Market Basket Analysis

This project is for deriving association rules using mlxtend and doing MARKET BASKET analysis.

The is done using a public dataset of real e-commerce transactions.

Executive Summary

DATA

The UCI Machine Learning Repository has made this dataset containing actual transactions from 2010 and 2011. The dataset is maintained on their site, where it can be found by the title "Online Retail".

The dataset contains 541909 rows of which 486165 are for UK customers and the rest are for other countries

EDA

See visuals in the notebook

DATA ENGINEERING

Descriptions
- Remove rows with NULL descriptions, remove leading and trailing spaces from descriptions. This is required as descriptions are to be used for Association Analysis

Invoice No
- InvoiceNo with a prefix of A are for adjustments and prefix of C are for credits. Remove rows with InvoiceNo with a C or A prefix

Split data into 2 DataFrames
- UK customers and EU customers
- UK DataFrame to be used for Association was limited to 200000 transactions due to performance constraints

RESULTS

(Note: See accompanying presentation for explanations on the metrics mentioned in the results)

The top 5 interesting itemsets for UK are

antecedants	consequents	antecedent support	consequent support	support	confide
(GREEN REGENCY TEACUP AND SAUCER, ROSES REGENC...	(PINK REGENCY TEACUP AND SAUCER)	0.046763	0.044282	0.030141	0.64
(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER, ROSES REGENC...	0.044282	0.046763	0.030141	0.68
(GREEN REGENCY TEACUP AND SAUCER)	(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	0.064376	0.032746	0.030141	0.46

(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	(GREEN REGENCY TEACUP AND SAUCER)	0.032746	0.064376	0.030141	0.92(
(PINK REGENCY TEACUP AND SAUCER, GREEN REGENCY...	(ROSES REGENCY TEACUP AND SAUCER)	0.034235	0.064004	0.030141	0.88(

The top 5 interesting itemsets for EU are

antecedants	consequents	antecedent support	consequent support	suppo
(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.05949	0.058924	0.0526(
(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.058924	0.05949	0.0526(
(SPACEBOY LUNCH BOX)	(DOLLY GIRL LUNCH BOX)	0.106516	0.078187	0.0566(
(DOLLY GIRL LUNCH BOX)	(SPACEBOY LUNCH BOX)	0.078187	0.106516	0.0566(
(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.121246	0.096884	0.0645(

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
import seaborn as sns # Advanced data visualization
import re # Regular expressions for advanced string selection
from mlxtend.frequent_patterns import apriori # Data pattern exploration
from mlxtend.frequent_patterns import association_rules # Association rules conversion
from mlxtend.preprocessing import OnehotTransactions # Transforming dataframe for apriori
import missingno as msno # Advanced missing values handling
%matplotlib inline
```

Load Data

In [2]:

```
#Load data into DataFrame
df = pd.read_csv('./data/datafile.csv',encoding="ISO-8859-1",dtype = {'CustomerID':str})
```

Data Engineering

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      541909 non-null object
StockCode      541909 non-null object
Description    540455 non-null object
Quantity       541909 non-null int64
InvoiceDate    541909 non-null object
UnitPrice      541909 non-null float64
CustomerID     406829 non-null object
Country        541909 non-null object
dtypes: float64(1), int64(1), object(6)
memory usage: 33.1+ MB
```

In [4]:

```
df.head()
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom

In [3]:

```
# Convert Invoice Date to datetime and get various components to help analyse data.
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['InvYr'] = df.InvoiceDate.dt.year
df['InvMth'] = df.InvoiceDate.dt.month
df['InvDOW'] = df.InvoiceDate.dt.weekday_name
df['InvHr'] = df.InvoiceDate.dt.hour
df.set_index(['InvoiceDate'],inplace=True)
```

In [4]:

```
# Invoice Numbers have an alpha prefix. Find out the prefixes
df['InvoiceNo'][df.InvoiceNo.str.match('^[A-Za-z]')].str[0].unique()
```

Out[4]:

```
array(['C', 'A'], dtype=object)
```

In [5]:

```
# Invoice No is present for all rows but is an object whereas it should be an integer.
# Find invoice numbers that have alpha characters. Invoices with a C prefix have negative q
# are credit invoices and have to be removed.
df = df[~df.InvoiceNo.str.contains('C')]   # Write to DataFrame all the rows where InvoiceN
df = df[~df.InvoiceNo.str.contains('A')]   # Write to DataFrame all the rows where InvoiceN
```

In [6]:

```
# Drop rows with null Descriptions as they are going to be used for association rules
df.dropna(inplace=True, subset=['Description'])
df.info()
```

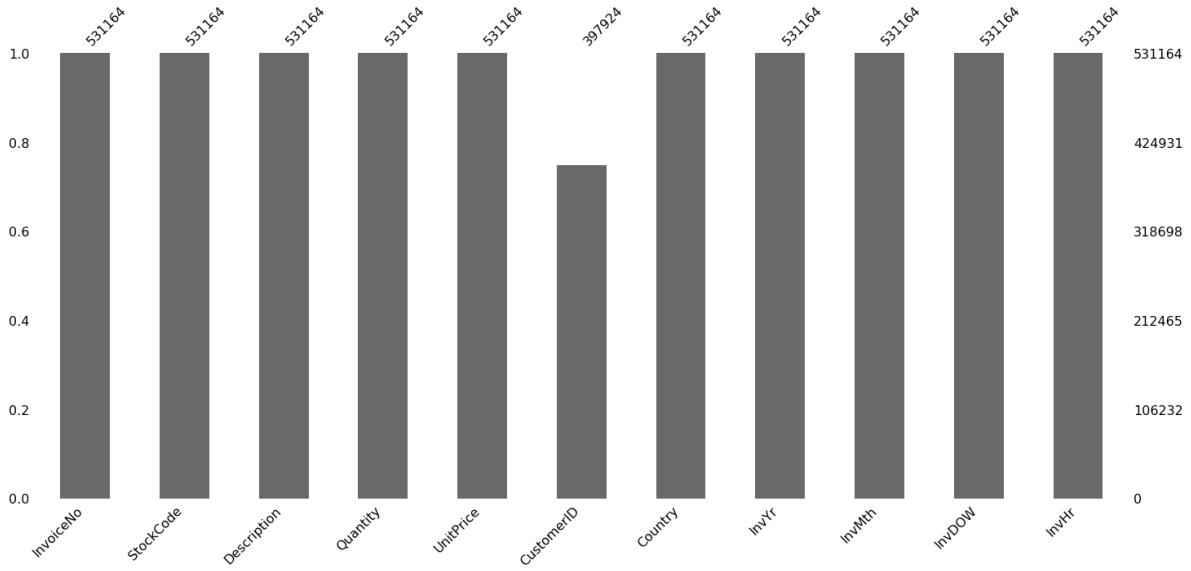
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 531164 entries, 2010-12-01 08:26:00 to 2011-12-09 12:50:00
Data columns (total 11 columns):
InvoiceNo      531164 non-null object
StockCode      531164 non-null object
Description    531164 non-null object
Quantity       531164 non-null int64
UnitPrice      531164 non-null float64
CustomerID     397924 non-null object
Country        531164 non-null object
InvYr          531164 non-null int64
InvMth         531164 non-null int64
InvDOW         531164 non-null object
InvHr          531164 non-null int64
dtypes: float64(1), int64(4), object(6)
memory usage: 48.6+ MB
```

In [10]:

```
# Visualise missing values. Descriptions are missing for some rows and Customer Id is missi
msno.bar(df)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a60c32a58>
```



In [7]:

```
# Change datatypes
df['InvoiceNo'] = df.InvoiceNo.astype('int')
df['Country'] = df.Country.astype('category')
```

In [8]:

```
# Create column for sale value
df['Amt'] = df.Quantity * df.UnitPrice
```

In [9]:

```
# Remove spaces from Description for Association Rules derivation
df['Description'] = df.Description.str.strip()
```

In [10]:

```
df.Country.value_counts()
```

Out[10]:

```
United Kingdom          486165
Germany                   9042
France                    8408
EIRE                      7894
Spain                     2485
Netherlands               2363
Belgium                   2031
Switzerland               1967
Portugal                  1501
Australia                 1185
Norway                    1072
Italy                      758
Channel Islands            748
Finland                    685
Cyprus                     614
Sweden                     451
Unspecified                446
Austria                    398
Denmark                    380
Poland                     330
Japan                      321
Israel                     295
Hong Kong                  284
Singapore                  222
Iceland                    182
USA                        179
Canada                     151
Greece                     145
Malta                      112
United Arab Emirates        68
European Community          60
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              25
Bahrain                     18
Saudi Arabia                 9
Name: Country, dtype: int64
```

Data Visualisation

In [14]:

```
# Create DataFrame with Country Information
dfcntry = df[['Country','InvoiceNo']].groupby(['Country']).count().reset_index()
```

In [15]:

```
dfcntry.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 2 columns):
Country      38 non-null category
InvoiceNo    38 non-null int64
dtypes: category(1), int64(1)
memory usage: 2.0 KB
```

In [16]:

```
plt.figure(figsize=(9,6))
plt.title('Distribution of Invoices by Country')
plt.xticks(rotation=60)
#sns.countplot(y='Country',data=df)
sns.barplot(x='Country',y='InvoiceNo',data =dfcntry)
```

Out[16]:

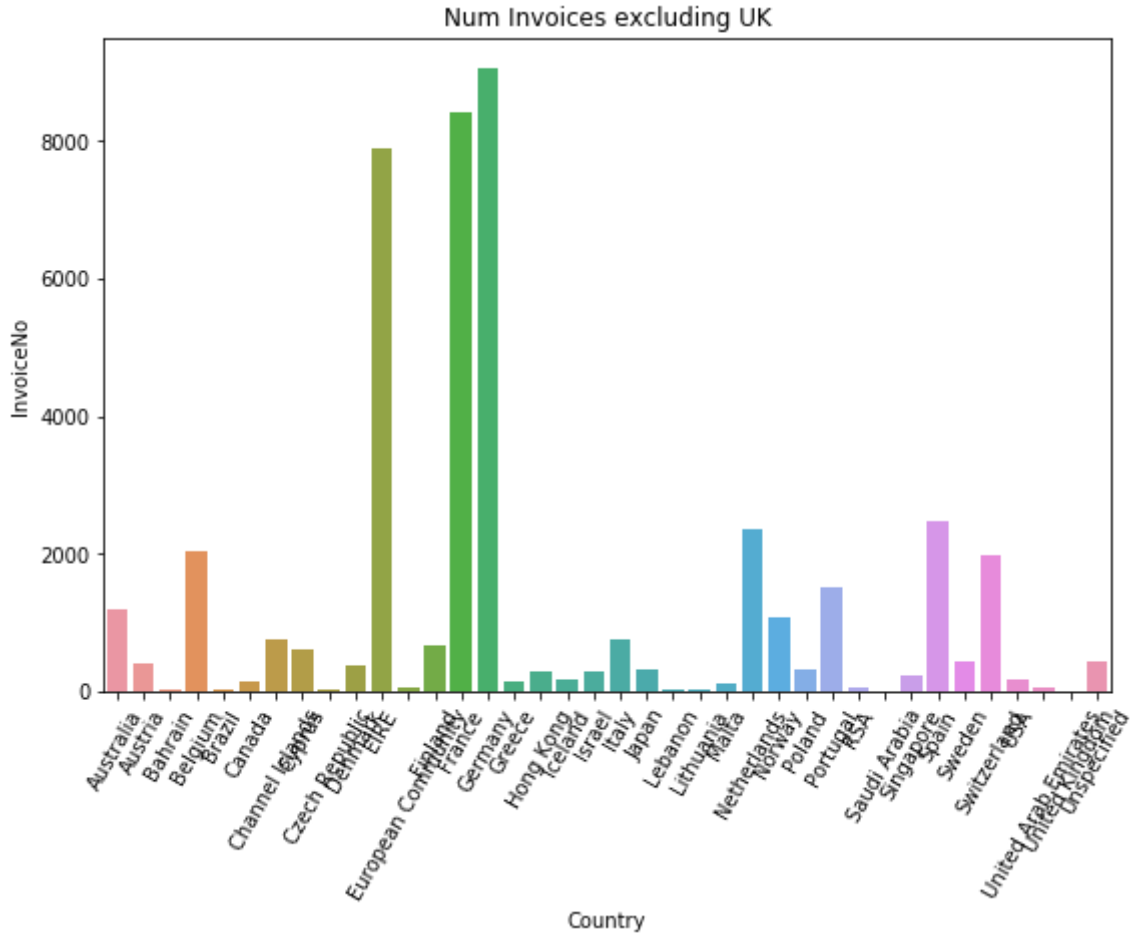
```
<matplotlib.axes._subplots.AxesSubplot at 0x18a6073e940>
```

In [17]:

```
plt.figure(figsize=(9,6))
plt.title('Num Invoices excluding UK')
plt.xticks(rotation=60)
sns.barplot(x='Country',y='InvoiceNo',data =dfcntry[dfcntry.Country!='United Kingdom'])
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a6095ca90>
```



In [11]:

```
# Create a dataframe for UK only
dfUK = df[df.Country == 'United Kingdom']
```

In [12]:

```
dfUK.shape
```

Out[12]:

```
(486165, 12)
```

In [13]:

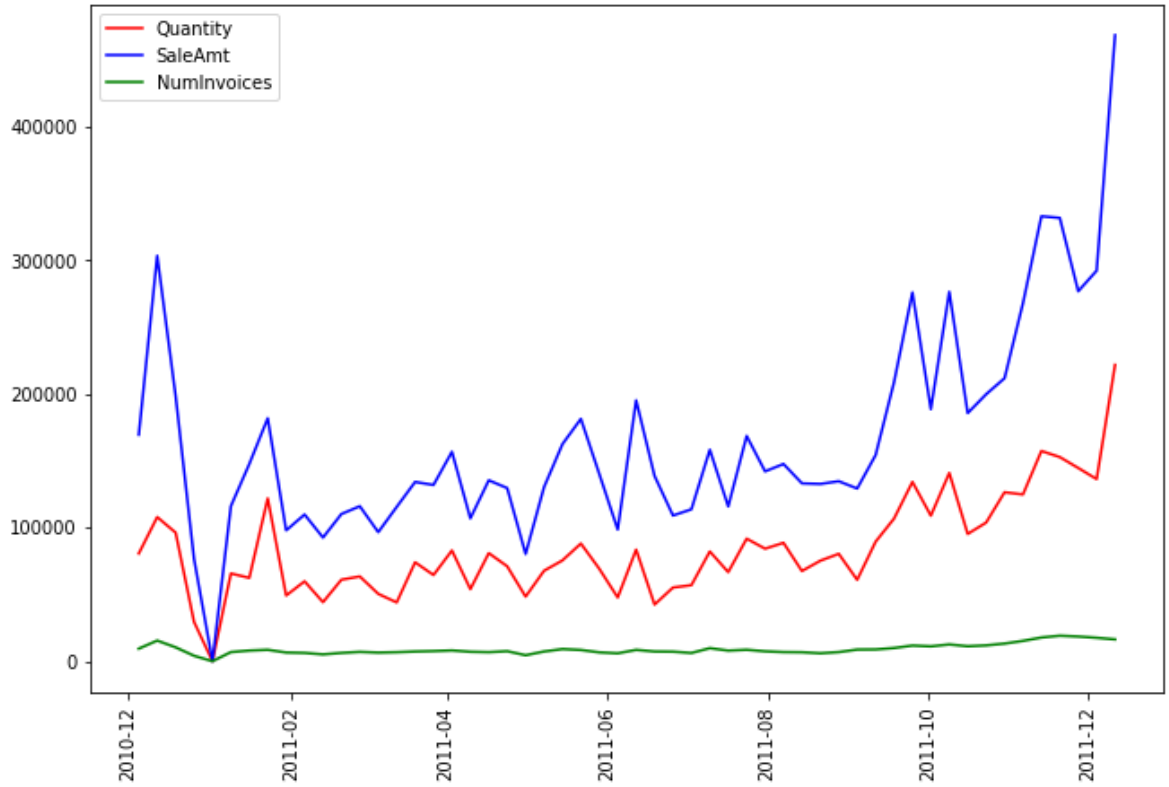
```
# Make a list of EU countries to club them together
EUlist = ['Germany','France','EIRE','Spain','Netherlands','Belgium','Switzerland','Portugal
          'Finland','Cyprus','Sweden','Austria','Denmark','Poland','Iceland','Greece','Malt
dfEU = df[df.Country.isin(EUlist)]
```

In [36]:

```
# Create time series dataframe
dftsUK = dfUK.groupby(['InvoiceDate'])['InvoiceNo','Quantity','Amt']\
       .agg({'InvoiceNo':'count','Quantity':'sum','Amt':'sum'})
dftsUK.columns = ['Count','Quantity','SaleAmt']
```

In [38]:

```
# Resample timeseries dataframe - fill Nans with next value - for Quantity
plt.figure(figsize=(9,6))
plt.title = 'Sale Trend for UK'
plt.xticks(rotation=90)
plt.tight_layout()
plt.plot(dftsUK.Quantity.resample('W').sum().bfill(),color='red',label = 'Quantity')
plt.plot(dftsUK.SaleAmt.resample('W').sum().bfill(),color='blue', label='SaleAmt')
plt.plot(dftsUK.Count.resample('W').sum().bfill(),color='green', label='NumInvoices')
plt.legend(loc=2)
plt.draw()
```



In [67]:

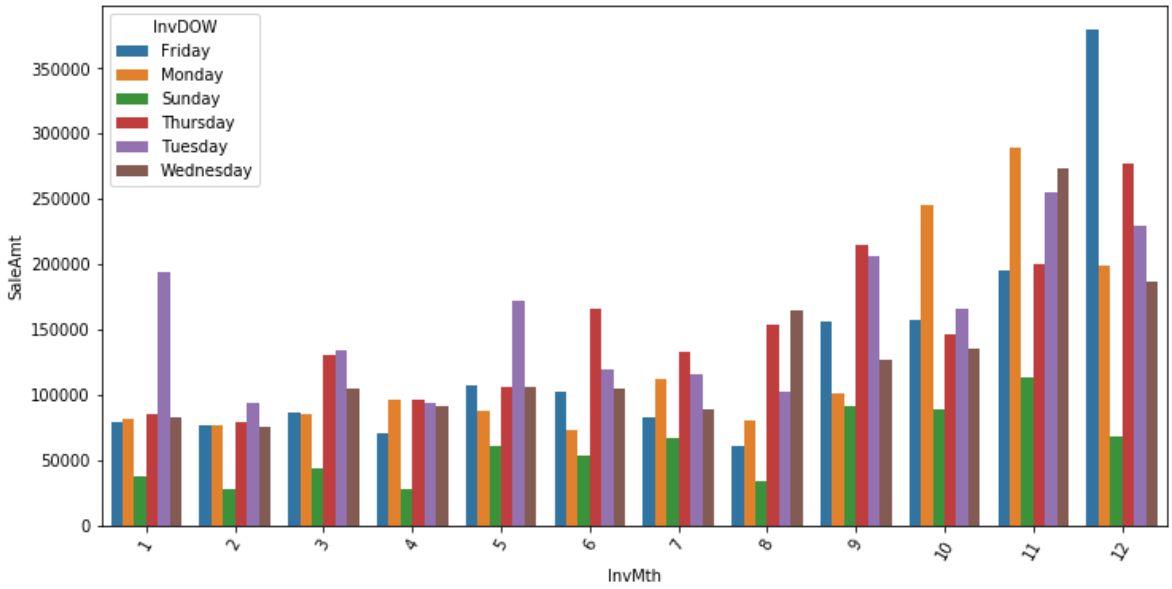
```
# Visualise Sales by Day of Week within Month
dfdowUK = dfUK.groupby(['InvMth','InvDOW'])['InvoiceNo','Quantity','Amt']\
       .agg({'InvoiceNo':'count','Quantity':'sum','Amt':'sum'})
dfdowUK.columns = ['Count','Quantity','SaleAmt']
dfdowUK.reset_index(inplace=True)
```

In [68]:

```
plt.figure(figsize=(12,6))
#plt.title('Sale by Day of Week')
plt.xticks(rotation=60)
sns.barplot(x='InvMth',y='SaleAmt',hue='InvDOW',data =dfdowUK)
```

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a6ab0fd30>
```

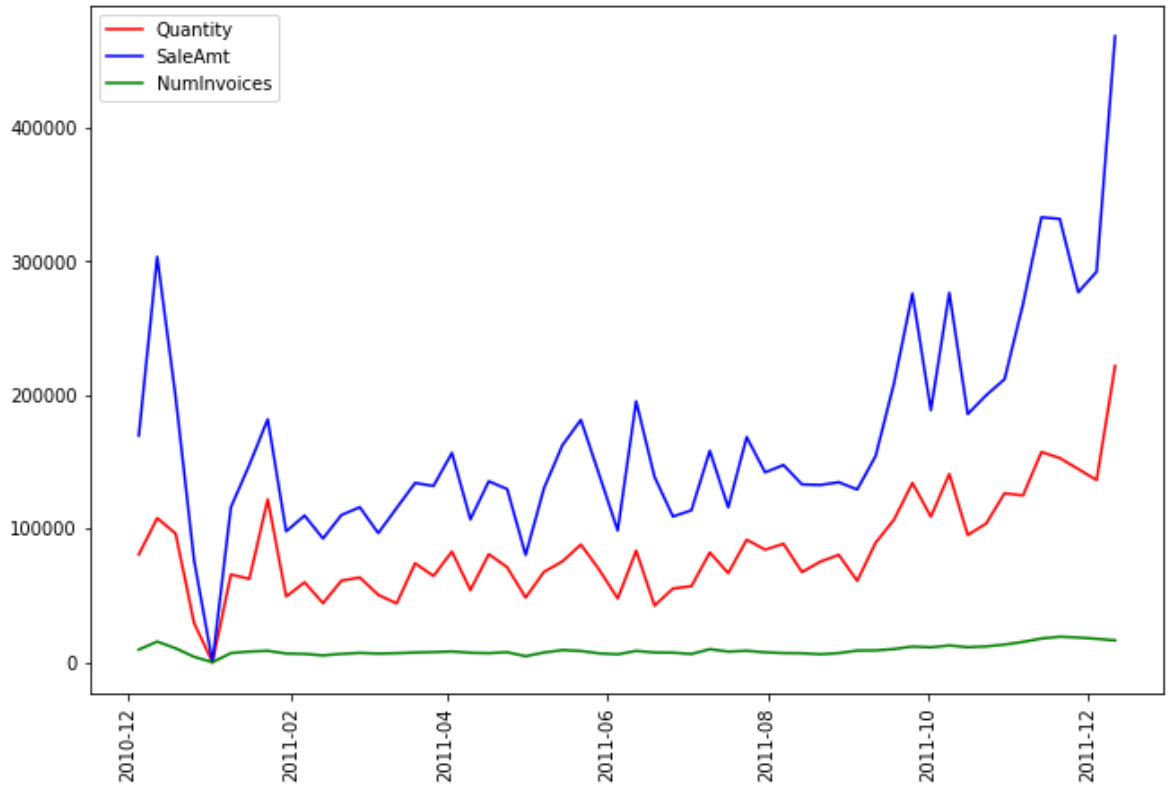


In [63]:

```
# Create time series datafrome
dftsEU = dfEU.groupby(['InvoiceDate'])['InvoiceNo','Quantity','Amt']\
       .agg({'InvoiceNo':'count','Quantity':'sum','Amt':'sum'})
dftsEU.columns = ['Count','Quantity','SaleAmt']
```

In [64]:

```
# Resample timeseries dataframe - fill Nans with next value - for Quantity
plt.figure(figsize=(9,6))
plt.title = 'Sale Trend for UK'
plt.xticks(rotation=90)
plt.tight_layout()
plt.plot(dftsUK.Quantity.resample('W').sum().bfill(),color='red',label = 'Quantity')
plt.plot(dftsUK.SaleAmt.resample('W').sum().bfill(),color='blue', label='SaleAmt')
plt.plot(dftsUK.Count.resample('W').sum().bfill(),color='green', label='NumInvoices')
plt.legend(loc=2)
plt.draw()
```



In [69]:

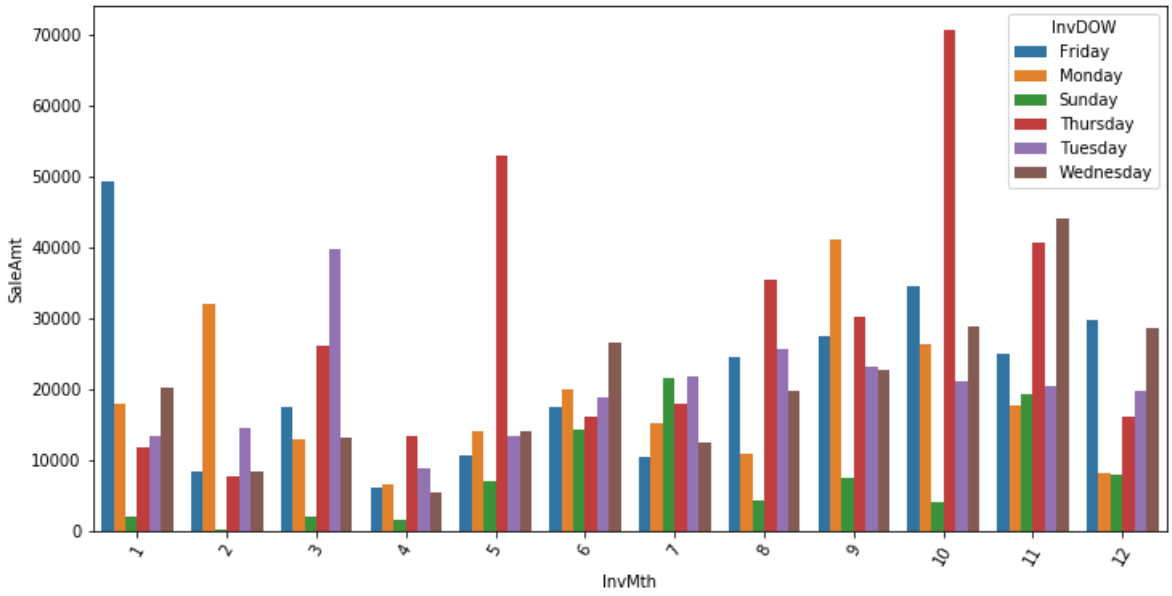
```
# Visualise Sales by Day of Week within Month
dfdowEU = dfEU.groupby(['InvMth','InvDOW'])['InvoiceNo','Quantity','Amt']\
       .agg({'InvoiceNo':'count','Quantity':'sum','Amt':'sum'})
dfdowEU.columns = ['Count','Quantity','SaleAmt']
dfdowEU.reset_index(inplace=True)
```

In [72]:

```
plt.figure(figsize=(12,6))
#plt.title('Sale by Day of Week')
plt.xticks(rotation=60)
sns.barplot(x='InvMth',y='SaleAmt',hue='InvDOW',data =dfdowEU)
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a6af0d8d0>
```



MARKET BASKET ANALYSIS

In [42]:

```
# Prepare the datasets - EU
# dfEU.reset_index(inplace=True)
bEU = dfEU[['InvoiceNo','Description']][dfEU.Description != 'POSTAGE']
gdEU = pd.get_dummies(bEU,columns=['Description'],prefix='',prefix_sep='')
basketEU = pd.pivot_table(gdEU,index='InvoiceNo',aggfunc='sum')
```

In [36]:

```
# dfUK.reset_index(inplace=True)
# x = dfUK[['InvoiceNo','Description']][dfUK.Description != 'POSTAGE']
# bUK = x.iloc[0:200000,:]
# gdUK = pd.get_dummies(bUK,columns=['Description'],prefix='',prefix_sep='')
# basketUK = pd.pivot_table(gdUK,index='InvoiceNo',aggfunc='sum')
```

In [37]:

```
basketUK.shape
```

Out[37]:

```
(8062, 3478)
```

In [43]:

```
# Use Apriori to generate frequent itemsets. Use minimum support of 5%
freqitemEU= apriori(basketEU, min_support=0.05, use_colnames=True)
# Generate rules
rulesEU = association_rules(freqitemEU, metric="lift", min_threshold=1)
```

In [47]:

```
# Use Apriori to generate frequent itemsets. Use minimum support of 5%
freqitemUK= apriori(basketUK, min_support=0.03, use_colnames=True)
# Generate rules
rulesUK = association_rules(freqitemUK, metric="lift", min_threshold=1)
```

In [49]:

```
rulesUK.shape
```

Out[49]:

```
(32, 9)
```

In [53]:

```
rulesUK.sort_values(by='lift',ascending=False).head(5)
```

Out[53]:

	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	lever
28	(GREEN REGENCY TEACUP AND SAUCER, ROSES REGENC...	(PINK REGENCY TEACUP AND SAUCER)	0.046763	0.044282	0.030141	0.644562	14.555915	0.028
29	(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER, ROSES REGENC...	0.044282	0.046763	0.030141	0.680672	14.555915	0.028
30	(GREEN REGENCY TEACUP AND SAUCER)	(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	0.064376	0.032746	0.030141	0.468208	14.298082	0.028
27	(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	(GREEN REGENCY TEACUP AND SAUCER)	0.032746	0.064376	0.030141	0.920455	14.298082	0.028
26	(PINK REGENCY TEACUP AND SAUCER, GREEN REGENCY...	(ROSES REGENCY TEACUP AND SAUCER)	0.034235	0.064004	0.030141	0.880435	13.755940	0.027

In [54]:

```
rulesEU.sort_values(by='lift',ascending=False).head()
```

Out[54]:

	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	lever
14	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.059490	0.058924	0.052691	0.885714	15.031593	0.049
15	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.058924	0.059490	0.052691	0.894231	15.031593	0.049
0	(SPACEBOY LUNCH BOX)	(DOLLY GIRL LUNCH BOX)	0.106516	0.078187	0.056657	0.531915	6.803114	0.048
1	(DOLLY GIRL LUNCH BOX)	(SPACEBOY LUNCH BOX)	0.078187	0.106516	0.056657	0.724638	6.803114	0.048
7	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.121246	0.096884	0.064589	0.532710	5.498442	0.052

In [ ]: