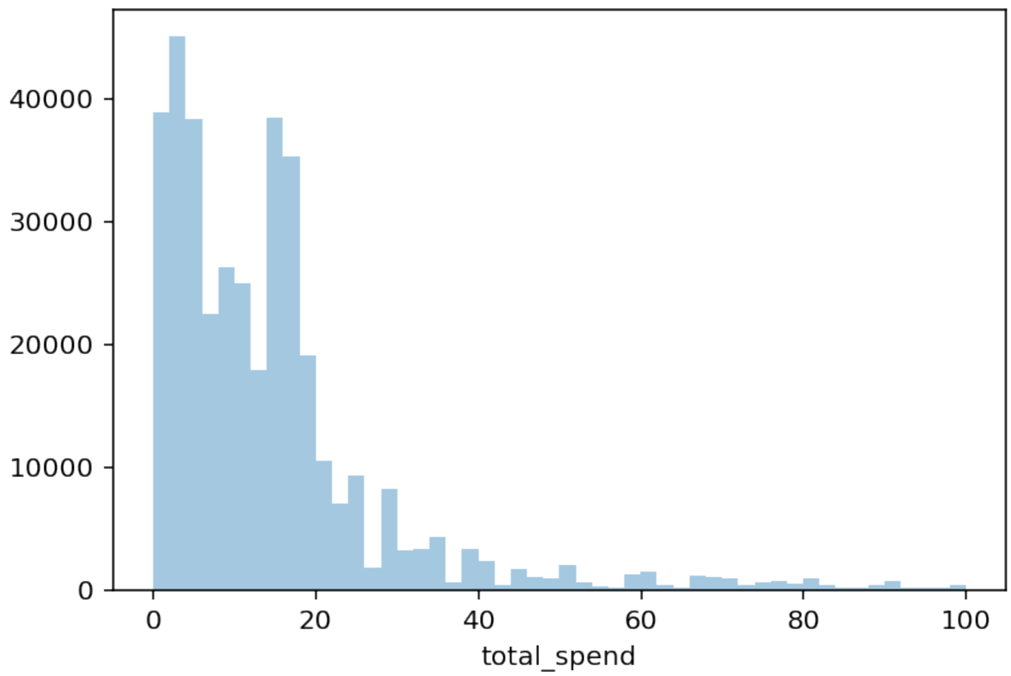
```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarni
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



This data is very imbalanced. For the purposes of this investigation the lower value baskets will be used for prediction.

Below I will identify a range suitable for binning:

```
In [4]: data2 = data[(data['total_spend'] >0) & (data['total_spend'] < 100)]

In [14]: sns.distplot(data2['total_spend'],kde = False)
         plt.show()

/Users/scheckley/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarni
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [5]: spend_label = []
        for i in range(0,len(data2),1):
            if data2['total_spend'].iloc[i] < 10:
                spend_label.append(0)
            elif data2['total_spend'].iloc[i] >40:
                spend_label.append(2)
            else:
                spend_label.append(1)
```

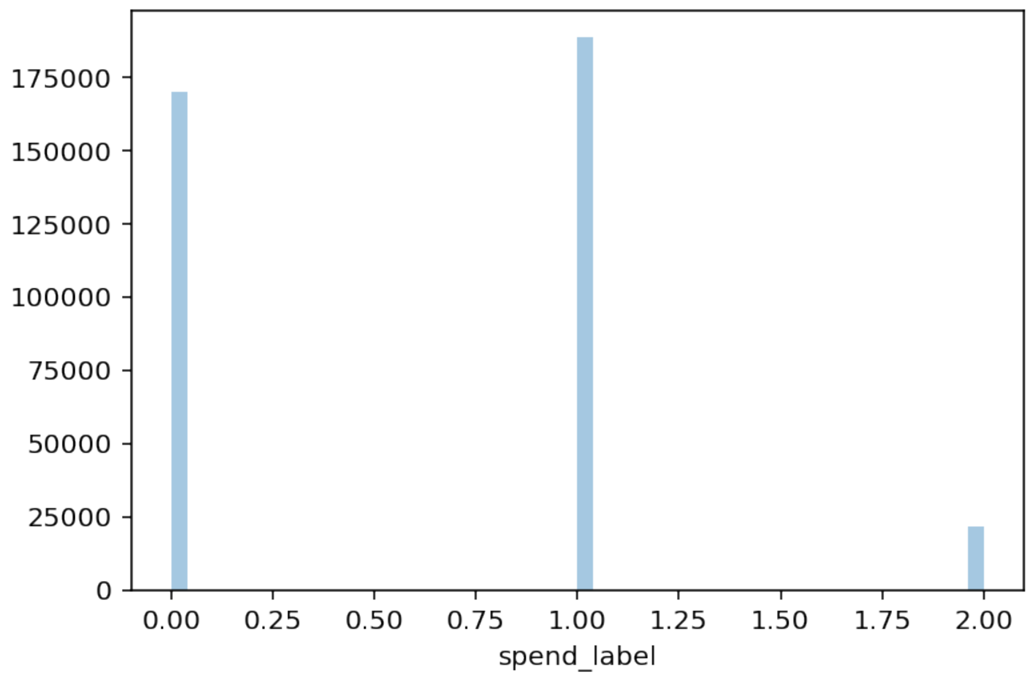
Numerical labels represent 0 - low value baskets, 1 - medium value baskets, 3 - higher value baskets.

```
In [6]: data2['spend_label'] = spend_label

/Users/scheckley/miniconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """Entry point for launching an IPython kernel.
```

```
In [7]: data2.shape # there is still a reasonably large sized data set to work with

Out[7]: (379870, 11)
```

```
In [62]: sns.distplot(data2['spend_label'],kde = False)
         plt.show()
```

```
/Users/scheckley/miniconda3/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarni
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



The data is still unbalanced in terms of representation from high value baskets, but more balanced than the full dataset

2.3.2 Create dummy variables from the string columns

```
In [8]: # encode the description label
        cols_to_transform = ['Description']
        type_hash = pd.get_dummies(data=data2['Description'])
        type_hash2 = pd.get_dummies(data=data2['Country'])
```

```
In [9]: learning_data = pd.concat([data2, type_hash, type_hash2], axis=1)
```

```
In [10]: # drop the columns that have been now been replaced and that are not required
         droplist = ['StockCode','InvoiceDate','InvoiceNo','UnitPrice','Description','time','Co
         learning_data = learning_data.drop(droplist, axis=1)
```

```
In [11]: learning_data = learning_data.reset_index(drop=True)
```

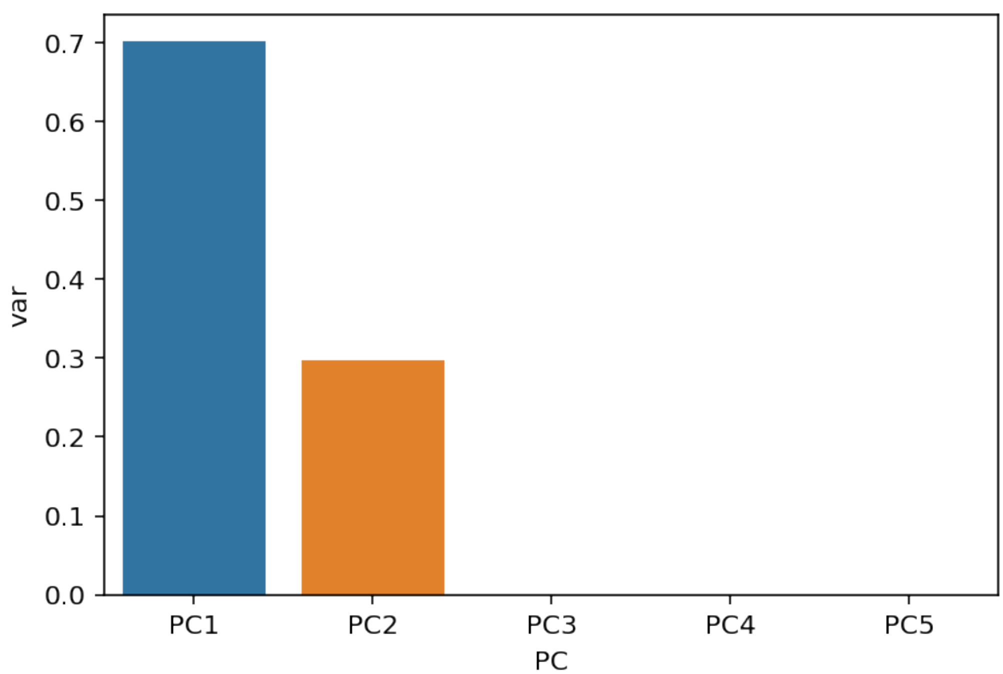
```
In [12]: xdata = learning_data.copy()
         ydata = learning_data['spend_label']
```

```
In [149]: pca = decomposition.PCA(n_components=5)
          pc = pca.fit_transform(xdata[:100000]) #PCA is being performed on the 1st 100000 dat

          pc_df = pd.DataFrame(data = pc ,
                  columns = ['PC1', 'PC2','PC3','PC4', 'PC5'])
          pc_df.head()
```

Out[149]:

	PC1	PC2	PC3	PC4	PC5
0	1383.4	759.2	-0.5	0.7	0.2
1	1035.6	561.3	-0.4	0.4	0.2
2	1035.6	561.3	-0.4	0.4	0.2
3	754.4	446.2	-0.3	0.3	0.1
4	749.9	384.4	-0.4	0.5	0.1

```
In [150]: # plot the variance
          df = pd.DataFrame({'var':pca.explained_variance_ratio_,
                       'PC':['PC1','PC2','PC3','PC4', 'PC5']})
          sns.barplot(x='PC',y="var",
                     data=df);
```



- From this cursory clustering analysis of the dataset, the majority of variance in the model is in the 1st and 2nd principle components.

2.4 Task 4 - Machine learning

```
In [13]: from sklearn.metrics import confusion_matrix,precision_recall_curve,auc,roc_auc_score
         from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
```

```
from sklearn.model_selection import  cross_val_score
from sklearn.externals import joblib #for saving the trained model
```

For the purposes of this work I will select a 70/30 split - 70% training data and 30% test data. This approach does not use a validation set however it provides a large dataset for training and testing. Cross validation of the training data set will also be used during model training.

```
In [14]: from sklearn.model_selection import train_test_split
         xtrain, xtest, ytrain, ytest = train_test_split(learning_data,ydata,test_size = 0.3, r
```

```
In [15]: xtrain.head()
```

```
Out[15]:         Quantity  total_spend  spend_label  10 COLOUR SPACEBOY PEN  \
         310253         2          3.9            0                       0
         274741         2          5.9            0                       0
         361291         1          2.1            0                       0
         99147         12         35.4            1                       0
         290412         2          4.2            0                       0

                 12 COLOURED PARTY BALLOONS  12 DAISY PEGS IN WOOD BOX  \
         310253                           0                          0
         274741                           0                          0
         361291                           0                          0
         99147                            0                          0
         290412                           0                          0

                 12 EGG HOUSE PAINTED WOOD  12 HANGING EGGS HAND PAINTED  \
         310253                          0                             0
         274741                          0                             0
         361291                          0                             0
         99147                           0                             0
         290412                          0                             0

                 12 IVORY ROSE PEG PLACE SETTINGS  12 MESSAGE CARDS WITH ENVELOPES  \
         310253                                 0                                0
         274741                                 0                                0
         361291                                 0                                0
         99147                                  0                                0
         290412                                 0                                0

                 12 PENCIL SMALL TUBE WOODLAND  12 PENCILS SMALL TUBE RED RETROSPOT  \
         310253                              0                                    0
         274741                              0                                    0
         361291                              0                                    0
         99147                               0                                    0
         290412                              0                                    0

                 12 PENCILS SMALL TUBE SKULL  12 PENCILS TALL TUBE POSY  \
         310253                            0                          0
```

```
274741                          0                          0
361291                          0                          0
99147                           0                          0
290412                          0                          0

        12 PENCILS TALL TUBE RED RETROSPOT  12 PENCILS TALL TUBE SKULLS  \
310253                                   0                            0
274741                                   0                            0
361291                                   0                            0
99147                                    0                            0
290412                                   0                            0

        12 PENCILS TALL TUBE WOODLAND  12 PINK HEN+CHICKS IN BASKET  \
310253                              0                             0
274741                              0                             0
361291                              0                             0
99147                               0                             0
290412                              0                             0

        12 PINK ROSE PEG PLACE SETTINGS  12 RED ROSE PEG PLACE SETTINGS  \
310253                                0                               0
274741                                0                               0
361291                                0                               0
99147                                 0                               0
290412                                0                               0

        15 PINK FLUFFY CHICKS IN BOX  15CM CHRISTMAS GLASS BALL 20 LIGHTS  \
310253                             0                                    0
274741                             0                                    0
361291                             0                                    0
99147                              0                                    0
290412                             0                                    0

        16 PC CUTLERY SET PANTRY DESIGN  16 PIECE CUTLERY SET PANTRY DESIGN  \
310253                                0                                   0
274741                                0                                   0
361291                                0                                   0
99147                                 0                                   0
290412                                0                                   0

        18PC WOODEN CUTLERY SET DISPOSABLE  2 DAISIES HAIR COMB  \
310253                                   0                    0
274741                                   0                    0
361291                                   0                    0
99147                                    0                    0
290412                                   0                    0

        2 PICTURE BOOK EGGS EASTER BUNNY  2 PICTURE BOOK EGGS EASTER CHICKS  \
```

```
310253                                  0                                   0
274741                                  0                                   0
361291                                  0                                   0
99147                                   0                                   0
290412                                  0                                   0

        2 PICTURE BOOK EGGS EASTER DUCKS  20 DOLLY PEGS RETROSPOT  \
310253                                  0                         0
274741                                  0                         0
361291                                  0                         0
99147                                   0                         0
290412                                  0                         0

        200 BENDY SKULL STRAWS  200 RED + WHITE BENDY STRAWS  \
310253                        0                             0
274741                        0                             0
361291                        0                             0
99147                         0                             0
290412                        0                             0

        3 BIRDS CANVAS SCREEN  3 BLACK CATS W HEARTS BLANK CARD  \
310253                      0                                 0
274741                      0                                 0
361291                      0                                 0
99147                       0                                 0
290412                      0                                 0

        3 DRAWER ANTIQUE WHITE WOOD CABINET  3 GARDENIA MORRIS BOXED CANDLES  \
310253                                    0                                0
274741                                    0                                0
361291                                    0                                0
99147                                     0                                0
290412                                    0                                0

        3 HEARTS HANGING DECORATION RUSTIC  3 HOOK HANGER MAGIC GARDEN  \
310253                                   0                            0
274741                                   0                            0
361291                                   0                            0
99147                                    0                            0
290412                                   0                            0

        3 HOOK PHOTO SHELF ANTIQUE WHITE  3 PIECE SPACEBOY COOKIE CUTTER SET  \
310253                                 0                                   0
274741                                 0                                   0
361291                                 0                                   0
99147                                  0                                   0
290412                                 0                                   0
```

```
        3 PINK HEN+CHICKS IN BASKET  3 RAFFIA RIBBONS 50'S CHRISTMAS  \
310253                            0                                0
274741                            0                                0
361291                            0                                0
99147                             0                                0
290412                            0                                0

        3 RAFFIA RIBBONS VINTAGE CHRISTMAS  3 ROSE MORRIS BOXED CANDLES  \
310253                                   0                            0
274741                                   0                            0
361291                                   0                            0
99147                                    0                            0
290412                                   0                            0

        3 STRIPEY MICE FELTCRAFT  3 TIER CAKE TIN GREEN AND CREAM  \
310253                         0                                0
274741                         0                                0
361291                         0                                0
99147                          0                                0
290412                         0                                0

        3 TIER CAKE TIN RED AND CREAM  3 TIER SWEETHEART GARDEN SHELF  \
310253                              0                               0
274741                              0                               0
361291                              0                               0
99147                               0                               0
290412                              0                               0

        3 TRADITIONAL COOKIE CUTTERS  SET  3 TRADITIONAl BISCUIT CUTTERS  SET  \
310253                                  0                                   0
274741                                  0                                   0
361291                                  0                                   0
99147                                   0                                   0
290412                                  0                                   0

           ...       ZINC HERB GARDEN CONTAINER  ZINC METAL HEART DECORATION  \
310253     ...                                0                            0
274741     ...                                0                            0
361291     ...                                0                            0
99147      ...                                0                            0
290412     ...                                0                            0

        ZINC PLANT POT HOLDER  ZINC STAR T-LIGHT HOLDER  \
310253                      0                         0
274741                      0                         0
361291                      0                         0
99147                       0                         0
290412                      0                         0
```

```
        ZINC SWEETHEART SOAP DISH  ZINC SWEETHEART WIRE LETTER RACK  \
310253                          0                                 0   
274741                          0                                 0   
361291                          0                                 0   
99147                           0                                 0   
290412                          0                                 0   

        ZINC T-LIGHT HOLDER STAR LARGE  ZINC T-LIGHT HOLDER STARS LARGE  \
310253                               0                                0   
274741                               0                                0   
361291                               0                                0   
99147                                0                                0   
290412                               0                                0   

        ZINC T-LIGHT HOLDER STARS SMALL  ZINC TOP  2 DOOR WOODEN SHELF  \
310253                                0                              0   
274741                                0                              0   
361291                                0                              0   
99147                                 0                              0   
290412                                0                              0   

        ZINC WILLIE WINKIE  CANDLE STICK  ZINC WIRE KITCHEN ORGANISER  \
310253                                 0                            0   
274741                                 0                            0   
361291                                 0                            0   
99147                                  0                            0   
290412                                 0                            0   

        ZINC WIRE SWEETHEART LETTER TRAY  Australia  Austria  Bahrain  \
310253                                 0          0        0        0   
274741                                 0          0        0        0   
361291                                 0          0        0        0   
99147                                  0          0        0        0   
290412                                 0          0        0        0   

        Belgium  Brazil  Canada  Channel Islands  Cyprus  Czech Republic  \
310253        0       0       0                0       0               0   
274741        0       0       0                0       0               0   
361291        0       0       0                0       0               0   
99147         0       0       0                0       0               0   
290412        0       0       0                0       0               0   

        Denmark  EIRE  European Community  Finland  France  Germany  Greece  \
310253        0     0                   0        0       0        0       0   
274741        0     0                   0        0       0        0       0   
361291        0     0                   0        0       0        0       0   
99147         0     0                   0        0       0        0       0   
```

```
290412        0      0                    0         0        0         0        0

        Iceland  Israel  Italy  Japan  Lebanon  Lithuania  Malta  Netherlands  \
310253        0       0      0      0        0          0      0            0
274741        0       0      0      0        0          0      0            0
361291        0       0      0      0        0          0      0            0
99147         0       0      0      0        0          0      0            0
290412        0       0      0      0        0          0      0            0

        Norway  Poland  Portugal  RSA  Saudi Arabia  Singapore  Spain  Sweden  \
310253       0       0         0    0             0          0      0       0
274741       0       0         0    0             0          0      0       0
361291       0       0         0    0             0          0      0       0
99147        0       0         0    0             0          0      0       0
290412       0       0         0    0             0          0      0       0

        Switzerland  USA  United Arab Emirates  United Kingdom  Unspecified
310253            0    0                     0               1            0
274741            0    0                     0               1            0
361291            0    0                     0               1            0
99147             0    0                     0               1            0
290412            0    0                     0               1            0

[5 rows x 3887 columns]
```

Some helper functions for visualizing model output

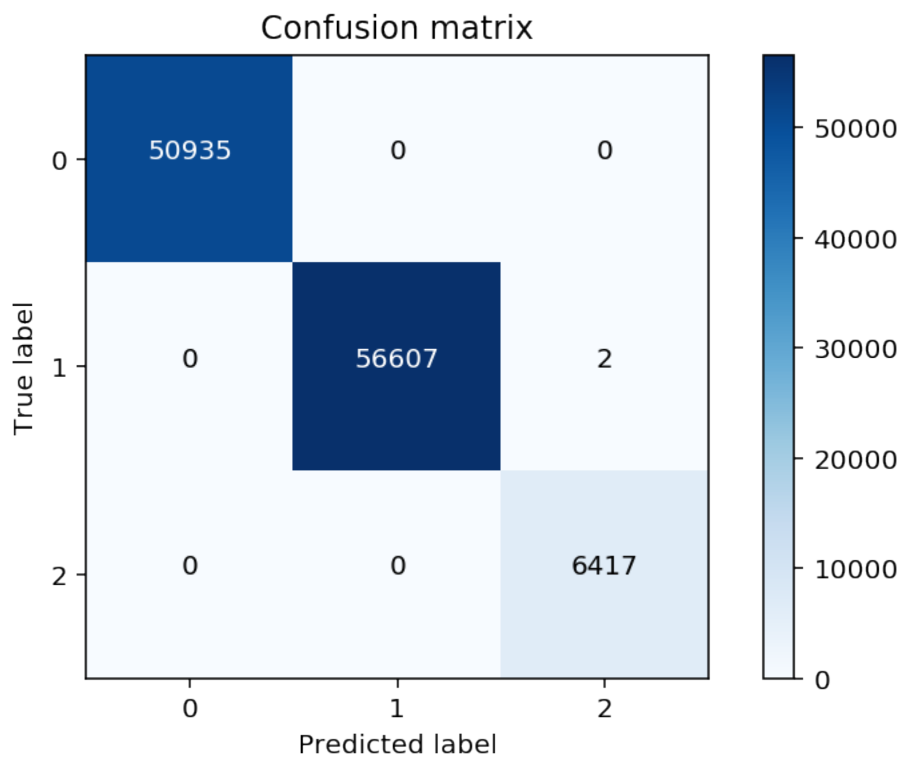
```
In [16]: import itertools

         # confusion matrix plotting function
         def plot_confusion_matrix(cm, classes,
                                   normalize=False,
                                   title='Confusion matrix',
                                   cmap=plt.cm.Blues):
             """
             This function prints and plots the confusion matrix.
             Normalization can be applied by setting `normalize=True`.
             """
             plt.imshow(cm, interpolation='nearest', cmap=cmap)
             plt.title(title)
             plt.colorbar()
             tick_marks = np.arange(len(classes))
             plt.xticks(tick_marks, classes, rotation=0)
             plt.yticks(tick_marks, classes)

             if normalize:
                 cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
                 #print("Normalized confusion matrix")
```

```
    else:
        1#print('Confusion matrix, without normalization')

    #print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

2.4.1 Naive Bayes

Naive bayes was chosen to build the 1st iteration of the model because it is a fast algorithm and requires no hyperparameters. This method will provide an indication of whether or not a model can be built using this dataset without using more computationally expensive methods.

```
In [73]: from sklearn.naive_bayes import GaussianNB #choose model class - done
         nb_model = GaussianNB() #instantiate the model - done (GaussianNB has no hyperparamet
         nb_model.fit(xtrain, ytrain) #fit the model to the data

Out[73]: GaussianNB(priors=None)

In [74]: # Use the trained model to predict on the test data
         predictions = list(nb_model.predict(xtest))
         print('Accuracy score:', metrics.accuracy_score(ytest, predictions))

Accuracy score: 0.9999824501364502


In [75]: dat = confusion_matrix(ytest, predictions)

         plot_confusion_matrix(
             dat, classes=[0,1,2], title='Confusion matrix')
         plt.show()
```




```
In [161]: xval_score = cross_val_score(nb_model, xtrain, ytrain, cv=10, n_jobs=-1).mean() #10-

In [162]: xval_score

Out[162]: 0.9999736753036423

In [77]: #joblib.dump(nb_model, 'nb_model.pkl')

Out[77]: ['nb_model.pkl']
```

The naive bayes model is 99% accurate in predicting how likely a customer is to be purchasing a low, medium, or high value basket.

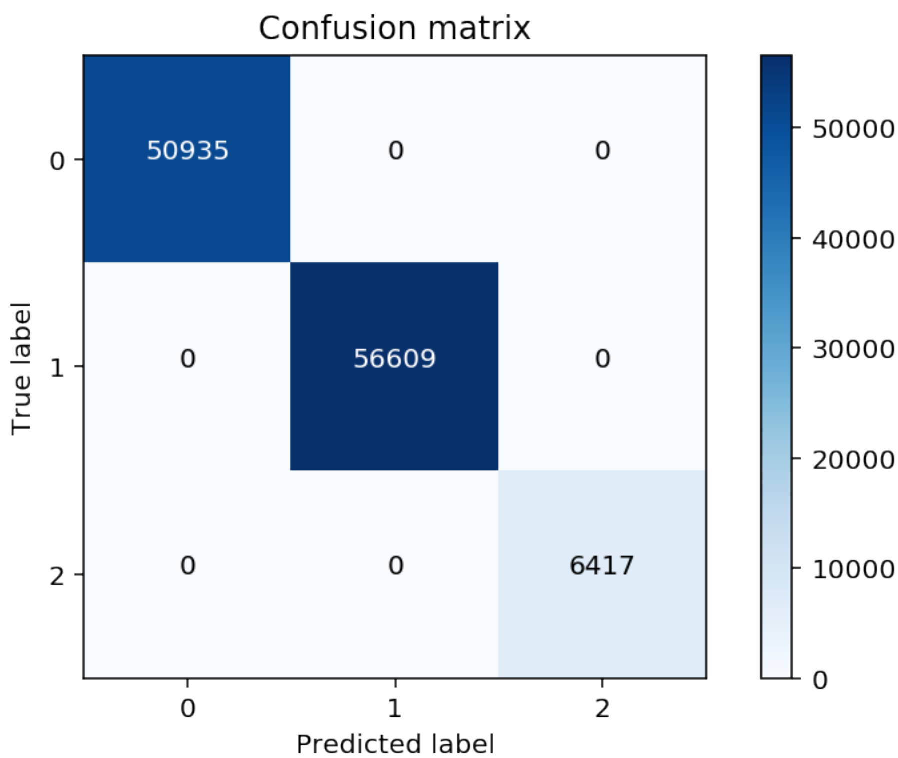
2.4.2 LightGBM

Microsoft's LightGBM algorithm was selected to build a 2nd model. This algorithm is relatively new and is designed to be faster and more accurate than XGBoost [1,2]

```
In [18]: import lightgbm as lgb

In [19]: train_data=lgb.Dataset(xtrain,label=ytrain)
```

Due to limited time and compute resource I will use 5-fold cross validation when training the LightGBM model.

```
In [ ]: params = {'task': 'train',
            'boosting_type': 'gbdt',
            'objective': 'multiclass',
            'num_class':3,
            'metric': 'multi_logloss',
            'learning_rate': 0.05,
            'max_depth': 7,
            'num_leaves': 17,
            'feature_fraction': 0.4,
            'bagging_fraction': 0.6,
            'bagging_freq': 17}

        lgb_cv = lgb.cv(params, train_data, num_boost_round=10000, nfold=5, shuffle=True, stra

        nround = lgb_cv['multi_logloss-mean'].index(np.min(lgb_cv['multi_logloss-mean']))
        #print(nround)

In [45]: model = lgb.train(params, train_data, num_boost_round=nround)

In [46]: #predicting on test set
         ypred=model.predict(xtest)

In [47]: predictions = []

         for x in ypred:
             predictions.append(np.argmax(x))

In [48]: accuracy = accuracy_score(ytest, predictions)
         print("Accuracy: %.2f%%" % (accuracy * 100.0))

Accuracy: 100.00%


In [49]: dat = confusion_matrix(ytest, predictions)

         plot_confusion_matrix(
             dat, classes=[0,1,2], title='Confusion matrix')
         plt.show()
```




```
In [51]: joblib.dump(model, 'lgbm_model.pkl')

Out[51]: ['lgbm_model.pkl']
```

The LightGBM model is 100% accurate at predicting how likely a customer is to be purchasing a low, medium, or high value basket. This algorithm is superior to Naive bayes but requires significantly higher compute resource to achieve a 1% improvement.

3 Task 5 - Conclusions and further work

The detail and scientific rigor of the data analysis and modeling performed for this assignment was limited by the short amount of time allocated for delivery of this report. However, given the constraints, the data analysis section of this work identified a number of useful metrics in the data, including:

- The frequency of invoices throughout the year.
- The number of orders placed per customer, and per country.
- The value of baskets per customer and per country.
- The contents of the highest spending customer baskets.

This data would enable the business to predict required staffing levels to pick and dispatch items and the most common global location of customers.

The observation of per customer baskets and their value enabled the identification and classification of customers based on their level of spend. This information has enabled the construction of a proof of concept machine learning model to predict new customers level of spend. This model could be used to predict icome levels for the business and combined with apriori modeling (and further work using NLP), likely combinations of basket items.

Further work

Further work is needed on feature engineering for this dataset. It was beyond the scope of this assignment to fully explore the time series component of the data. In addition, further natural language processing (NLP) work is required to classify item descriptions for clustering.

Combining the NLP work with the time series analysis would enable a higher resolution observation of sales of individual items throughout the year, rather than the high level observation of the number of invoices received.

The NLP classfication of items into groups would also provide an additional classifier to cluster customers with the items they purchase, and also enable a model to be trained to predict which items a customer is likely to purchase in a basket, as well changes in shopping trends during the year (i.e, the identification and prediction of seasonality).

To obviate some of the issues with not having predicted clusters and trends in basket contents, apriori modeling was employed as an additional methodology as part of this report, which enables a less computationally expensive estimate of the correlation between purchased items, enabling prediction of items that may be purchased in combination. Using this method, it was possible to make a prediction for which items the top spending customer would purchase which could assist with stocking bulk order or difficult to source/rare items for high value customers. In addition, baskets can be grouped per country and regional trends in basket contents could be predicted using this modeling approach.

The dataset as provided for this assignment contains an imbalance in that there is a bimodal distribution with the majority of baskets clustering at a lower value than a small number of high value baskets. For model training, the highest value baskets were removed as "outliers", however this has removed the most valued customers from the dataset. An alternative approach to removing the high value baskets would be to use oversampling methods to create synthetic data for higher spend baskets. This approach is computationally expensive however, and beyond the scope of this assignment. A similar imbalance exists in the location data, with the majority of purchases from the UK. The UK could be removed from the dataset in order to model other countries, however an alternative approach could be to build country specific models.

2 machine learning models were selected for this assignment, the 1st was the Naive bayes algorithm as this algorithm is fast in terms of training models on large data sets and also requires no hyper-parameters therefore no additional parameter optimization steps are required. This method also provides a fast and efficient method of testing the suitability of a dataset for modeling, and if problems are encountered with Naive bayes it may be assumed that more sophisticated algorithms may also struggle to find a solution for the data being used. In this case, the naive bayes algorithm performed well and produced a model with 99% accuracy, in terms of predicting the likely basket value of a customer. The 2nd algorithm chosen was the LightGBM algorithm. This algorithm was chosen as it has proven to be fast in terms of training on large datasets and provides robust solutions compated with current top performing algorithms such as xgboost and random forest. Due to the time constraints placed on this assignment LightGBM provided a computationally fast and reliable solution using an algorithm with greater flexibility in terms or fitting complex relationships in large, multivariate datasets, and take advantage of multiple cpu cores and hardware acceleration "out of the box".

Further work should be performed using the random forest algorithm to perform feature analysis. This analysis would inform on which variables contribute most to the variation in the data, reduce the number of variables required to fit the model, and provide additonal information on top of the principle component analysis. Further to this issue, feature engineering using dummy variables, as was performed with the country data generates wide, sparse matrices which add additional computational cost and potential imbalance issues to the modeling process. One possible solution to this would be to sue one-hot encoding instead of dummy variables.

Further work could be performed by applying hyper-parameter tuning to the LightGBM algorithm using methods contained in Python packages such as hyperopt, however this would require the procurement of high performance computing resource which was beyond the scope of this assignment.

3.1 Addendum - Apriori modeling example

Apriori modeling requires no feature engineering, machine learning, or significant compute time and is an alternative approach to machine learning for this form of exercise. [4]

Modeling will be applied to the top customer identified in the earlier data analysis as a proof of concept recommender system.

```
In [691]: from mlxtend.frequent_patterns import apriori
          from mlxtend.frequent_patterns import association_rules
```

3.1.1 Generate a basket for the top spending customer identified from the dataset

As an example, the customerid corresponding with the highest spend was identified during the data analysis and the customer's basket extracted for further analysis. This method could optimize the prediction of stock levels to facilitate customers who make large, bulk orders or order expensive or difficult to source items ahead of schedule.

```
In [692]: basket = top_basket.groupby(['InvoiceNo', 'Description'])['Quantity'].sum().unstack(

In [693]: basket.head()

Out[693]: Description  12 DAISY PEGS IN WOOD BOX  12 EGG HOUSE PAINTED WOOD  \
          InvoiceNo
          536557                             0.0                        0.0
          536984                             0.0                        0.0
          537405                             0.0                        0.0
          538163                             0.0                        0.0
          538866                             0.0                        0.0

          Description  12 IVORY ROSE PEG PLACE SETTINGS  \
          InvoiceNo
          536557                                    0.0
          536984                                    0.0
          537405                                    0.0
          538163                                    0.0
          538866                                    0.0
```

```
Description  12 MESSAGE CARDS WITH ENVELOPES  12 PENCIL SMALL TUBE WOODLAND  \
InvoiceNo
536557                                 0.0                              0.0
536984                                 0.0                              0.0
537405                                 0.0                              0.0
538163                                 0.0                              0.0
538866                                 0.0                              0.0

Description  12 PENCILS SMALL TUBE RED RETROSPOT  12 PENCILS SMALL TUBE SKULL  \
InvoiceNo
536557                                     0.0                            0.0
536984                                     0.0                            0.0
537405                                     0.0                            0.0
538163                                     0.0                            0.0
538866                                    25.0                            6.0

Description  12 PENCILS TALL TUBE RED RETROSPOT  12 PENCILS TALL TUBE SKULLS  \
InvoiceNo
536557                                    0.0                            0.0
536984                                    0.0                            0.0
537405                                    0.0                            0.0
538163                                    0.0                            0.0
538866                                    0.0                            0.0

Description  12 PINK HEN+CHICKS IN BASKET  12 RED ROSE PEG PLACE SETTINGS  \
InvoiceNo
536557                              0.0                              0.0
536984                              0.0                              1.0
537405                              0.0                              0.0
538163                              0.0                              0.0
538866                              0.0                              0.0

Description  16 PIECE CUTLERY SET PANTRY DESIGN  \
InvoiceNo
536557                                    0.0
536984                                    0.0
537405                                    0.0
538163                                    0.0
538866                                    0.0

Description  2 PICTURE BOOK EGGS EASTER BUNNY  200 BENDY SKULL STRAWS  \
InvoiceNo
536557                                  0.0                      0.0
536984                                  0.0                      0.0
537405                                  0.0                      0.0
538163                                  0.0                      0.0
538866                                  0.0                      0.0
```

Description	200 RED + WHITE BENDY STRAWS
InvoiceNo	
536557	0.0
536984	0.0
537405	0.0
538163	0.0
538866	0.0

Description	3 DRAWER ANTIQUE WHITE WOOD CABINET	3 HOOK HANGER MAGIC GARDEN
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	3 HOOK PHOTO SHELF ANTIQUE WHITE
InvoiceNo	
536557	0.0
536984	0.0
537405	0.0
538163	0.0
538866	0.0

Description	3 PIECE SPACEBOY COOKIE CUTTER SET	3 PINK HEN+CHICKS IN BASKET
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	3 STRIPEY MICE FELTCRAFT	3 TIER CAKE TIN GREEN AND CREAM
InvoiceNo		
536557	0.0	0.0
536984	0.0	1.0
537405	0.0	1.0
538163	0.0	0.0
538866	0.0	1.0

Description	3 TIER CAKE TIN RED AND CREAM
InvoiceNo	
536557	0.0
536984	0.0
537405	0.0
538163	0.0
538866	0.0

Description	3 TRADITIONAl BISCUIT CUTTERS SET	36 FOIL HEART CAKE CASES	\
InvoiceNo			
536557	0.0	0.0	
536984	0.0	0.0	
537405	0.0	0.0	
538163	0.0	0.0	
538866	0.0	0.0	

Description	36 FOIL STAR CAKE CASES	36 PENCILS TUBE RED RETROSPOT	\
InvoiceNo			
536557	0.0	0.0	
536984	0.0	0.0	
537405	0.0	0.0	
538163	0.0	0.0	
538866	0.0	0.0	

Description	36 PENCILS TUBE SKULLS	3D CHRISTMAS STAMPS STICKERS	\
InvoiceNo			
536557	0.0	1.0	
536984	0.0	0.0	
537405	0.0	0.0	
538163	0.0	0.0	
538866	0.0	0.0	

Description	4 BLUE DINNER CANDLES SILVER FLOCK	\
InvoiceNo		
536557	0.0	
536984	0.0	
537405	0.0	
538163	0.0	
538866	0.0	

Description	4 BURGUNDY WINE DINNER CANDLES	\
InvoiceNo		
536557	0.0	
536984	0.0	
537405	0.0	
538163	0.0	
538866	0.0	

Description	4 IVORY DINNER CANDLES SILVER FLOCK	\
InvoiceNo		
536557	0.0	
536984	0.0	
537405	0.0	
538163	0.0	
538866	0.0	

Description	4 PINK DINNER CANDLE SILVER FLOCK	4 PINK FLOCK CHRISTMAS BALLS \
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	4 PURPLE FLOCK DINNER CANDLES	4 SKY BLUE DINNER CANDLES \
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	5 HOOK HANGER RED MAGIC TOADSTOOL \
InvoiceNo	
536557	0.0
536984	0.0
537405	0.0
538163	0.0
538866	0.0

Description	50CM METAL STRING WITH 7 CLIPS	6 CHOCOLATE LOVE HEART T-LIGHTS \
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	6 EGG HOUSE PAINTED WOOD	6 RIBBONS RUSTIC CHARM \
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

Description	6 RIBBONS SHIMMERING PINKS	6 ROCKET BALLOONS \
InvoiceNo		
536557	0.0	0.0
536984	0.0	0.0
537405	0.0	0.0
538163	0.0	0.0
538866	0.0	0.0

```
Description  60 TEATIME FAIRY CAKE CASES  72 SWEETHEART FAIRY CAKE CASES  \
InvoiceNo
536557                              0.0                              0.0
536984                              0.0                              0.0
537405                              0.0                              0.0
538163                              0.0                              0.0
538866                              0.0                              0.0

Description  75 GREEN FAIRY CAKE CASES  ABC TREASURE BOOK BOX  \
InvoiceNo
536557                            0.0                    0.0
536984                            0.0                    0.0
537405                            0.0                    0.0
538163                            0.0                    0.0
538866                            0.0                    0.0

Description  ACRYLIC JEWEL ICICLE, BLUE  ACRYLIC JEWEL SNOWFLAKE, PINK  \
InvoiceNo
536557                              0.0                            0.0
536984                              0.0                            0.0
537405                              0.0                            0.0
538163                              0.0                            0.0
538866                              0.0                            0.0

Description  ADULT APRON APPLE DELIGHT               ...                \
InvoiceNo                                            ...
536557                            0.0               ...
536984                            0.0               ...
537405                            0.0               ...
538163                            0.0               ...
538866                            0.0               ...

Description  WOOD STAMP SET BEST WISHES  WOOD STAMP SET FLOWERS  \
InvoiceNo
536557                              0.0                     0.0
536984                              0.0                     0.0
537405                              0.0                     0.0
538163                              0.0                     0.0
538866                              0.0                     0.0

Description  WOOD STAMP SET HAPPY BIRTHDAY  WOOD STAMP SET THANK YOU  \
InvoiceNo
536557                                 0.0                       0.0
536984                                 0.0                       0.0
537405                                 0.0                       0.0
538163                                 0.0                       0.0
538866                                 0.0                       0.0
```

```
Description  WOOD STOCKING CHRISTMAS SCANDISPOT  WOODEN ADVENT CALENDAR CREAM  \
InvoiceNo
536557                                      0.0                           0.0
536984                                      0.0                           0.0
537405                                      0.0                           0.0
538163                                      0.0                           0.0
538866                                      0.0                           0.0

Description  WOODEN ADVENT CALENDAR RED  WOODEN BOX OF DOMINOES  \
InvoiceNo
536557                              0.0                     0.0
536984                              0.0                     0.0
537405                              0.0                     0.0
538163                              0.0                     0.0
538866                              0.0                     0.0

Description  WOODEN FRAME ANTIQUE WHITE  WOODEN HAPPY BIRTHDAY GARLAND  \
InvoiceNo
536557                              0.0                            0.0
536984                              0.0                            0.0
537405                              0.0                            0.0
538163                              0.0                            0.0
538866                              0.0                            0.0

Description  WOODEN HEART CHRISTMAS SCANDINAVIAN  \
InvoiceNo
536557                                       0.0
536984                                       0.0
537405                                       0.0
538163                                       0.0
538866                                       0.0

Description  WOODEN PICTURE FRAME WHITE FINISH  WOODEN REGATTA BUNTING  \
InvoiceNo
536557                                     0.0                     0.0
536984                                     0.0                     0.0
537405                                     0.0                     0.0
538163                                     0.0                     0.0
538866                                     0.0                     1.0

Description  WOODEN SCHOOL COLOURING SET  WOODEN STAR CHRISTMAS SCANDINAVIAN  \
InvoiceNo
536557                               0.0                                 0.0
536984                               0.0                                 0.0
537405                               0.0                                 0.0
538163                               0.0                                 0.0
538866                               0.0                                 0.0
```

```
Description  WOODEN TREE CHRISTMAS SCANDINAVIAN  WOODEN UNION JACK BUNTING  \
InvoiceNo
536557                                      0.0                        0.0
536984                                      0.0                        0.0
537405                                      0.0                        0.0
538163                                      0.0                        0.0
538866                                      0.0                        0.0

Description  WOODLAND CHARLOTTE BAG  WOODLAND DESIGN  COTTON TOTE BAG  \
InvoiceNo
536557                          0.0                                0.0
536984                          5.0                                0.0
537405                          0.0                                0.0
538163                          0.0                                0.0
538866                          0.0                                0.0

Description  WOODLAND MINI BACKPACK  WOVEN ROSE GARDEN CUSHION COVER  \
InvoiceNo
536557                          0.0                              0.0
536984                          0.0                              0.0
537405                          0.0                              0.0
538163                          0.0                              0.0
538866                          0.0                              0.0

Description  WOVEN SUNSET CUSHION COVER  WRAP 50'S  CHRISTMAS  \
InvoiceNo
536557                              0.0                   0.0
536984                              0.0                   0.0
537405                              0.0                   0.0
538163                              0.0                   0.0
538866                              0.0                   0.0

Description  WRAP ALPHABET DESIGN  WRAP CHRISTMAS VILLAGE  WRAP COWBOYS  \
InvoiceNo
536557                        0.0                     0.0           0.0
536984                        0.0                     0.0           0.0
537405                        0.0                     0.0           0.0
538163                        0.0                     0.0           0.0
538866                        0.0                     0.0           0.0

Description  WRAP DOILEY DESIGN  WRAP DOLLY GIRL  WRAP ENGLISH ROSE  \
InvoiceNo
536557                      0.0              0.0                0.0
536984                      0.0              0.0                0.0
537405                      0.0              0.0                0.0
538163                      0.0              0.0                0.0
538866                      0.0              0.0                0.0
```

```
Description  WRAP GREEN PEARS  WRAP I LOVE LONDON  WRAP MAGIC FOREST  \
InvoiceNo
536557                    0.0                 0.0                0.0
536984                    0.0                 0.0                0.0
537405                    0.0                 0.0                0.0
538163                    0.0                 0.0                0.0
538866                    0.0                 0.0                0.0

Description  WRAP PINK FAIRY CAKES  WRAP POPPIES  DESIGN  WRAP RED APPLES  \
InvoiceNo
536557                         0.0                   0.0              0.0
536984                         0.0                   0.0              0.0
537405                         0.0                   0.0              0.0
538163                         0.0                   0.0              0.0
538866                         0.0                   0.0              0.0

Description  WRAP SUKI AND FRIENDS  WRAP VINTAGE LEAF DESIGN  \
InvoiceNo
536557                         0.0                       0.0
536984                         0.0                       0.0
537405                         0.0                       0.0
538163                         0.0                       0.0
538866                         0.0                       0.0

Description  WRAP VINTAGE PETALS  DESIGN  YELLOW EASTER EGG HUNT START POST  \
InvoiceNo
536557                               0.0                                0.0
536984                               0.0                                0.0
537405                               0.0                                0.0
538163                               0.0                                0.0
538866                               0.0                                0.0

Description  ZINC  HEART T-LIGHT HOLDER  ZINC FINISH 15CM PLANTER POTS  \
InvoiceNo
536557                              0.0                            0.0
536984                              0.0                            0.0
537405                              0.0                            0.0
538163                              0.0                            0.0
538866                              0.0                            0.0

Description  ZINC FOLKART SLEIGH BELLS  ZINC HEART FLOWER T-LIGHT HOLDER  \
InvoiceNo
536557                             0.0                               0.0
536984                             0.0                               0.0
537405                             0.0                               0.0
538163                             0.0                               0.0
538866                             0.0                               0.0
```

```
Description  ZINC HERB GARDEN CONTAINER  ZINC METAL HEART DECORATION  \
InvoiceNo
536557                              0.0                          0.0
536984                              0.0                          0.0
537405                              0.0                          0.0
538163                              0.0                          0.0
538866                              0.0                          0.0

Description  ZINC SWEETHEART SOAP DISH  ZINC SWEETHEART WIRE LETTER RACK  \
InvoiceNo
536557                             0.0                                0.0
536984                             0.0                                0.0
537405                             0.0                                0.0
538163                             0.0                                0.0
538866                             0.0                                0.0

Description  ZINC T-LIGHT HOLDER STAR LARGE  ZINC T-LIGHT HOLDER STARS SMALL  \
InvoiceNo
536557                                  0.0                               0.0
536984                                  0.0                               0.0
537405                                  0.0                               0.0
538163                                  0.0                               0.0
538866                                  0.0                               0.0

Description  ZINC WIRE SWEETHEART LETTER TRAY
InvoiceNo
536557                                    0.0
536984                                    0.0
537405                                    0.0
538163                                    0.0
538866                                    0.0

[5 rows x 1343 columns]
```

```
In [694]: def encode_units(x):
              if x <= 0:
                  return 0
              if x >= 1:
                  return 1

          basket_sets = basket.applymap(encode_units)
```

Frequent items sets where calculated using the apriori algorithm, with a minimum support of 20%, that is 20% probability that one item will be purchased with another item in the same order.

```
In [695]: frequent_itemsets = apriori(basket_sets, min_support=0.2, use_colnames=True) #70% su
```

Association rules were generated using the “lift” metric [4]; the ratio of the observed support that would be expected if the antecedent and consequent were independent.

```
In [696]: rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
          rules.head(20)
```

```
Out[696]:                          antecedents                          consequents  \
          0  (BLUE/CREAM STRIPE CUSHION COVER)                      (CHILLI LIGHTS)
          1                    (CHILLI LIGHTS)  (BLUE/CREAM STRIPE CUSHION COVER)
          2                (GUMBALL COAT RACK)  (BLUE/CREAM STRIPE CUSHION COVER)
          3  (BLUE/CREAM STRIPE CUSHION COVER)                  (GUMBALL COAT RACK)
          4   (PACK OF 60 DINOSAUR CAKE CASES)  (BLUE/CREAM STRIPE CUSHION COVER)
          5  (BLUE/CREAM STRIPE CUSHION COVER)   (PACK OF 60 DINOSAUR CAKE CASES)
          6        (CHARLOTTE BAG SUKI DESIGN)                      (CHILLI LIGHTS)
          7                    (CHILLI LIGHTS)        (CHARLOTTE BAG SUKI DESIGN)
          8   (PACK OF 60 DINOSAUR CAKE CASES)                      (CHILLI LIGHTS)
          9                    (CHILLI LIGHTS)   (PACK OF 60 DINOSAUR CAKE CASES)

             antecedent support  consequent support  support  confidence  lift  \
          0                 0.4                 0.5      0.2         0.5   1.0
          1                 0.5                 0.4      0.2         0.4   1.0
          2                 0.4                 0.4      0.2         0.6   1.5
          3                 0.4                 0.4      0.2         0.5   1.5
          4                 0.4                 0.4      0.2         0.5   1.4
          5                 0.4                 0.4      0.2         0.5   1.4
          6                 0.4                 0.5      0.2         0.6   1.1
          7                 0.5                 0.4      0.2         0.4   1.1
          8                 0.4                 0.5      0.2         0.6   1.2
          9                 0.5                 0.4      0.2         0.5   1.2

             leverage  conviction
          0       0.0         1.0
          1       0.0         1.0
          2       0.1         1.4
          3       0.1         1.4
          4       0.1         1.3
          5       0.1         1.3
          6       0.0         1.2
          7       0.0         1.1
          8       0.0         1.3
          9       0.0         1.2
```

3.1.2 Generate a basket for a country

Per country trends could be identified which could assist with business metrics such as optimum location of distribution centres, load balancing of servers, and also placement of items within a warehouse to optimise item picking.

```
In [726]: basket = (data[data['Country'] == "Switzerland"]
                    .groupby(['InvoiceNo', 'Description'])['Quantity']
                    .sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))
```

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

basket_sets = basket.applymap(encode_units)
```

In [731]:
```
frequent_itemsets = apriori(basket_sets, min_support=0.2, use_colnames=True) #70% su
```

In [732]:
```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head(20)
```

Out[732]:
```
                          antecedents                          consequents  \
0     (PLASTERS IN TIN CIRCUS PARADE)          (PLASTERS IN TIN SPACEBOY)
1          (PLASTERS IN TIN SPACEBOY)     (PLASTERS IN TIN CIRCUS PARADE)
2  (PLASTERS IN TIN WOODLAND ANIMALS)          (PLASTERS IN TIN SPACEBOY)
3          (PLASTERS IN TIN SPACEBOY)  (PLASTERS IN TIN WOODLAND ANIMALS)

   antecedent support  consequent support  support  confidence  lift  \
0                 0.2                 0.4      0.2         0.9   2.4
1                 0.4                 0.2      0.2         0.6   2.4
2                 0.3                 0.4      0.3         0.8   2.1
3                 0.4                 0.3      0.3         0.7   2.1

   leverage  conviction
0       0.1         6.8
1       0.1         1.7
2       0.1         3.3
3       0.1         2.4
```

4 References

[1] https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54

[2] https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[3] https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

[4] https://www.wikiwand.com/en/Association_rule_learning#/Lift