E-COMMERCE DATASET

Vikrant Patil

Overview:

This is a dataset which contains the transactions of a company over the period of one year taking place in different countries. The dataset contains 8 variables or attributes which describe the dataset.

Here, I have initially found out the dimensions of the loaded dataset.

The next stage is of data preparation and data cleaning. Some new columns are added in this stage. Also, in this stage we find out the unique occurrences of some variables.

After this stage I have performed Market Basket Analysis to find out which products the customers tend to buy together and also a Product Recommendation which predicts what the customer is likely to buy based on his previous purchases.

The last stage consists of exploratory data analysis which consists of finding out the top revenue generating countries and the most sold products.

Load the data from the Excel file using library(readxl)

```
data <- read_excel("data_science_analytics_2018_data.xlsx")
```

View the loaded data in tabular format

```
View(data)
```

Dimensions of the Loaded data. It is observed that the data has 541909 rows and 8 columns or variables.

```
dim(data)
[1] 541909      8
```

View the data along with some of its values and data types

```
str(data)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   541909 obs. of  8 variables:
$ InvoiceNo  : chr  "C581569" "C581569" "C581568" "C581499" ...
$ StockCode  : chr  "20979" "84978" "21258" "M" ...
$ Description: chr  "36 PENCILS TUBE RED RETROSPOT" "HANGING HEART JAR T-LIGHT HOLDER" "V
ICTORIAN SEWING BOX LARGE" "Manual" ...
$ Quantity   : num  -5 -1 -5 -1 -12 …
$ InvoiceDate: POSIXct, format: "2011-12-09 11:58:00" "2011-12-09 11:58:00" "2011-12-09 1
1:57:00" "2011-12-09 10:28:00" ...
$ UnitPrice  : num  1.25 1.25 10.95 224.69 1.95 ...
$ CustomerID : num  17315 17315 15311 15498 14397 ...
$ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" .
..
```

Convert the 'InvoiceDate' variable into Hours only format. Here, a variable 'tmp' is used to initially generate a copy of the original dataset. The Date variable from this copy dataset is loaded into variable called 'inhours'. This variable is then converted into standard Date-Time format using strptime() function. Using hour () function the time is converted into hours only format.

```
tmp <- data

inhours <- tmp$InvoiceDate

strptime(inhours, format = '%Y-%m-%d %H:%M:%S', tz = '')
[1] "2011-12-09 11:58:00 CST" "2011-12-09 11:58:00 CST" "2011-12-09 11:57:00 CST" "2011-1
2-09 10:28:00 CST"
[5] "2011-12-09 09:57:00 CST" "2011-12-09 09:57:00 CST" "2011-12-09 09:27:00 CST" "2011-1
2-08 19:28:00 CST"
(displaying initial 2 rows only)

time_inhours <- hour(inhours)

data$Time_in_hours <- time_inhours

data$Time_in_hours
[1] 11 11 11 10  9  9  9 19 19 19 19 19 19 19 19 18 18 18 18 18 18 18 18 18 18 18 14 14 1
4 14 14 14 14 14 14 14 14 14 14
[40] 13 13 13 13 11 11 11 11 11 11 11 11 11 11 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 15 15 14 14
```

Now, to clear off the NA data in the dataset, we use the complete.cases () function. Using is.na() we can confirm that there are no more NA entries left in the dataset.

```
data <- data[complete.cases(data), ]

colSums(is.na(data))

InvoiceNo     StockCode   Description      Quantity   InvoiceDate     UnitPrice    Custom
erID       Country
            0             0             0             0             0             0
0            0
Time_in_hours
            0
```

We now add a new column called Date which stores only the dates (1-31). In the second line we create an important variable called 'TotalCost' which stores the product of the Quantity of the items purchased and its Unit Price. Lastly we convert the Country into factors. It generates 37 levels, each for every country.

```
data$Date <- as.Date(data$InvoiceDate)

data$TotalCost <- (data$Quantity * data$UnitPrice)

data$Country <- as.factor(data$Country)
```

While looking at the dataset, we come across duplicate values. In this code section, we focus on removing the duplicate entries from the dataset. Using the duplicated() function, we search for the duplicate values and remove them.

```
dataRemoveDuplicate <- data.table(data)
dataRemoveDuplicate <- dataRemoveDuplicate[!duplicated(dataRemoveDuplicate)]
data <- dataRemoveDuplicate
```

Here, we find how many Countries the dataset has. The unique() function is used to find only the unique values in the dataset. We find that there are 37 different countries present in this dataset.

```
data.table(unique(data$Country))

                     V1
 1:      United Kingdom
 2:             Germany
 3:     Channel Islands
 4:              France
 5:                 USA
 6:               Spain
 7:            Portugal
 8:             Finland
 9:               Japan
10:              Sweden
11:             Belgium
12:              Cyprus
13:                EIRE
14:               Malta
15:               Italy
16:         Switzerland
17:         Netherlands
18:              Poland
19:      Czech Republic
20:           Australia
21:             Denmark
22:           Singapore
23:              Norway
24:              Greece
25:             Austria
26:  European Community
27:        Saudi Arabia
28:              Israel
29:             Iceland
30:                 RSA
31: United Arab Emirates
32:              Canada
33:         Unspecified
34:             Bahrain
35:              Brazil
36:             Lebanon
37:           Lithuania
                     V1
```

This map plot shows the top 10 countries with respect to the transactions in the dataset.

Now we find the number of different products present in the dataset. We find that there are 3684 different products.

```
data.table(unique(data$StockCode))

          V1
   1:  20979
   2:  84978
   3:  21258
   4:      M
   5:  22178
  ---
3680: 35271S
3681:  21488
3682:  82615
3683:  21895
3684:  84854
```

Now we find the number of different Customers present in the dataset. We find that there are 4372 different customers. We can also say that over the period of 1 year, the company had 4372 different customers over the world.

```
data.table(unique(data$CustomerID))

         V1
   1: 17315
   2: 15311
   3: 15498
   4: 14397
   5: 16446
  ---
4368: 16583
4369: 17908
4370: 12791
4371: 13747
4372: 18074
```

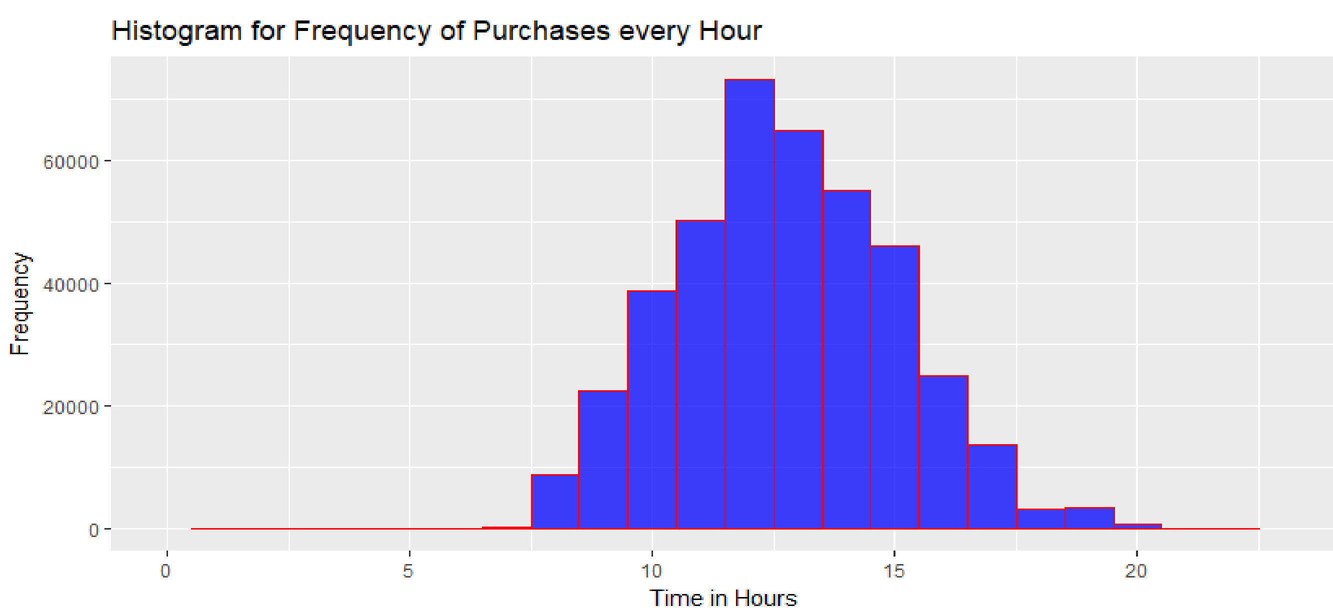
Now we find the number of different Transactions in the dataset. We find that there are 22190 different transactions. We can also say that over the period of 1 year, the company had 22190 different transactions over the world.

```
data.table(unique(data$InvoiceNo))

            V1
    1: C581569
    2: C581568
    3: C581499
    4: C581490
    5: C581484
   ---
22186:  536369
22187:  536368
22188:  536367
22189:  536366
22190:  536365
```

We plot the frequency of purchases every hour. We observe that most of the purchases occur between 10 am to 3 pm in the afternoon.

```
qplot(data$Time_in_hours, geom="histogram", xlim = c(0,23), binwidth = 1, main = "H
istogram for Frequency of Purchases every Hour", xlab = "Time in Hours", ylab = "Fr
equency", fill=I("blue"), col=I("red"), alpha=I(0.75))
```



Now, we remove the negative values. We can see that the dataset has a code 'C' before some Invoice numbers. Also, the quantities for these transactions are in negative values. Hence, we remove them from the dataset.

```
detach(package:plyr)
library(dplyr)
```

Using filter(), we remove all the values from Quantity variable which are less than 0

```
a <- filter(data, Quantity > 0)
View(a)
data <- a
detach(package:dplyr)
```

Market Basket Analysis:

Market Basket Analysis is used to find which products do the customers buy together. This relation is then used to arrange the store according, by placing the frequently purchased products close to each other.

Important Terminologies:

Before we see the analysis, we look at some of the common terminologies used in this analysis.

Support: Percentage of transactions that contain both the itemsets together.

Confidence: Probability of having a Item B in the basket given Item A is already present in the basket

Apriori Algorithm: Algorithm to find the frequent itemsets from a dataset.

Association Rules: Statements which tell us the likeliness of a set of items going to be brought together.

We create a subset of the data frame by using ddply() function. After that, we delete the Customer ID and the Date from the created subset. We do this step as it groups the Products names (Description variable) according to the date they were purchased by a customer.

```
itemList <- ddply(data,c("CustomerID","Date"),
+                    function(df1)paste(df1$Description,
+                                       collapse = ","))
itemList$CustomerID <- NULL
itemList$Date <- NULL
colnames(itemList) <- c("items")
```

We then copy this subset into a csv file.

```
write.csv(itemList,"market_basket.csv", quote = FALSE, row.names = TRUE)
```

arules and arulesViz are the libraries required to perform Apriori algorithm.

```
library(arules)
library(arulesViz)
```

We read the transactions from the csv file and store it in form of 'baskets'.

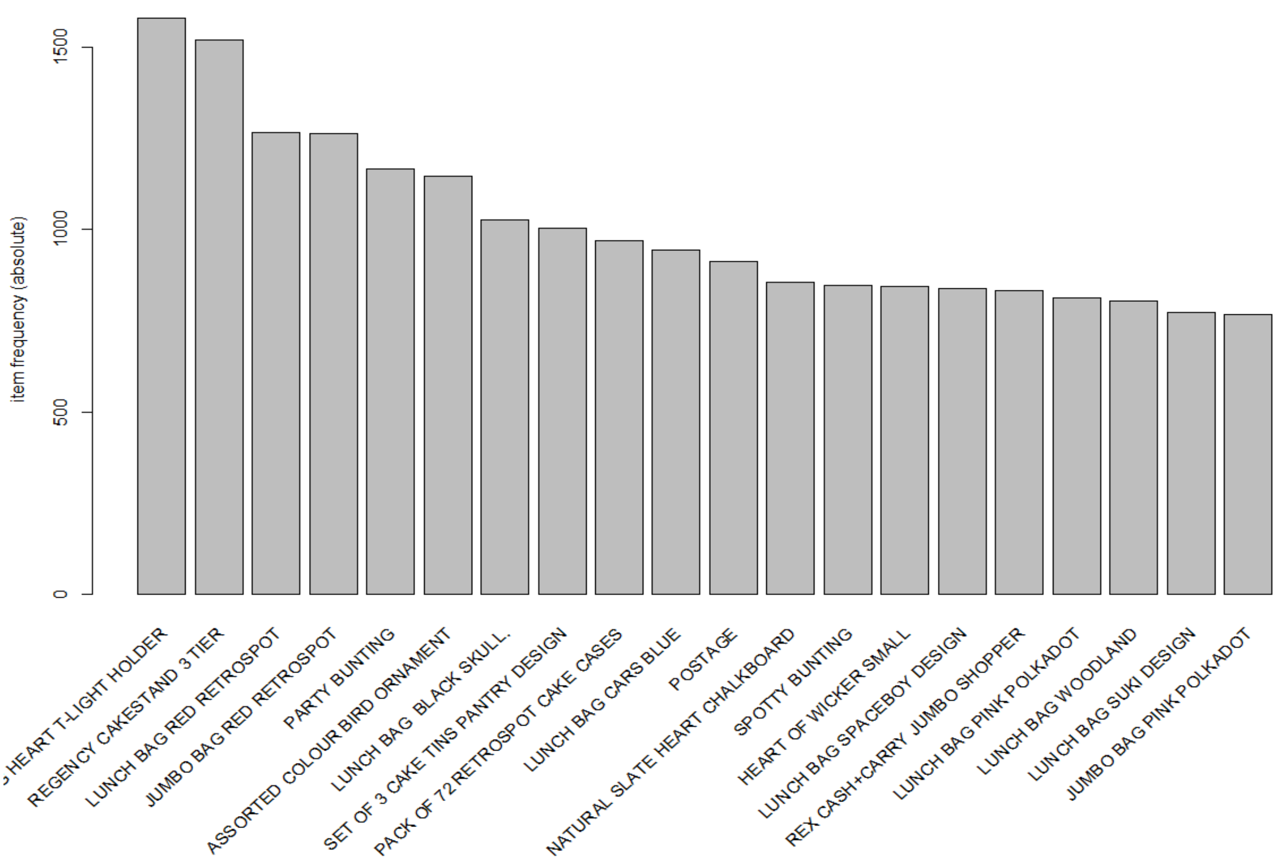
```
transactions <- read.transactions('market_basket.csv', format = 'basket', sep=',')

transactions
transactions in sparse format with
16767 transactions (rows) and
24477 items (columns)

summary(transactions)
transactions as itemMatrix in sparse format with
16767 rows (elements/itemsets/transactions) and
24477 columns (items) and a density of 0.0008277197
```

The following plot shows us the frequency of the products.

```
itemFrequencyPlot(transactions, topN=20, type='absolute')
```



We now implement Apriori Algorithm. We keep the minimum support threshold of 10% and confidence level of 80%. Keeping those thresholds, we find a set of 22 rules i.e. we find 22 itemsets which match the confidence level and support level we have set.

```
rules <- apriori(transactions, parameter = list(supp=0.01, conf=0.8))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen ta
rget   ext
        0.8    0.1    1 none FALSE            TRUE       5    0.01      1     10  r
ules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 167

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[24477 item(s), 16767 transaction(s)] done [0.05s].
sorting and recoding items ... [546 item(s)] done [0.02s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.01s].
writing ... [22 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
rules <- sort(rules, by='confidence', decreasing = TRUE)
summary(rules)
set of 22 rules

rule length distribution (lhs + rhs):sizes
 2  3  4
11  9  2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.500   2.591   3.000   4.000

summary of quality measures:
    support          confidence          lift           count
 Min.   :0.01002   Min.   :0.8027   Min.   :20.20   Min.   :168.0
 1st Qu.:0.01264   1st Qu.:0.8413   1st Qu.:24.02   1st Qu.:212.0
 Median :0.01378   Median :0.9051   Median :50.20   Median :231.0
 Mean   :0.01424   Mean   :0.9169   Mean   :45.40   Mean   :238.7
 3rd Qu.:0.01396   3rd Qu.:1.0000   3rd Qu.:56.65   3rd Qu.:234.0
 Max.   :0.02421   Max.   :1.0000   Max.   :72.58   Max.   :406.0

mining info:
 data ntransactions support confidence
   tr         16767    0.01        0.8
```

We view the top 10 rules sorted with respect to confidence value. We see that all these itemset have confidence level of 100%. The 10th itemset has a confidence value of 91.84%.

```
inspect(rules[1:10])
     lhs      rhs                                  support confidence     lift count
[1]  {SET 3 RETROSPOT TEA}                      => {SUGAR}                        0.0137
7706  1.0000000 72.58442   231
[2]  {SUGAR}                                    => {SET 3 RETROSPOT TEA}          0.0137
7706  1.0000000 72.58442   231
[3]  {SET 3 RETROSPOT TEA}                      => {COFFEE}                       0.0137
7706  1.0000000 56.64527   231
[4]  {SUGAR}                                    => {COFFEE}                       0.0137
7706  1.0000000 56.64527   231
[5]  {BACK DOOR}                                => {KEY FOB}                      0.0122
2640  1.0000000 51.27523   205
[6]  {SHED}                                     => {KEY FOB}                      0.0137
7706  1.0000000 51.27523   231
[7]  {SET 3 RETROSPOT TEA,
      SUGAR}                                    => {COFFEE}                       0.0137
7706  1.0000000 56.64527   231
[8]  {COFFEE,
      SET 3 RETROSPOT TEA}                      => {SUGAR}                        0.0137
7706  1.0000000 72.58442   231
[9]  {COFFEE,
      SUGAR}                                    => {SET 3 RETROSPOT TEA}          0.0137
7706  1.0000000 72.58442   231
[10] {SET/20 RED RETROSPOT PAPER NAPKINS,
      SET/6 RED SPOTTY PAPER CUPS}              => {SET/6 RED SPOTTY PAPER PLATES} 0.0100
7932  0.9184783 49.35938   169
```

```
detach(package:plyr)
library(dplyr)
```

Product Recommendation:

A recommendation algorithm is prepared for the users which tells the user which products he might like to buy based on his previous purchases. A sample of 100 is taken a training dataset.

```
data$Description=data.frame(data$Description)
train1 <- data.frame(data$Description[1:100,])
train2 <- as(train1, "transactions")
train3 <- as(train2, "binaryRatingMatrix")
train1 <- data$Description[1:100,]
train2 <- as(train1, "transactions")

rec <- Recommender(train3, method = "UBCF")
pre <- predict(rec, train3, n = 5)
as(pre, "list")
```

Output:

```
$`1`
[1] "data.Description.1.100...=10 COLOUR SPACEBOY PEN"       "data.Descriptio
n.1.100...=12 COLOURED PARTY BALLOONS"
[3] "data.Description.1.100...=12 DAISY PEGS IN WOOD BOX"    "data.Descriptio
n.1.100...=12 EGG HOUSE PAINTED WOOD"
[5] "data.Description.1.100...=12 HANGING EGGS HAND PAINTED"

$`2`
[1] "data.Description.1.100...=10 COLOUR SPACEBOY PEN"       "data.Descriptio
n.1.100...=12 COLOURED PARTY BALLOONS"
[3] "data.Description.1.100...=12 DAISY PEGS IN WOOD BOX"    "data.Descriptio
n.1.100...=12 EGG HOUSE PAINTED WOOD"
[5] "data.Description.1.100...=12 HANGING EGGS HAND PAINTED"
```

Product list predicted for two users.

Now we move over to exploratory analysis of the data. Here we are calculating the most frequently occurring products in the dataset. Using group_by() and summarize() functions, we create a subset of the data having the variables StockCode, Description and count. We find that the product 'White Hanging Heart T-Light Holder' occurs the most with the count of 2016. The below output shows the counts of 5 most occurring products and that of 5 least occurring products.

```
productGroup <- group_by(data, StockCode, Description)
productDescOrder <- data.table(summarize(productGroup, count = n()))
productDescOrder[order(-count)]
      StockCode                        Description count
   1:    85123A WHITE HANGING HEART T-LIGHT HOLDER  2016
   2:     22423           REGENCY CAKESTAND 3 TIER  1714
   3:    85099B            JUMBO BAG RED RETROSPOT  1615
   4:     84879      ASSORTED COLOUR BIRD ORNAMENT  1395
   5:     47566                      PARTY BUNTING  1390
  ---
3890:    90214O          LETTER "O" BLING KEY RING     1
3891:    90214T          LETTER "T" BLING KEY RING     1
3892:    90214U          LETTER "U" BLING KEY RING     1
3893:    90214W          LETTER "W" BLING KEY RING     1
3894:    90214Z          LETTER "Z" BLING KEY RING     1
```

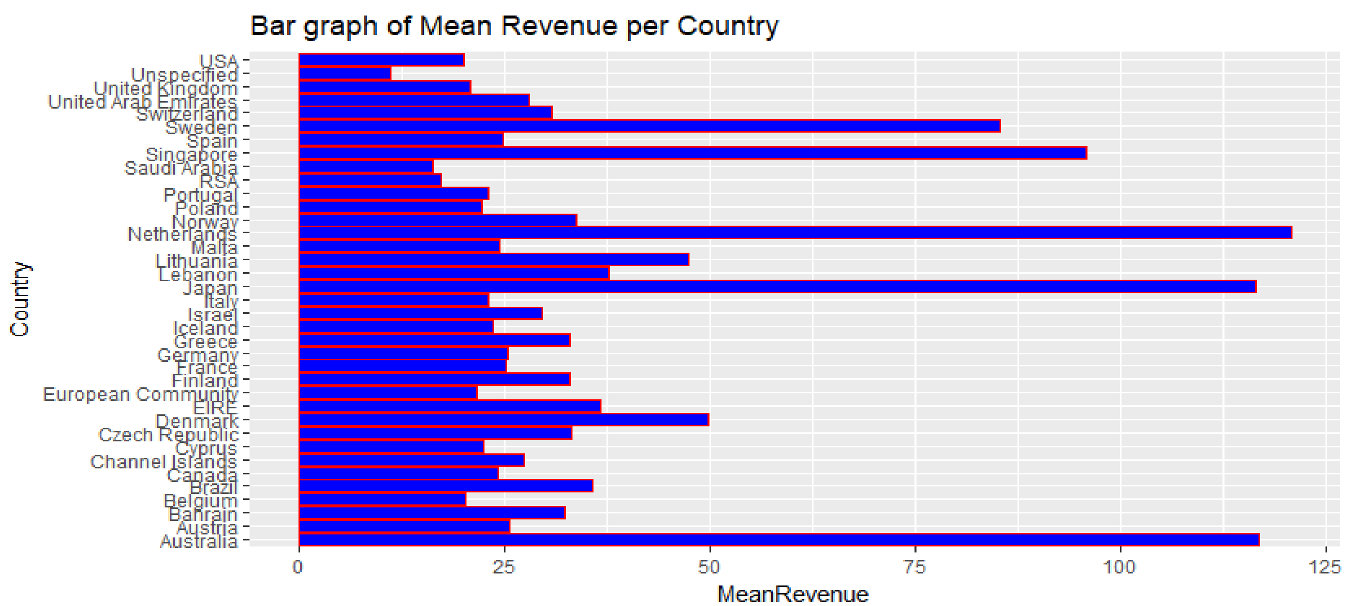
The Word Cloud shows the 10 most occurring products in the dataset. The product 'White Hanging Heart T-Light Holder' occurs the most and is hence seen in the biggest font-size.

Here, we have grouped the data into Country, Mean Revenue per country and the number of transactions for that country.

```
countriesGroup <- group_by(data, Country)
countriesDescOrder <- data.table(summarize(countriesGroup, MeanRevenue = mean(Total
Cost),count = n()))
countriesDescOrder[order(-MeanRevenue)]
                 Country MeanRevenue  count
 1:          Netherlands   120.79828   2363
 2:            Australia   116.93734   1184
 3:                Japan   116.56190    321
 4:            Singapore    95.85266    222
 5:               Sweden    85.26184    450
 6:              Denmark    49.88247    380
 7:            Lithuania    47.45886     35
 8:              Lebanon    37.64178     45
 9:                 EIRE    36.69929   7228
10:               Brazil    35.73750     32
11:               Norway    33.73642   1072
12:       Czech Republic    33.06960     25
13:              Finland    32.91399    685
14:               Greece    32.83117    145
15:              Bahrain    32.25882     17
16:          Switzerland    30.64275   1842
17:               Israel    29.45241    245
18: United Arab Emirates    27.97471     68
19:      Channel Islands    27.36351    747
20:              Austria    25.62482    398
21:              Germany    25.33271   9027
22:               France    25.09119   8327
23:                Spain    24.82200   2480
24:                Malta    24.33563    112
25:               Canada    24.28066    151
26:              Iceland    23.68132    182
27:                Italy    23.06496    758
28:             Portugal    22.97030   1453
29:               Cyprus    22.39279    603
30:               Poland    22.22621    330
31:   European Community    21.67083     60
32:       United Kingdom    20.86043 349227
33:              Belgium    20.28377   2031
34:                  USA    20.00218    179
35:                  RSA    17.28121     58
36:         Saudi Arabia    16.21333      9
37:          Unspecified    11.04054    241
                 Country MeanRevenue  count
```

The plot shows us the mean revenue of all the 37 countries present in the dataset. We find that even th ough the number of transactions in United Kingdom is the highest, the mean revenue is one of the least. From this, we can say that the range of the Unit Price of products purchased in UK is far less than that in countries like Netherlands, Australia and Lebanon.

```
ggplot(data=countriesDescOrder, aes(x=Country, y=MeanRevenue)) +
+   geom_bar(stat="identity", color = "red", fill = "blue") +
+   coord_flip() +
+   ggtitle("Bar graph of Mean Revenue per Country")
```



This plot shows us the number of transactions per country. We see that United Kingdom has the highest number of transactions in that year in the range of 35000.

```
ggplot(data=countriesDescOrder, aes(x=Country, y=count)) +
+   geom_bar(stat="identity", color = "red", fill="blue") +
+   coord_flip() +
+   ggtitle("Bar graph of Transactions per Country")
```