Customer Segmentation using RFM analysis and Clustering

Name:Mahesh Jayaraman

Customer segmentation is the practive of dividing a company's customers into groups that reflects similarity among customers in each group.We've chosen a UK based online retail dataset which was collected between dec 2010 to dec 2011. The dataset contains 8 columns and over 500,000 rows.The goal of segmenting customers will helps the company to decide how they can market their products and how to appeal the respective groups.

#Load necessary packages

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.0.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
require(tigerstats)
```

```
## Loading required package: tigerstats
```

```
## Loading required package: abd
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## Loading required package: lattice
```

```
## Loading required package: grid
```

```
## Loading required package: mosaic
```

```
## Loading required package: ggformula
```

```
## Loading required package: ggstance
```

```
##
## Attaching package: 'ggstance'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
```

```
##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")
```

```
## Loading required package: mosaicData
```

```
## Loading required package: Matrix
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by thi
s.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.
##
## Have you tried the ggformula package for your plots?
```

```
##
## Attaching package: 'mosaic'
```

```
## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following object is masked from 'package:ggplot2':
##
##     stat
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
```

```
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
```

```
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
## Welcome to tigerstats!
## To learn more about this package, consult its website:
##  http://homerhanumat.github.io/tigerstats
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------
-- tidyverse 1.3.0 --
```

```
## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tid
yverse_conflicts() --
## x nlme::collapse()           masks dplyr::collapse()
## x mosaic::count()            masks dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x mosaic::do()               masks dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x dplyr::filter()            masks stats::filter()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x dplyr::lag()               masks stats::lag()
## x tidyr::pack()              masks Matrix::pack()
## x mosaic::stat()             masks ggplot2::stat()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
```

```
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## The following object is masked from 'package:mosaic':
##
##     dotPlot
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
## The following objects are masked from 'package:mosaic':
##
##     deltaMethod, logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(moments)
```

```
## Warning: package 'moments' was built under R version 4.0.3
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(agricolae)
```

```
## Warning: package 'agricolae' was built under R version 4.0.5
```

```
##
## Attaching package: 'agricolae'
```

```
## The following objects are masked from 'package:moments':
##
##     kurtosis, skewness
```

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.0.3
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.0.3
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:viridis':
##
##     viridis_pal
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## The following object is masked from 'package:mosaic':
##
##     rescale
```

```
library(DT)
```

```
## Warning: package 'DT' was built under R version 4.0.5
```

```
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.5
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:mosaic':
##
##     do
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.0.5
```

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.5
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
##
## Attaching package: 'factoextra'
```

```
## The following object is masked from 'package:agricolae':
##
##     hcut
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.4
```

```
library(rpart.plot)
```

#Load the data

```
data<-read.csv("C:/Users/ADMIN/Downloads/project/project.csv")
names(data)
```

```
## [1] "InvoiceNo"   "StockCode"   "Description" "Quantity"    "InvoiceDate"
## [6] "UnitPrice"   "CustomerID"  "Country"
```

```
head(data)
```

```
##   InvoiceNo StockCode                          Description Quantity
## 1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6
## 2    536365     71053                 WHITE METAL LANTERN        6
## 3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8
## 4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6
## 5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6
## 6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2
##      InvoiceDate UnitPrice CustomerID        Country
## 1 12/1/2010 8:26      2.55      17850 United Kingdom
## 2 12/1/2010 8:26      3.39      17850 United Kingdom
## 3 12/1/2010 8:26      2.75      17850 United Kingdom
## 4 12/1/2010 8:26      3.39      17850 United Kingdom
## 5 12/1/2010 8:26      3.39      17850 United Kingdom
## 6 12/1/2010 8:26      7.65      17850 United Kingdom
```

```
str(data)
```

```
## 'data.frame':    541909 obs. of  8 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPID
HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER BOTTLE" ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: chr  "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
```

```
summary(data)
```

```
##   InvoiceNo          StockCode         Description          Quantity
##  Length:541909      Length:541909      Length:541909      Min.   :-80995.00
##  Class :character   Class :character   Class :character   1st Qu.:     1.00
##  Mode  :character   Mode  :character   Mode  :character   Median :     3.00
##                                                           Mean   :     9.55
##                                                           3rd Qu.:    10.00
##                                                           Max.   : 80995.00
##
##  InvoiceDate          UnitPrice           CustomerID       Country
##  Length:541909      Min.   :-11062.06   Min.   :12346    Length:541909
##  Class :character   1st Qu.:     1.25   1st Qu.:13953    Class :character
##  Mode  :character   Median :     2.08   Median :15152    Mode  :character
##                     Mean   :     4.61   Mean   :15288
##                     3rd Qu.:     4.13   3rd Qu.:16791
##                     Max.   : 38970.00   Max.   :18287
##                                         NA's   :135080
```

```
sum(is.na(data))
```

```
## [1] 135080
```

```
typeof(data)
```

```
## [1] "list"
```

```
data <- na.omit(data)
dim(data)
```

```
## [1] 406829      8
```

```
sum(is.na(data))
```

```
## [1] 0
```

our datset contains 8 columns and by eyeing the structure and summary of the set, we can see the that it requires some cleaning and the InvoiceDate column has the date and time combined which must be split for further analysis. The cleaning is done and we are left with some 400,000 rows.

##separate date and time. The date is separated into month, week and hour.

```
data$date <- sapply(data$InvoiceDate, FUN = function(x) {strsplit(x, split = '[ ]')[[1]][1]})
data$time <- sapply(data$InvoiceDate, FUN = function(x) {strsplit(x, split = '[ ]')[[1]][2]})

data$month <- sapply(data$date, FUN = function(x) {strsplit(x, split = '[/]')[[1]][1]})
data$year <- sapply(data$date, FUN = function(x) {strsplit(x, split = '[/]')[[1]][3]})
data$hourOfDay <- sapply(data$time, FUN = function(x) {strsplit(x, split = '[:]')[[1]][1]})
head(data, n =5)
```

```
##   InvoiceNo StockCode                         Description Quantity
## 1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6
## 2    536365     71053                 WHITE METAL LANTERN        6
## 3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8
## 4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6
## 5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6
##      InvoiceDate UnitPrice CustomerID        Country      date time month year
## 1 12/1/2010 8:26      2.55      17850 United Kingdom 12/1/2010 8:26    12 2010
## 2 12/1/2010 8:26      3.39      17850 United Kingdom 12/1/2010 8:26    12 2010
## 3 12/1/2010 8:26      2.75      17850 United Kingdom 12/1/2010 8:26    12 2010
## 4 12/1/2010 8:26      3.39      17850 United Kingdom 12/1/2010 8:26    12 2010
## 5 12/1/2010 8:26      3.39      17850 United Kingdom 12/1/2010 8:26    12 2010
##   hourOfDay
## 1         8
## 2         8
## 3         8
## 4         8
## 5         8
```

extra columns with date, time, month, year and hour is added.

converting the date variable to an appropriate class and we add an extra column for sale=quantity*price

```
data$date <- as.Date(data$date, "%m/%d/%Y")
data$dayOfWeek <- wday(data$date, label=TRUE)
data <- data %>% mutate(sale = Quantity * UnitPrice)
data_pos<-data %>% filter(sale>0)
data_neg <- data %>% filter (sale<=0)
head(data)
```

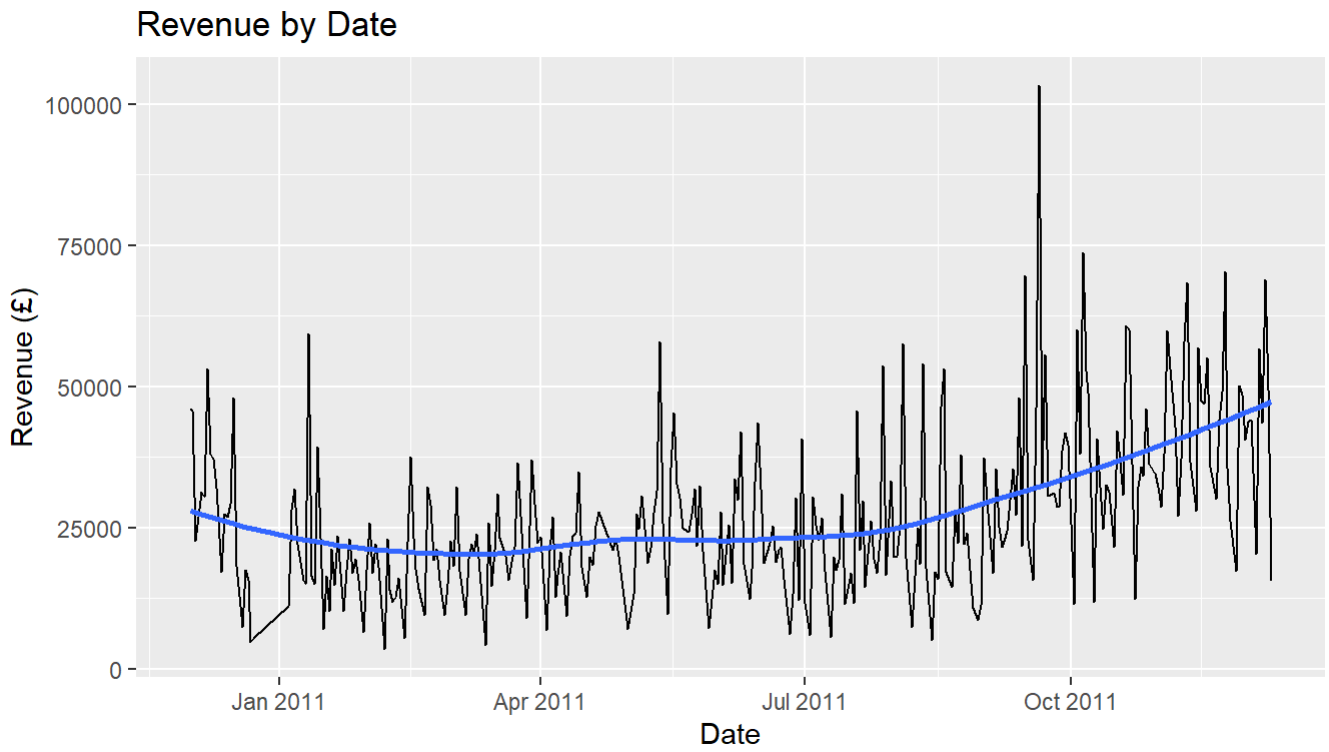
```
##   InvoiceNo StockCode                         Description Quantity
## 1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6
## 2    536365     71053                 WHITE METAL LANTERN        6
## 3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8
## 4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6
## 5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6
## 6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2
##      InvoiceDate UnitPrice CustomerID        Country       date time month year
## 1 12/1/2010 8:26      2.55      17850 United Kingdom 2010-12-01 8:26    12 2010
## 2 12/1/2010 8:26      3.39      17850 United Kingdom 2010-12-01 8:26    12 2010
## 3 12/1/2010 8:26      2.75      17850 United Kingdom 2010-12-01 8:26    12 2010
## 4 12/1/2010 8:26      3.39      17850 United Kingdom 2010-12-01 8:26    12 2010
## 5 12/1/2010 8:26      3.39      17850 United Kingdom 2010-12-01 8:26    12 2010
## 6 12/1/2010 8:26      7.65      17850 United Kingdom 2010-12-01 8:26    12 2010
##   hourOfDay dayOfWeek  sale
## 1         8       Wed 15.30
## 2         8       Wed 20.34
## 3         8       Wed 22.00
## 4         8       Wed 20.34
## 5         8       Wed 20.34
## 6         8       Wed 15.30
```

converting appropriate columns using as.factor

```
data$Country <- as.factor(data$Country)
data$month <- as.factor(data$month)
data$year <- as.factor(data$year)
levels(data$year) <- c(2010,2011)
data$hourOfDay <- as.factor(data$hourOfDay)
data$dayOfWeek <- as.factor(data$dayOfWeek)

data %>%
  group_by(date) %>%
  summarise(revenue = sum(sale)) %>%
  ggplot(aes(x = date, y = revenue)) + geom_line() + geom_smooth(method = 'auto', se = FALSE) +
  labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
```

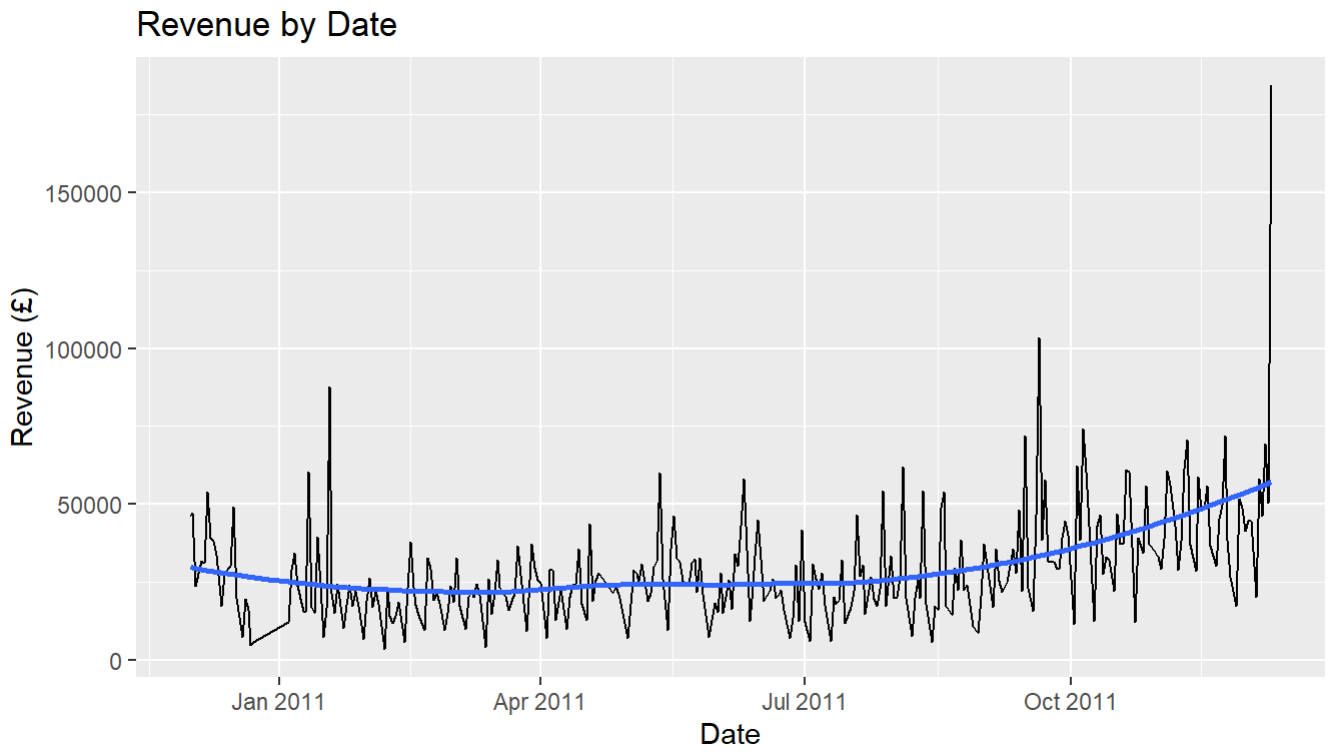
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



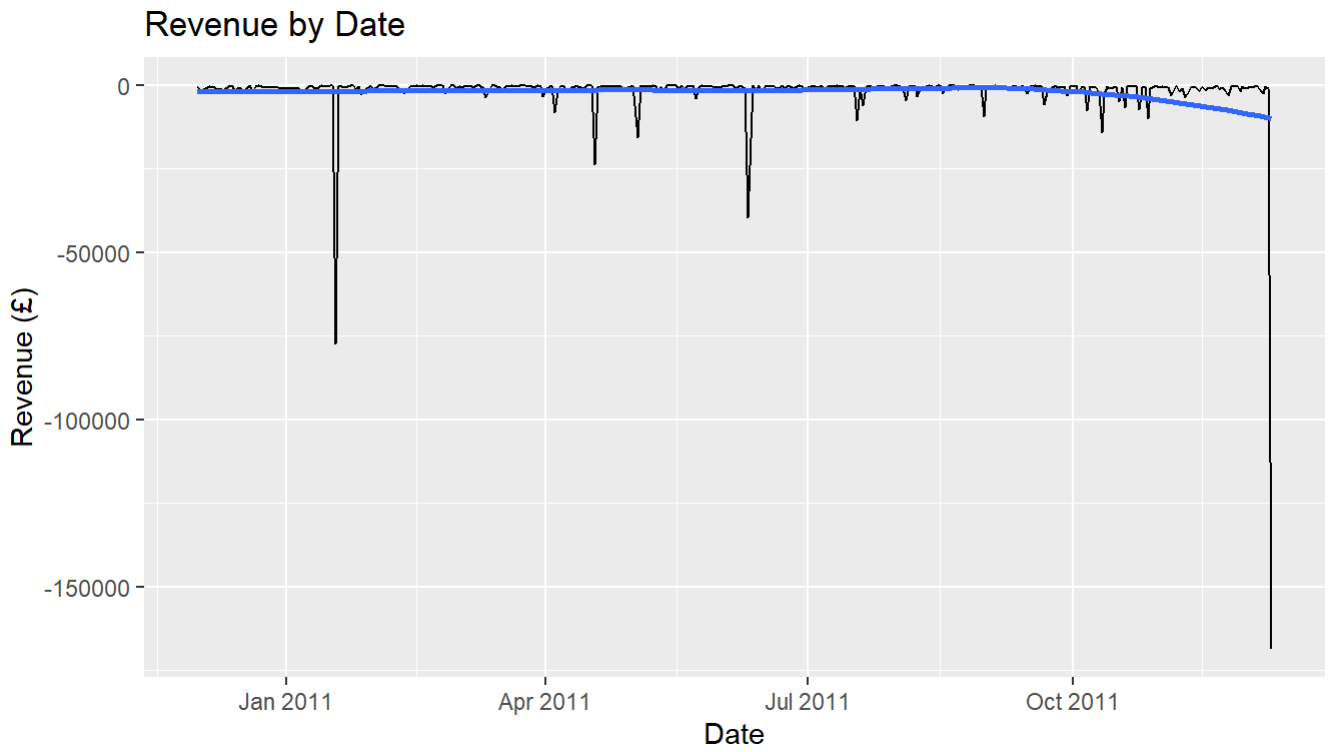

```
#positive revenue
data_pos %>%
  group_by(date) %>%
  summarise(revenue = sum(sale)) %>%
  ggplot(aes(x = date, y = revenue)) + geom_line() + geom_smooth(method = 'auto', se = FALSE) +
  labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
#negative revenue
data_neg %>%
  group_by(date) %>%
  summarise(revenue = sum(sale)) %>%
  ggplot(aes(x = date, y = revenue)) + geom_line() + geom_smooth(method = 'auto', se = FALSE) +
  labs(x = 'Date', y = 'Revenue (£)', title = 'Revenue by Date')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

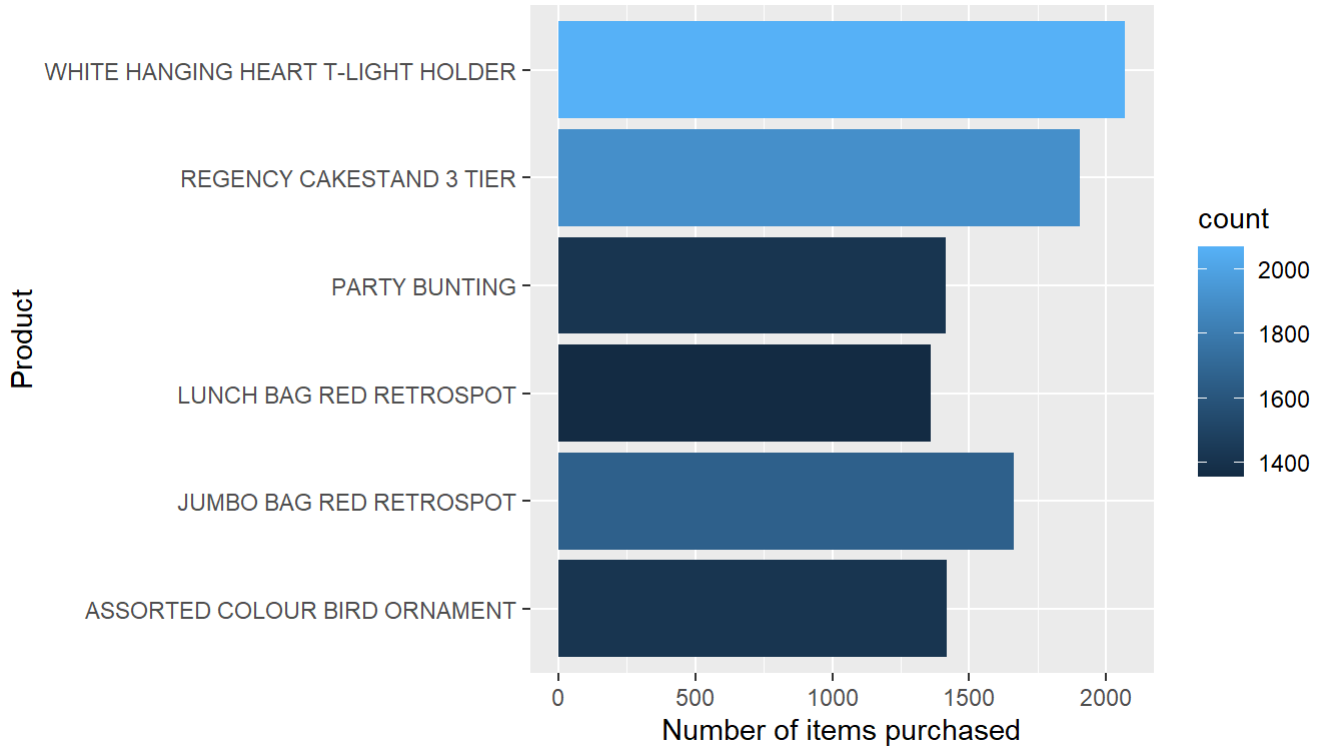


we can notice the trends in the first graph- date vs revenue. We further split our revenue into positive and negative, and we post the graph for the same, just by seeing the third graph we can see there are some refunds in the set(negative revenue).

#visualising the description column

```
#product
data %>% group_by(StockCode, Description) %>% summarise(count= n()) %>% arrange(desc(count)) %>%
  head() %>%
ggplot(aes(x=Description, y=count, fill = count)) + geom_bar(stat= "identity") + coord_flip() +
       labs(y="Number of items purchased", x="Product")
```

```
## `summarise()` regrouping output by 'StockCode' (override with `.groups` argument)
```



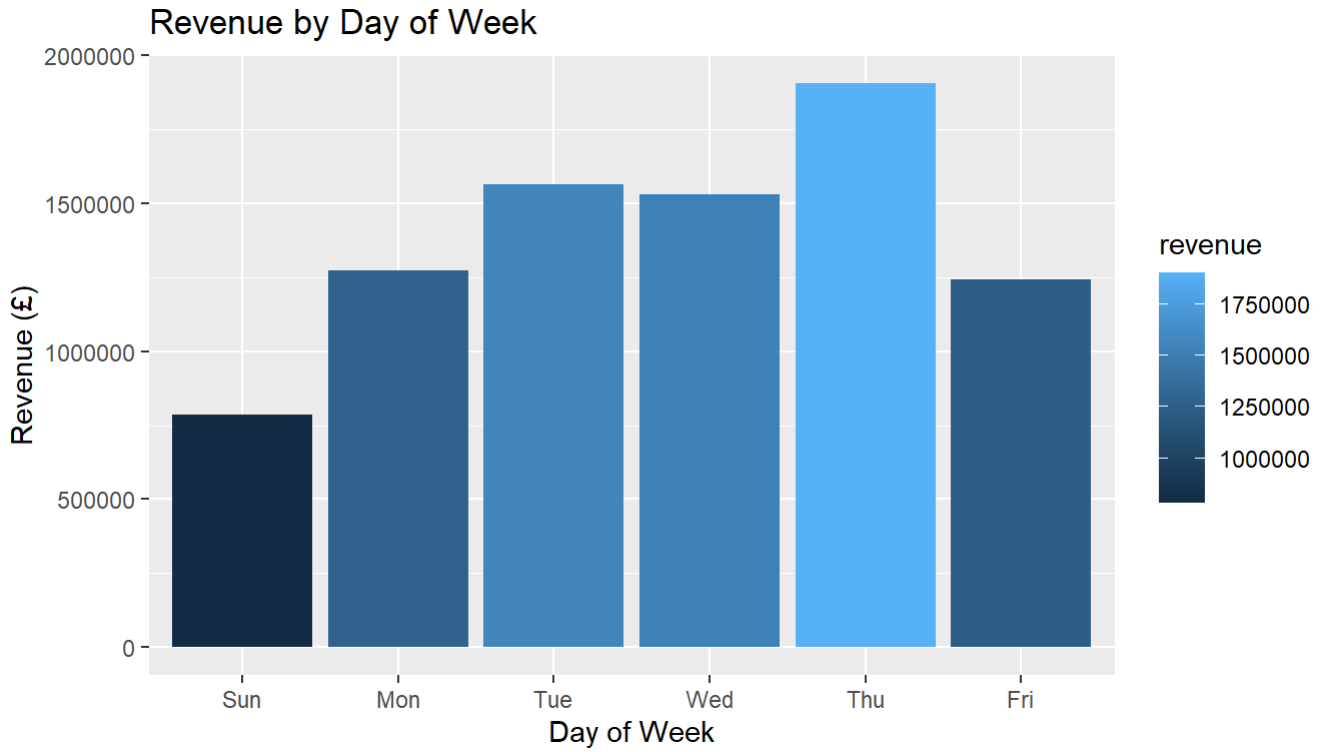
just by looking at the count we can see that the light holder sits at the top.

###day of the week analysis

we factor the day of the week and assign each day a number. since it is believed that customers mindset changes as the week goes on, this following snippet will offer a brief insight

```
#tuesday and wednesday generates equal revenue(almost) but the interesting trend is that thur ba
  r is higher than the fri bar.
data %>%
  group_by(dayOfWeek) %>%
  summarise(revenue = sum(sale)) %>%
  ggplot(aes(x = dayOfWeek, y = revenue,fill=revenue)) + geom_col() + labs(x = 'Day of Week', y
   = 'Revenue (£)', title = 'Revenue by Day of Week')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Tue and Wed generates almost equal revenue and to our surprise, sunday generates the lowest revenue among the lot while thursday sits at the top.

#creating a new dataframe to notice the days trend

```
weekdaySummary <- data %>%
  group_by(date, dayOfWeek) %>%
  summarise(revenue = sum(sale), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup()
```

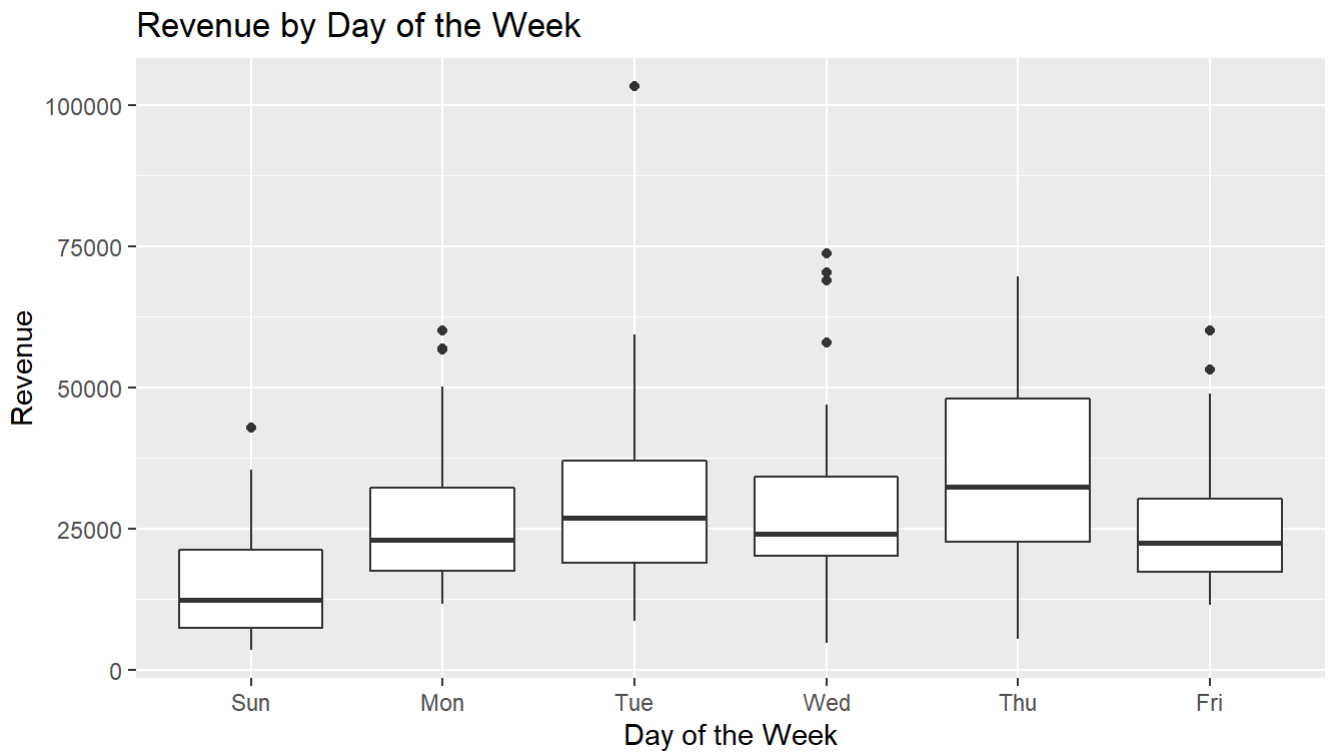
```
## `summarise()` regrouping output by 'date' (override with `.groups` argument)
```

```
head(weekdaySummary, n = 10)
```

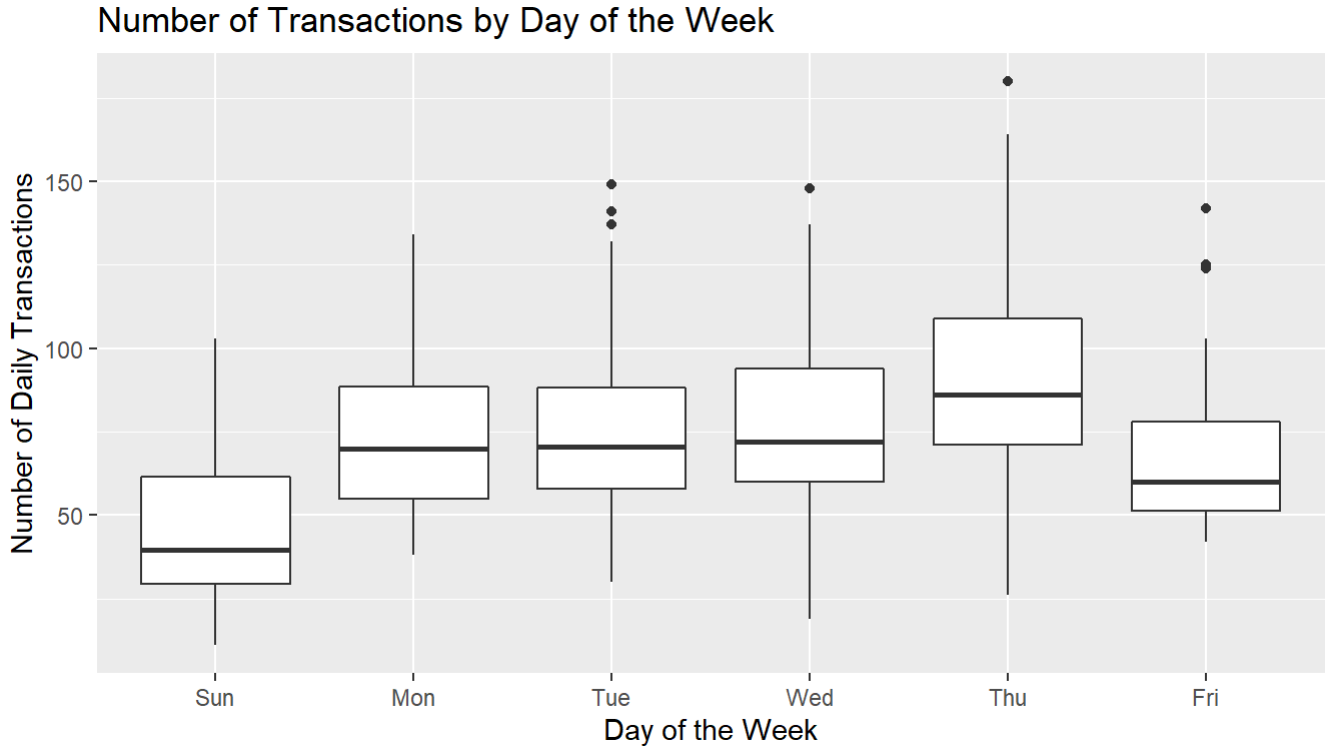
```
## # A tibble: 10 x 5
##    date       dayOfWeek revenue transactions aveOrdVal
##    <date>     <ord>       <dbl>        <int>     <dbl>
##  1 2010-12-01 Wed        46051.          127      363.
##  2 2010-12-02 Thu        45775.          160      286.
##  3 2010-12-03 Fri        22598.           64      353.
##  4 2010-12-05 Sun        31381.           94      334.
##  5 2010-12-06 Mon        30465.          111      274.
##  6 2010-12-07 Tue        53126.           79      672.
##  7 2010-12-08 Wed        38049.          134      284.
##  8 2010-12-09 Thu        37178.          132      282.
##  9 2010-12-10 Fri        32005.           78      410.
## 10 2010-12-12 Sun        17218.           50      344.
```

now we can see what's happening on each day with respective revenue and dates.

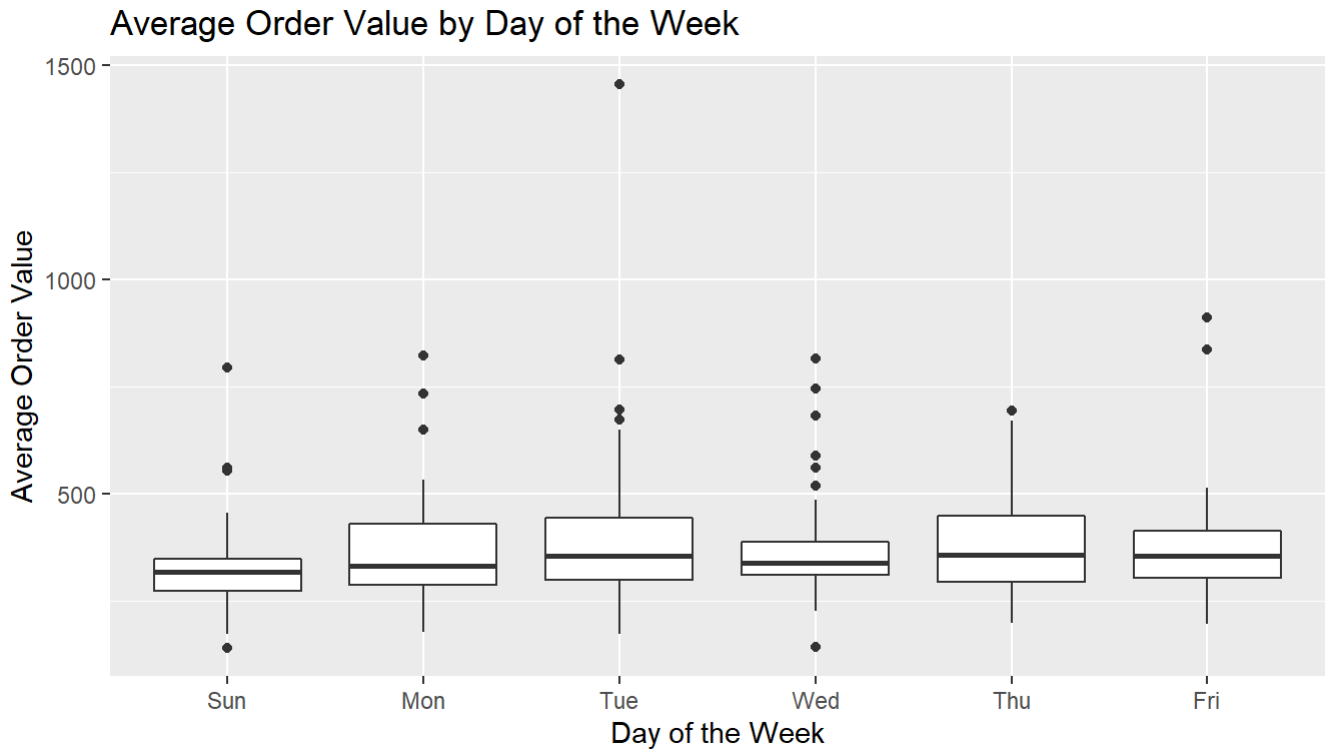
```
ggplot(weekdaySummary, aes(x = dayOfWeek, y = revenue)) + geom_boxplot() + labs(x = 'Day of the
  Week', y = 'Revenue', title = 'Revenue by Day of the Week')
```




```
ggplot(weekdaySummary, aes(x = dayOfWeek, y = transactions)) + geom_boxplot() + labs(x = 'Day of
  the Week', y = 'Number of Daily Transactions', title = 'Number of Transactions by Day of the W
  eek')
```



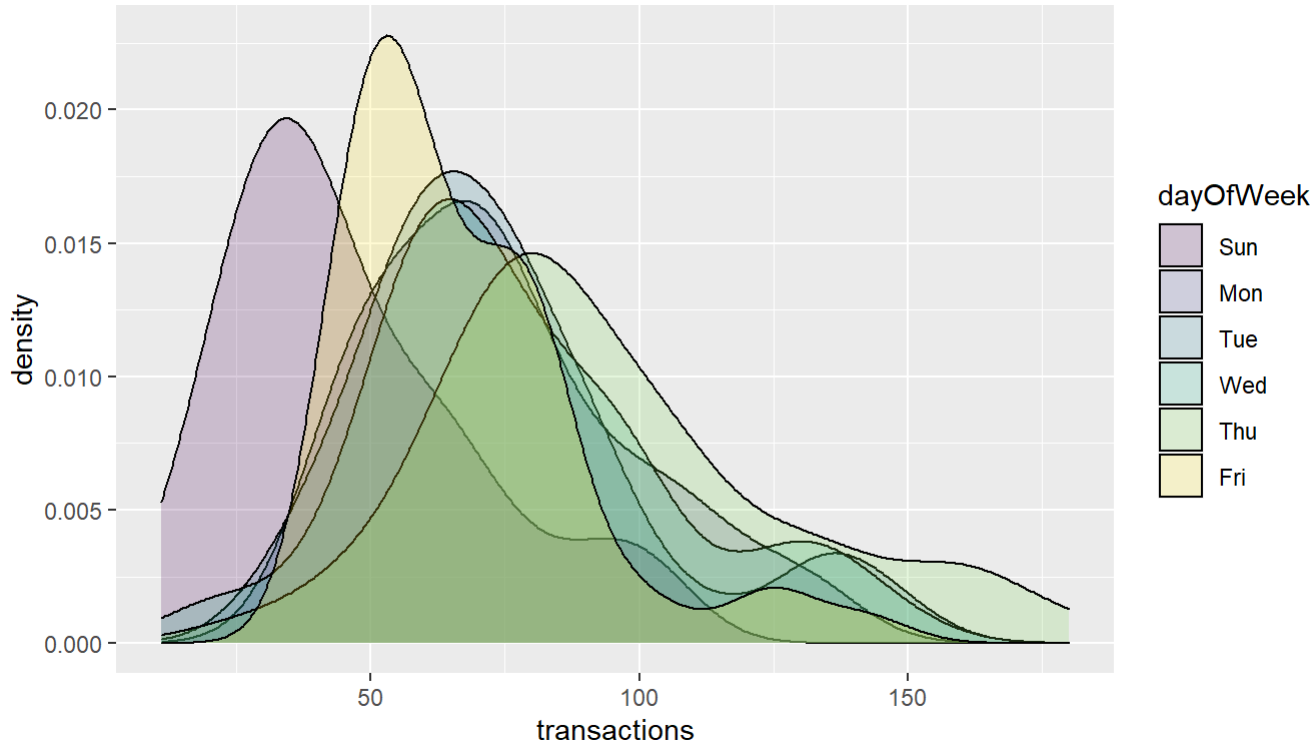
```
ggplot(weekdaySummary, aes(x = dayOfWeek, y = aveOrdVal)) + geom_boxplot() + labs(x = 'Day of th
  e Week', y = 'Average Order Value', title = 'Average Order Value by Day of the Week')
```



There are differences in the amount of revenue on each day of the week, and the said difference is driven by a difference in the number of transactions rather than the average order value.

a reasonable amount of skewness present in our data so we will use non parametric test for statistically significant differences in our data

```
ggplot(weekdaySummary, aes(transactions, fill = dayOfWeek)) + geom_density(alpha = 0.2)
```




```
kruskal.test(transactions ~ dayOfWeek, data = weekdaySummary)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  transactions by dayOfWeek
## Kruskal-Wallis chi-squared = 71.744, df = 5, p-value = 4.441e-14
```

that's quite a p value so now we can see which days are significantly diff from others

```
kruskal(weekdaySummary$transactions, weekdaySummary$dayOfWeek, console = TRUE)
```

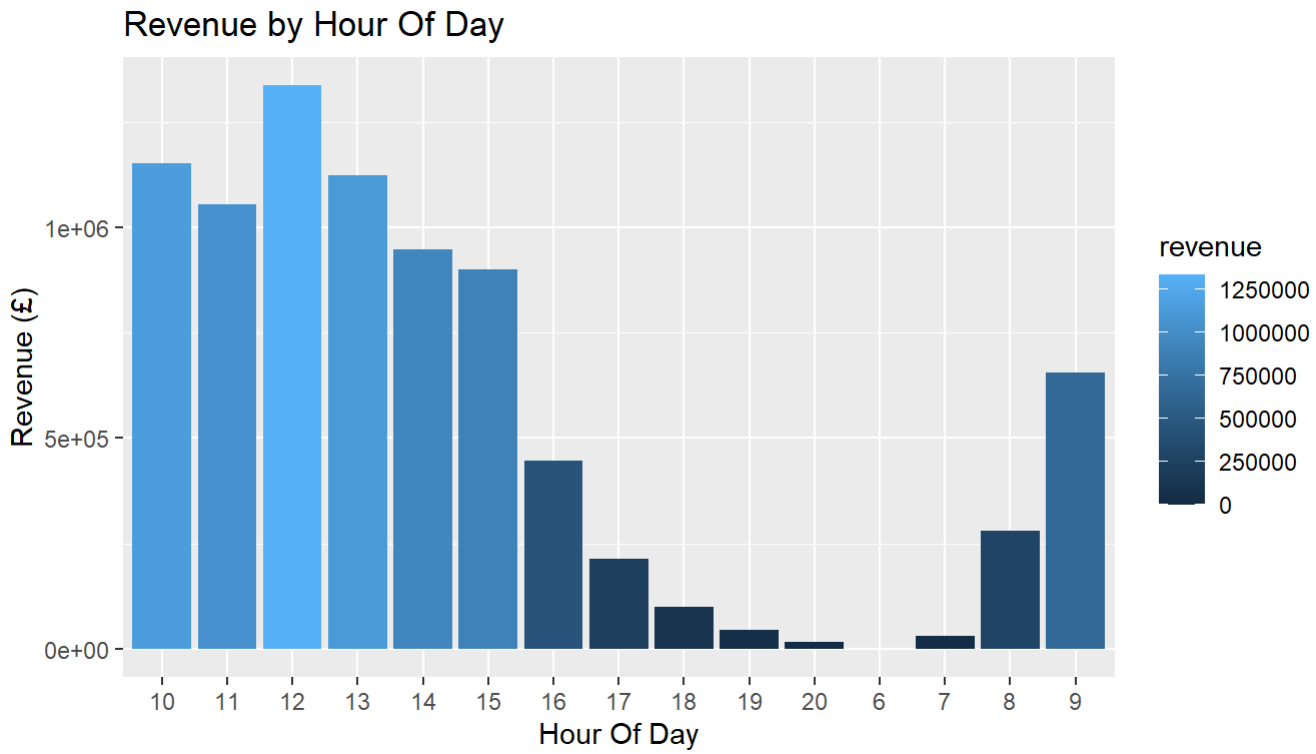
```
##
## Study: weekdaySummary$transactions ~ weekdaySummary$dayOfWeek
## Kruskal-Wallis test's
## Ties or no Ties
##
## Critical Value: 71.7443
## Degrees of freedom: 5
## Pvalue Chisq  : 4.440892e-14
##
## weekdaySummary$dayOfWeek,  means of the ranks
##
##     weekdaySummary.transactions  r
## Fri                   135.0100 50
## Mon                   162.4574 47
## Sun                    72.3600 50
## Thu                   213.5000 53
## Tue                   160.0769 52
## Wed                   170.2170 53
##
## Post Hoc Analysis
##
## t-Student: 1.96793
## Alpha    : 0.05
## Groups according to probability of treatment differences and alpha level.
##
## Treatments with the same letter are not significantly different.
##
##     weekdaySummary$transactions groups
## Thu                   213.5000      a
## Wed                   170.2170      b
## Mon                   162.4574     bc
## Tue                   160.0769     bc
## Fri                   135.0100      c
## Sun                    72.3600      d
```

as we can see sunday has the lowest number of transactions and thur has the highest.As the avg order value remains somewhat consistent, this transaltes to differences in revenue. we can suggest that ads can be circulated more on thur rather than sunday but that might be a hasty decision without full analysis.

###hour of the day analysis

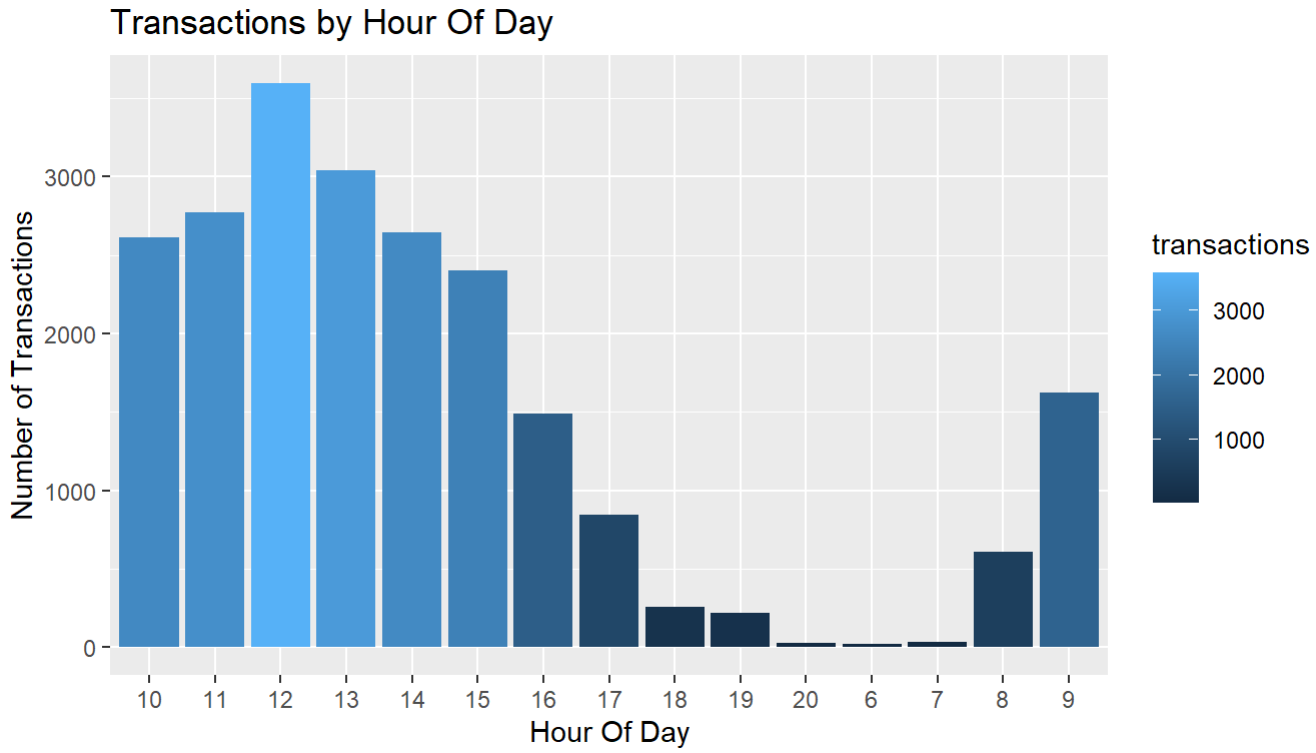
```
#revenue by hour
data %>%
  group_by(hourOfDay) %>%
  summarise(revenue = sum(sale)) %>%
  ggplot(aes(x = hourOfDay, y = revenue,fill=revenue)) + geom_col() + labs(x = 'Hour Of Day', y
   = 'Revenue (£)', title = 'Revenue by Hour Of Day')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```




```
#transaction by hour
data %>%
  group_by(hourOfDay) %>%
  summarise(transactions = n_distinct(InvoiceNo)) %>%
  ggplot(aes(x = hourOfDay, y = transactions,fill=transactions)) + geom_col() + labs(x = 'Hour O
  f Day', y = 'Number of Transactions', title = 'Transactions by Hour Of Day')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



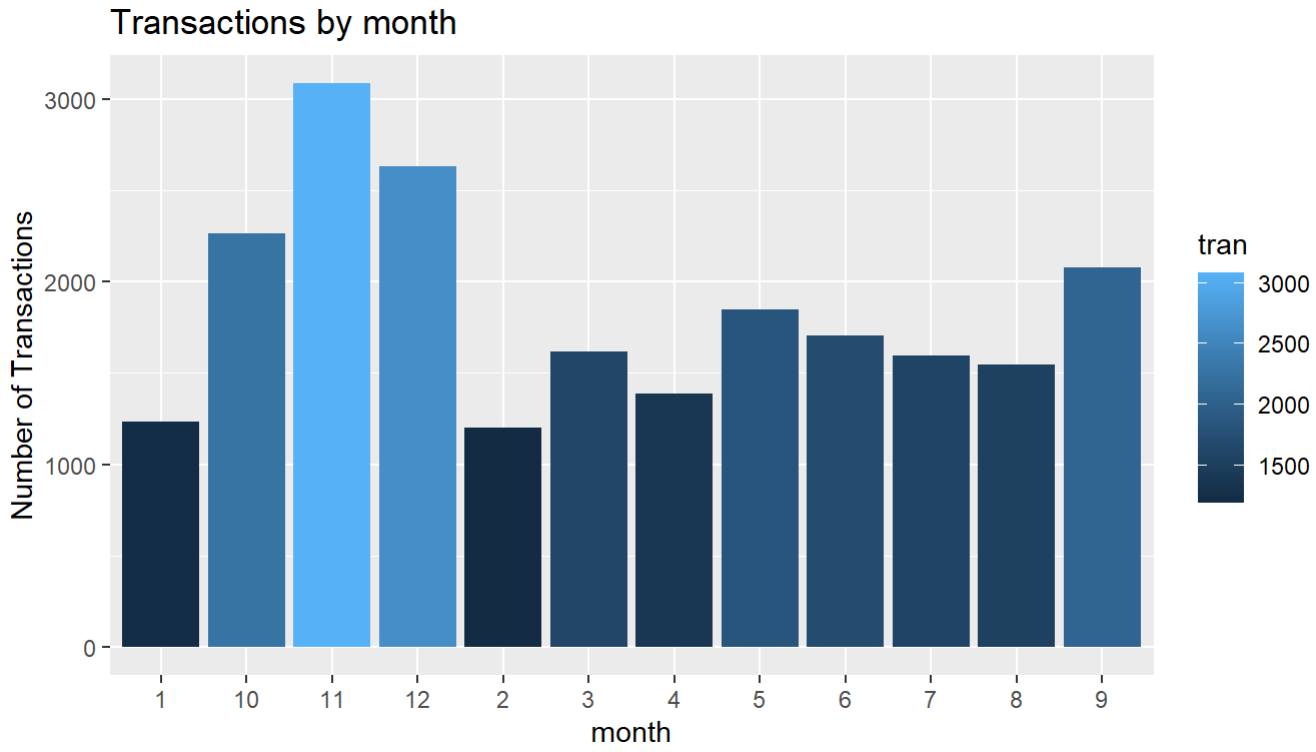
the transactions are more in the morning to mid afternoon period and a fall in the evening period. Some of the hours are missing.

#visualising month vs number of transaction

```
#transaction vs month
data %>%group_by(month, dayOfWeek, hourOfDay) %>%
summarise(revenue = sum(sale),
          transactions = n_distinct(InvoiceNo)) %>%
group_by(month) %>%
summarise(tran = sum(transactions)) %>%
ggplot(aes(x=month, y = tran)) +
geom_col(aes(fill = tran))+labs(x = 'month', y = 'Number of Transactions', title = 'Transactions
  by month')
```

```
## `summarise()` regrouping output by 'month', 'dayOfWeek' (override with `.groups` argument)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



october, november and december seems to be busiest months of the year.

###country summary

```
countrySummary <- data %>%
  group_by(Country) %>%
  summarise(revenue = sum(sale), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))   #averageordervalue
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(countrySummary, n = 15)
```

```
## # A tibble: 15 x 4
##    Country          revenue transactions aveOrdVal
##    <fct>              <dbl>        <int>     <dbl>
##  1 United Kingdom  6767873.        19857      341.
##  2 Netherlands      284662.          101     2818.
##  3 EIRE             250285.          319      785.
##  4 Germany          221698.          603      368.
##  5 France           196713.          458      430.
##  6 Australia        137077.           69     1987.
##  7 Switzerland       55739.           71      785.
##  8 Spain             54775.          105      522.
##  9 Belgium           40911.          119      344.
## 10 Sweden            36596.           46      796.
## 11 Japan             35341.           28     1262.
## 12 Norway            35163.           40      879.
## 13 Portugal          29060.           70      415.
## 14 Finland           22327.           48      465.
## 15 Channel Islands   20086.           33      609.
```

```
unique(countrySummary$Country)
```

```
##  [1] United Kingdom       Netherlands          EIRE
##  [4] Germany              France               Australia
##  [7] Switzerland          Spain                Belgium
## [10] Sweden               Japan                Norway
## [13] Portugal             Finland              Channel Islands
## [16] Denmark              Italy                Cyprus
## [19] Austria              Singapore            Poland
## [22] Israel               Greece               Iceland
## [25] Canada               Unspecified          Malta
## [28] United Arab Emirates USA                  Lebanon
## [31] Lithuania            European Community   Brazil
## [34] RSA                  Czech Republic       Bahrain
## [37] Saudi Arabia
## 37 Levels: Australia Austria Bahrain Belgium Brazil Canada ... USA
```

```
countryCustSummary <- data %>%
  group_by(Country) %>%
  summarise(revenue = sum(sale), customers = n_distinct(CustomerID)) %>%
  mutate(aveCustVal = (round((revenue / customers),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))  #averagecustomervalue
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(countryCustSummary, n = 15)
```

```
## # A tibble: 15 x 4
##    Country          revenue customers aveCustVal
##    <fct>              <dbl>     <int>      <dbl>
##  1 United Kingdom  6767873.      3950      1713.
##  2 Netherlands      284662.         9     31629.
##  3 EIRE             250285.         3     83428.
##  4 Germany          221698.        95      2334.
##  5 France           196713.        87      2261.
##  6 Australia        137077.         9     15231.
##  7 Switzerland       55739.        21      2654.
##  8 Spain             54775.        31      1767.
##  9 Belgium           40911.        25      1636.
## 10 Sweden            36596.         8      4574.
## 11 Japan             35341.         8      4418.
## 12 Norway            35163.        10      3516.
## 13 Portugal          29060.        19      1529.
## 14 Finland           22327.        12      1861.
## 15 Channel Islands   20086.         9      2232.
```

UK sits at top obviously, the customervalue varies because the amount of cusomters. We can pull out UK and try k means clustering.

###clustering ###RFM

recency is calculated as one of the features for the segmentation analysis. In this case, time of customer's last purchase minus the last transaction date in days.

```
UK_data <- data %>%
  filter(Country == 'United Kingdom')

Users_Recency <- UK_data %>%
  group_by(CustomerID) %>%
  summarise(Last_Customer_Activity = max(date)) %>%
  mutate(Last_Invoice = max(Last_Customer_Activity))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Users_Recency$Recency<- round(as.numeric(difftime(Users_Recency$Last_Invoice, Users_Recency$Last
  _Customer_Activity , units = c("days"))))
Users_Recency <- Users_Recency %>%
  select(CustomerID, Recency)
print(summary(Users_Recency$Recency))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   16.00   50.00   91.32  143.00  373.00
```

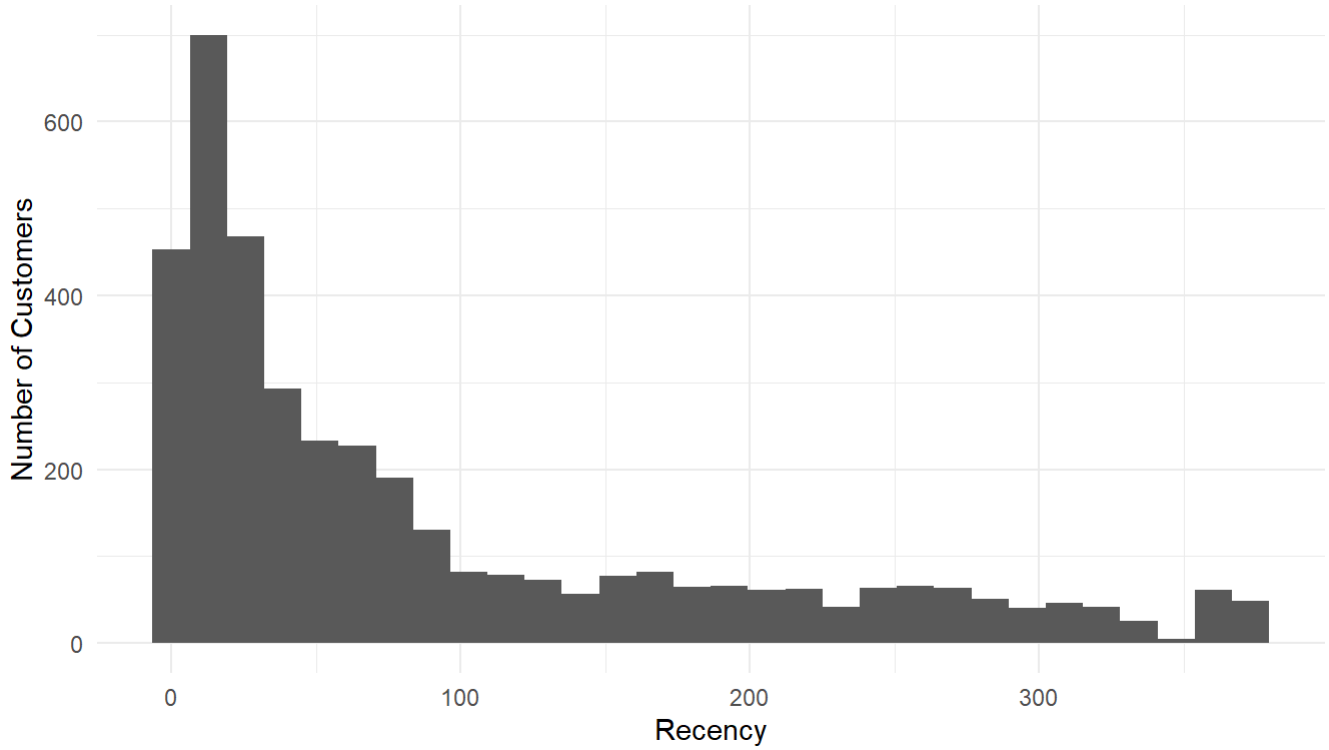
Recency summary: average 3 months wihtout making a single purchase and a small group of people have not made a singlr transaction in over a year. below 50 or so days of inactivity (50 percent of the customers).

```
head(Users_Recency)
```

```
## # A tibble: 6 x 2
##   CustomerID Recency
##        <int>   <dbl>
## 1      12346     325
## 2      12747       2
## 3      12748       0
## 4      12749       3
## 5      12820       3
## 6      12821     214
```

```
ggplot(Users_Recency, aes(Recency)) +
  geom_histogram() +
  ylab('Number of Customers') +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



the graph translates the same, where we have some customers with no activity over an year, the average is less than 3 months, and 50% percent of the customers with < 50days of inactivity.

Frequency is calculated by counting number of times a customer has made a transaction with an online retailer in a year.

```
User_Frequency <- UK_data %>%
  group_by(CustomerID) %>%
  summarise(Frequency = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
summary(User_Frequency$Frequency)
```

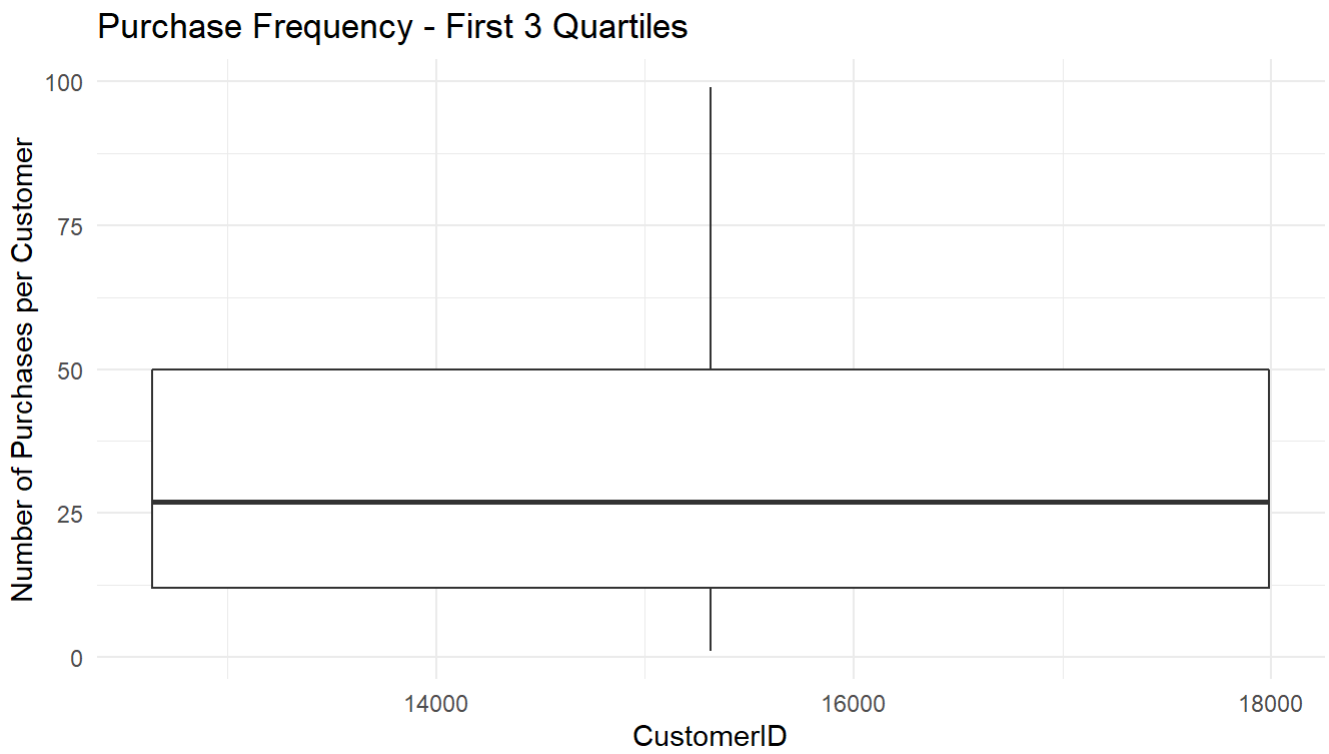
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   17.00   41.00   91.61  101.00 7983.00
```

average of 90 transactions a year, 75% of users have less than 100 purchases a year. The difference between the 3rd and max is really huge.

since the difference between 3rd quartile and max is very high, we plot the first 3 quartile and the max separately

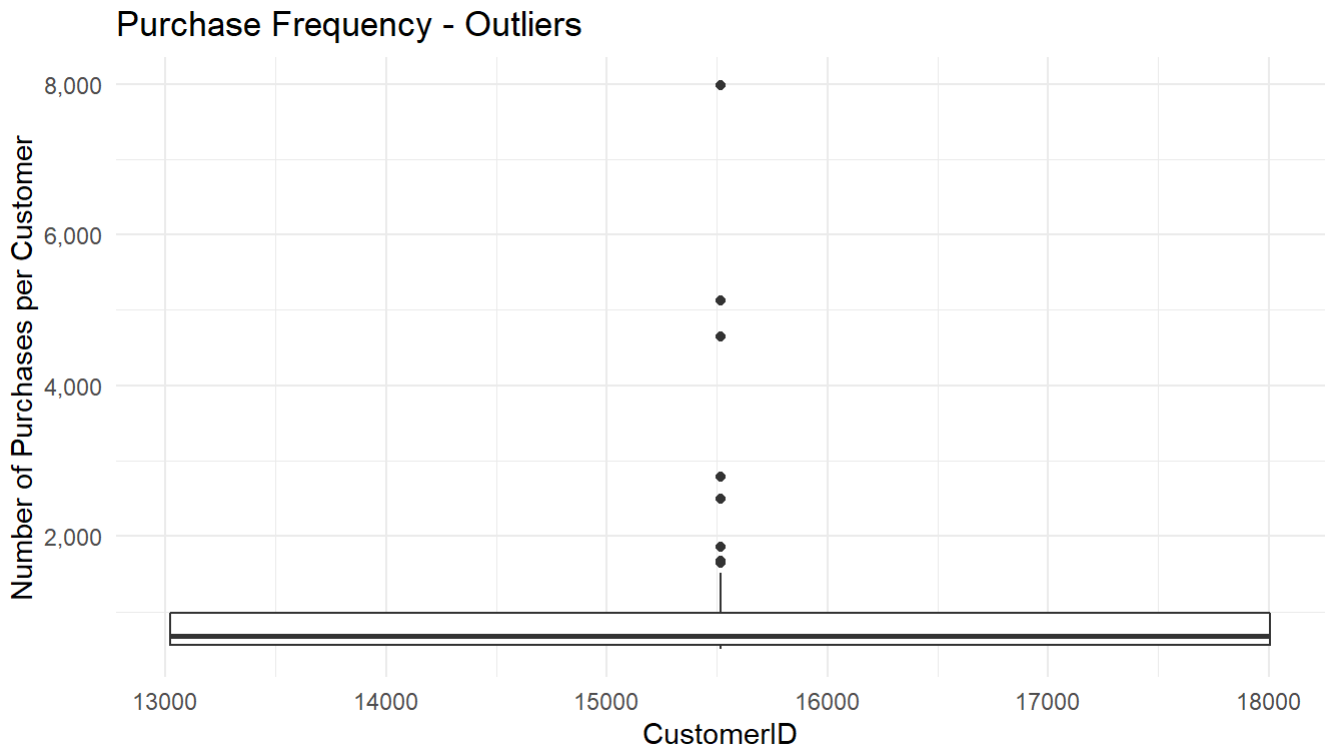
```
Below_3Q <- User_Frequency %>%
  filter(Frequency <= 99)

Outliers <- User_Frequency %>%
  filter(Frequency >= 500)

# Plotting first 3 Quartile
 ggplot(Below_3Q, aes(CustomerID, Frequency)) +
  geom_boxplot() +
  ylab('Number of Purchases per Customer') +
  ggtitle('Purchase Frequency - First 3 Quartiles') +
  theme(axis.ticks.x = element_blank()) +
  theme_minimal()
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```




```
#plotting the outliers
ggplot(Outliers, aes(CustomerID, Frequency)) +
 geom_boxplot() +
 ylab('Number of Purchases per Customer') +
 scale_y_continuous(labels= scales::comma) +
 ggtitle('Purchase Frequency - Outliers') +
 theme(axis.ticks.x = element_blank()) +
 theme_minimal()
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



we can see the outliers in the second plot.

monetary value refers to the total sum of revenue generated by the user over the course of a year. estimated using sale=unit price*qty and by grouping customer id.

```
Users_Monetary_Value <- UK_data %>%
  mutate(sale) %>%
  group_by(CustomerID) %>%
  summarise(Monetary_Value=sum(sale))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# Summary Statistics
summary(Users_Monetary_Value$Monetary_Value)
```

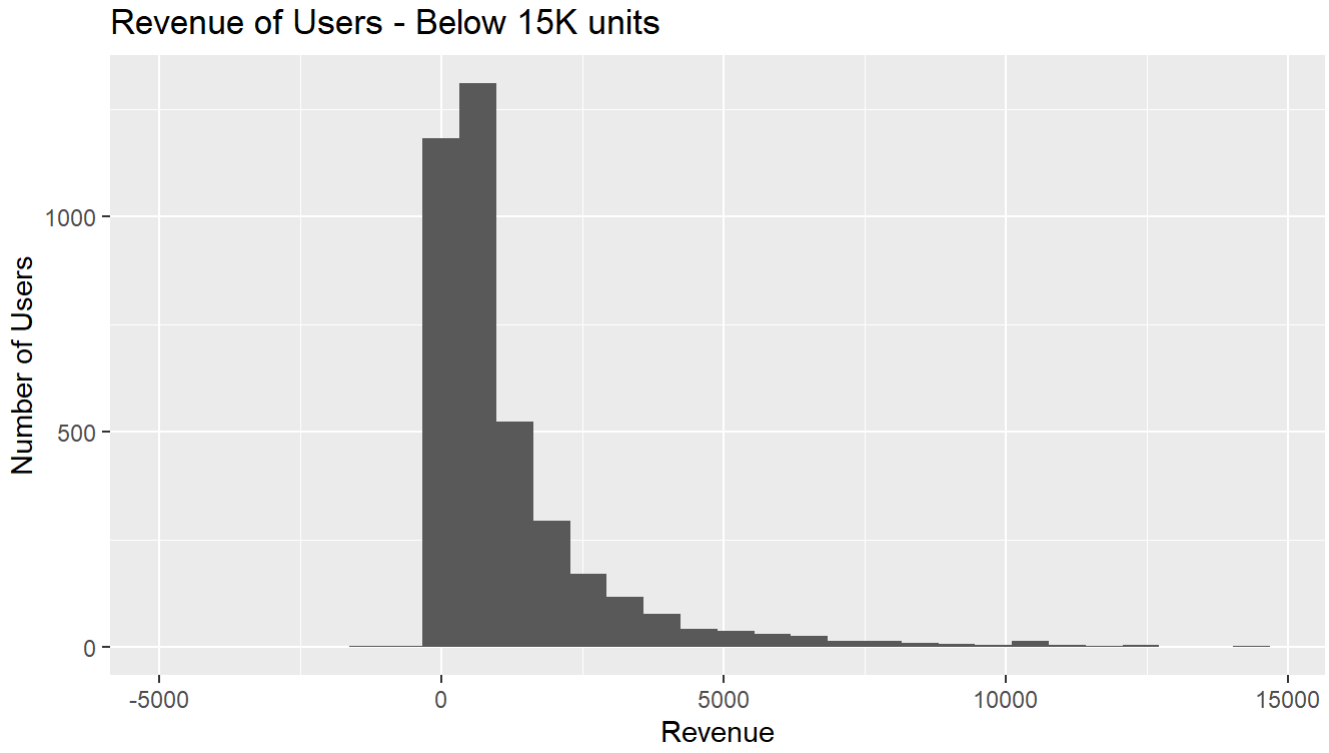
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  -4287.6    282.3    627.1   1713.4   1521.8 256438.5
```

we are splitting our plot into below third quartile and greater than 15k

#plot

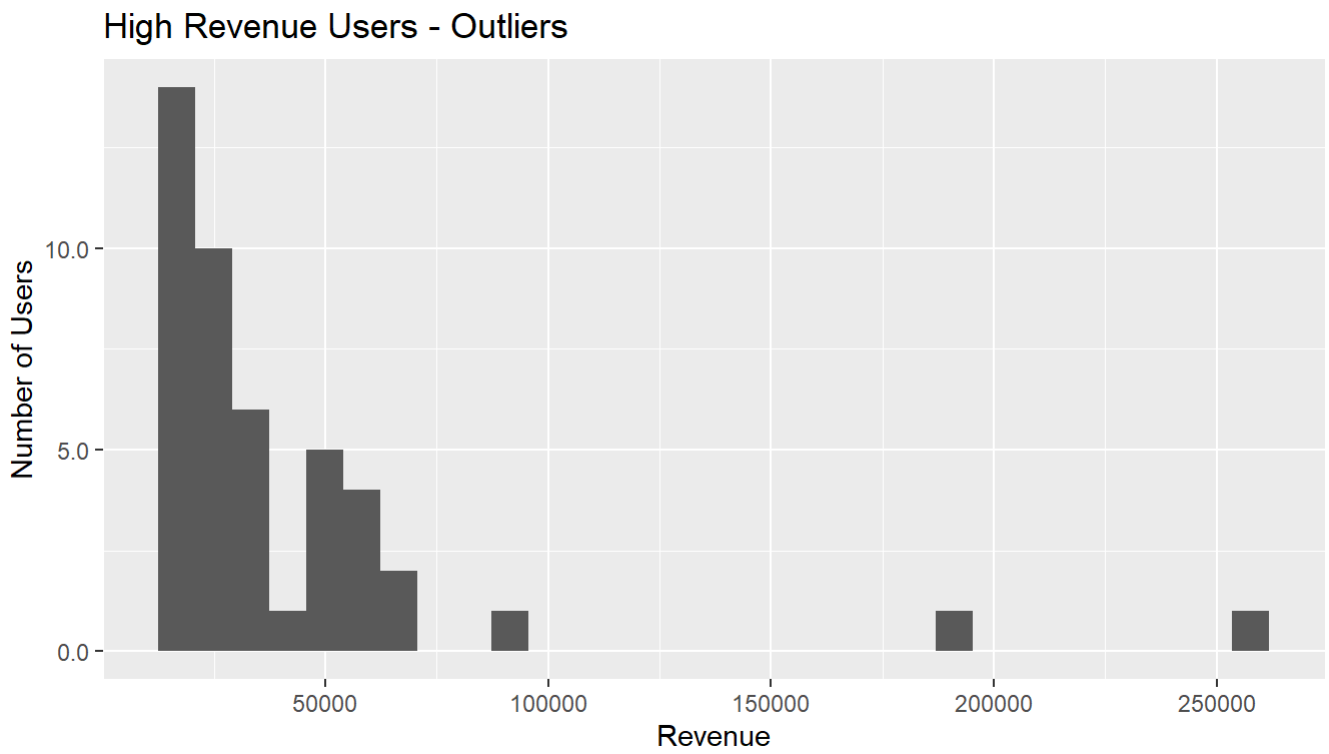
```
MV_3Q <- Users_Monetary_Value %>%
  filter(Monetary_Value <= 15000)

MV_Outliers <- Users_Monetary_Value %>%
  filter(Monetary_Value > 15000)

# Visualizing a histogram of revenue generated by user
 ggplot(MV_3Q, aes(Monetary_Value)) +
  geom_histogram() +
  ggtitle('Revenue of Users - Below 15K units') +
  ylab('Number of Users') +
  xlab('Revenue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
 ggplot(MV_Outliers, aes(Monetary_Value)) +
  geom_histogram() +
  ggtitle('High Revenue Users - Outliers') +
  ylab('Number of Users') +
  xlab('Revenue') +
  scale_x_continuous( breaks = c(50000, 100000, 150000, 200000, 250000, 300000, 350000)) +
  scale_y_continuous(labels = scales::comma)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



negative points to the purchase returns.

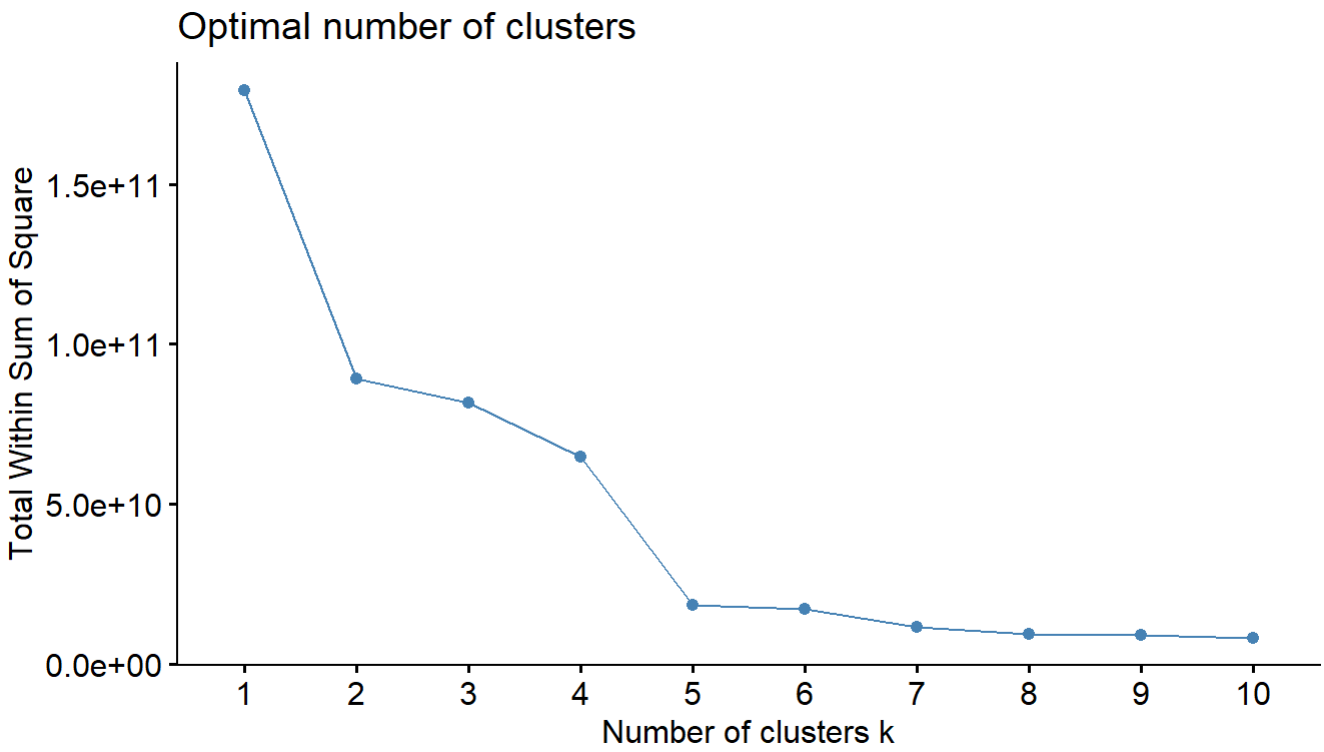
merging rfm to perform cluster segmentation

```
Users_RFM <- merge(Users_Recency, User_Frequency) # Merging Recency and Frequency
Users_RFM <- merge(Users_RFM, Users_Monetary_Value) # Merging Monetary Value
head(Users_RFM,10)
```

```
##    CustomerID Recency Frequency Monetary_Value
## 1       12346     325         2           0.00
## 2       12747       2       103        4196.01
## 3       12748       0      4642       29072.10
## 4       12749       3       231        3868.20
## 5       12820       3        59         942.34
## 6       12821     214         6          92.72
## 7       12822      70        47         918.98
## 8       12823      74         5        1759.50
## 9       12824      59        25         397.12
## 10      12826       2        94        1468.12
```

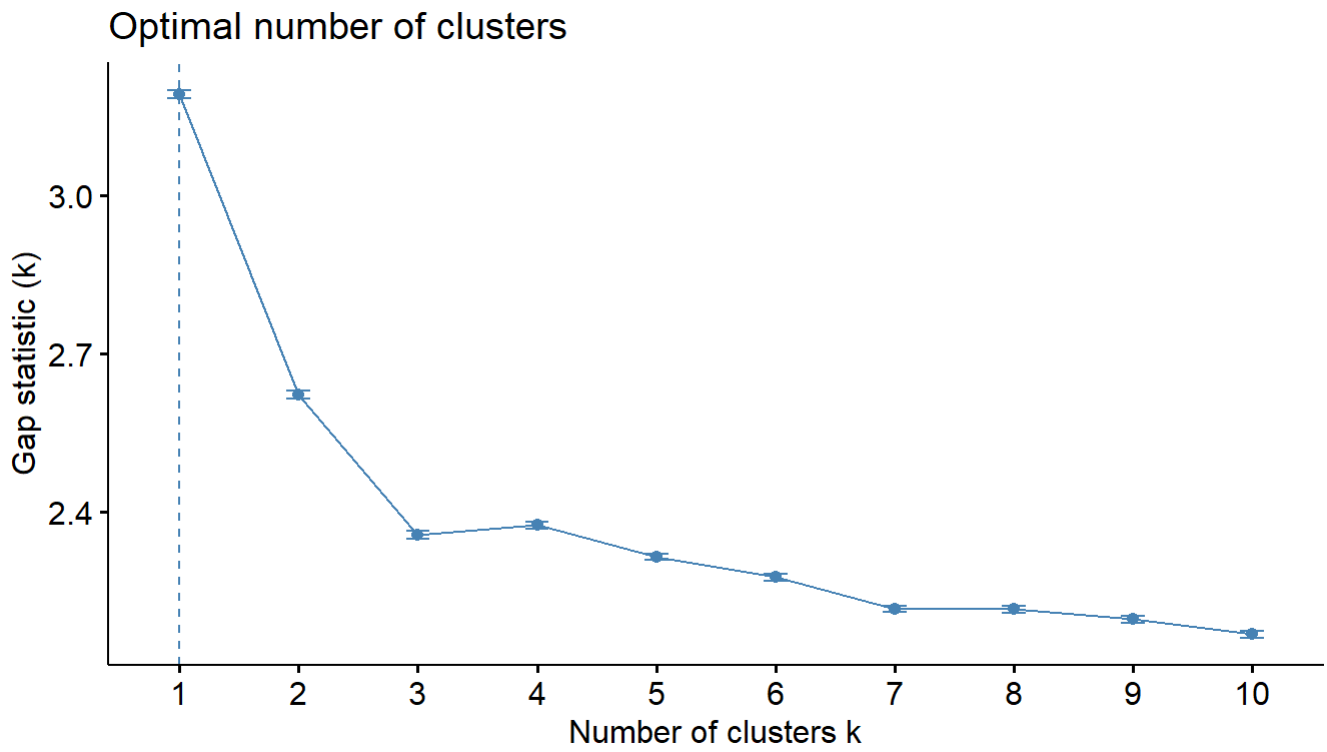
#applying k means clustering

```
fviz_nbclust(Users_RFM, kmeans, method = "wss")
```

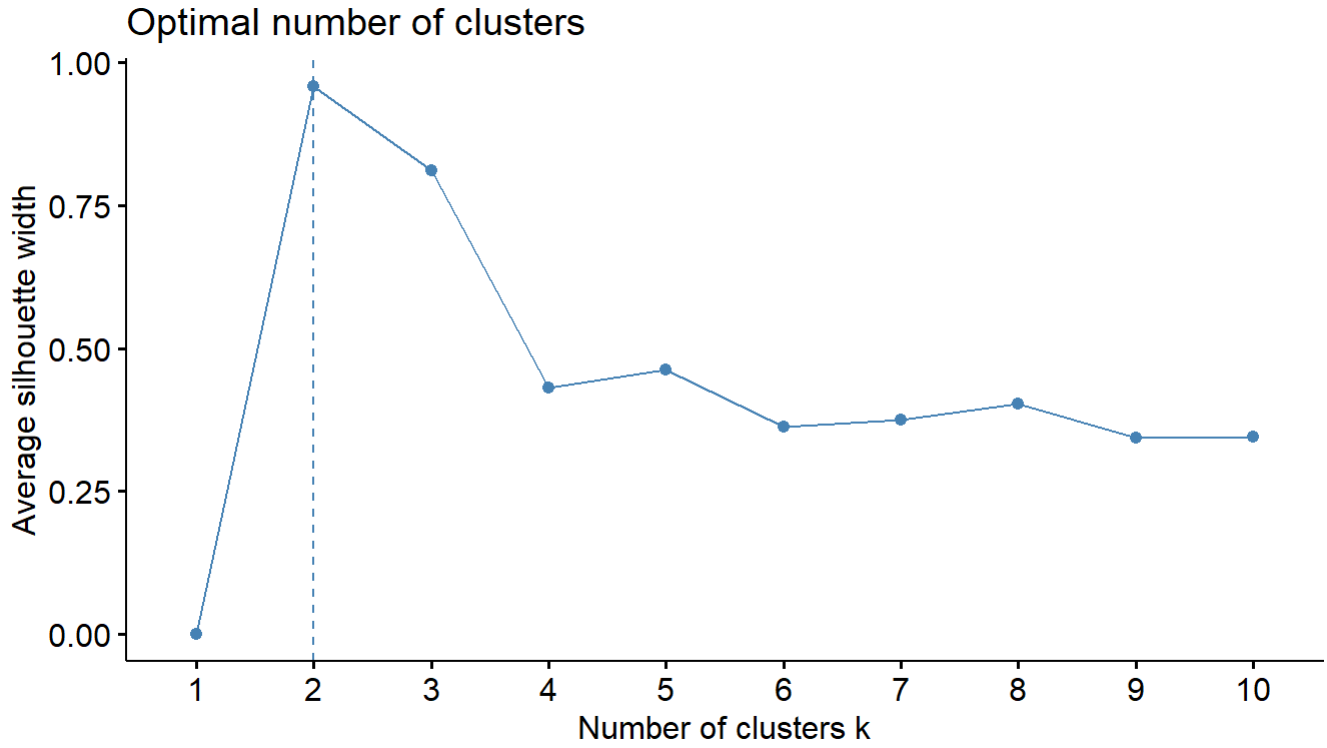



```
fviz_nbclust(Users_RFM,kmeans,method = "gap_stat")
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 197500)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 197500)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 197500)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 197500)
```




```
fviz_nbclust(Users_RFM,kmeans,method = "silhouette")
```



all of them suggest difference k values, we'll go with k=3 where we can split our customers into good medium and bad or top tier, mid tier and low tier.

we'll go with k=3

```
set.seed(123)
clusters <- kmeans(scale(Users_RFM[,2:4]), 3, nstart = 1) # Performing kmeans with RFM variables
  and creating 3 clusters.

Users_RFM$Cluster <- as.factor(clusters$cluster)

KMeans_Results <- Users_RFM %>%
  group_by(Cluster) %>%
  summarise('Number of Users' = n(),
            'Recency Mean' = round(mean(Recency)),
            'Frequency Mean' = scales::comma(round(mean(Frequency))),
            'Monetary Value Mean' = round(mean(Monetary_Value)),
            'Cluster Revenue' =sum(Monetary_Value))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

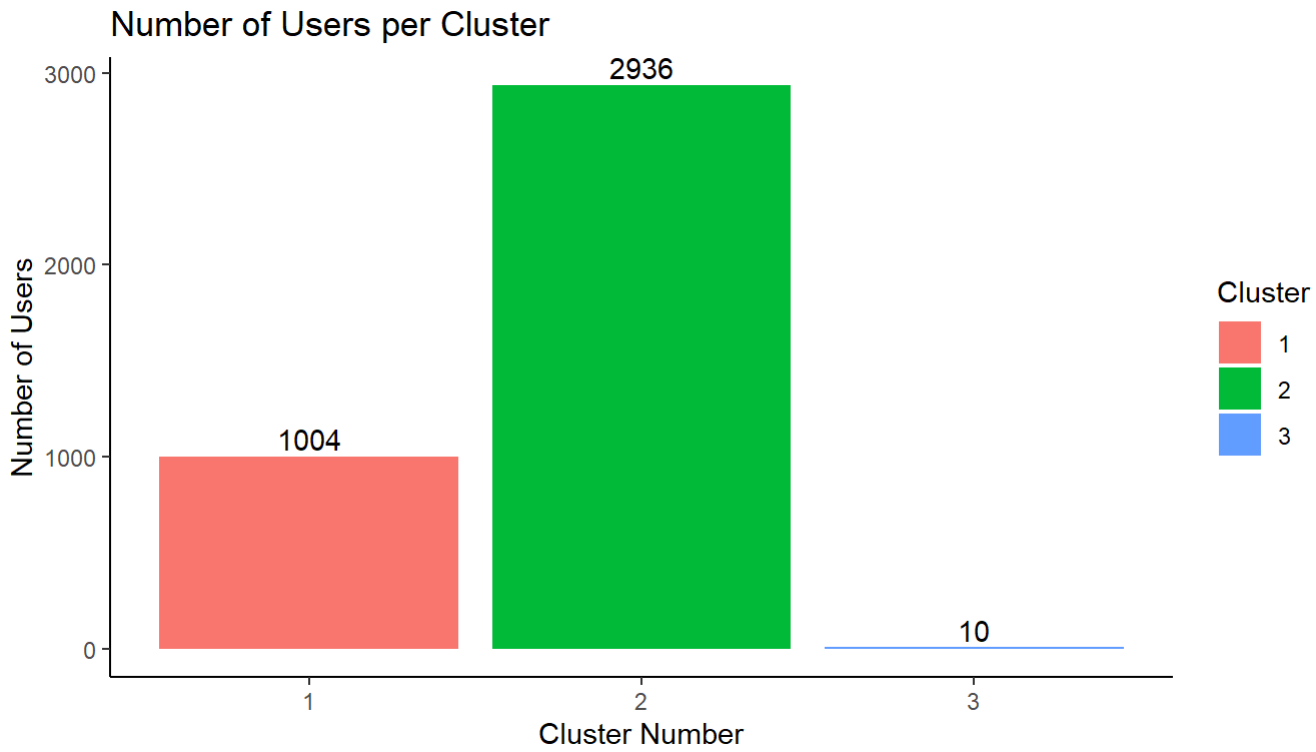
```
DT::datatable((KMeans_Results),
              rownames = FALSE)
```

Show 10 entries Search:

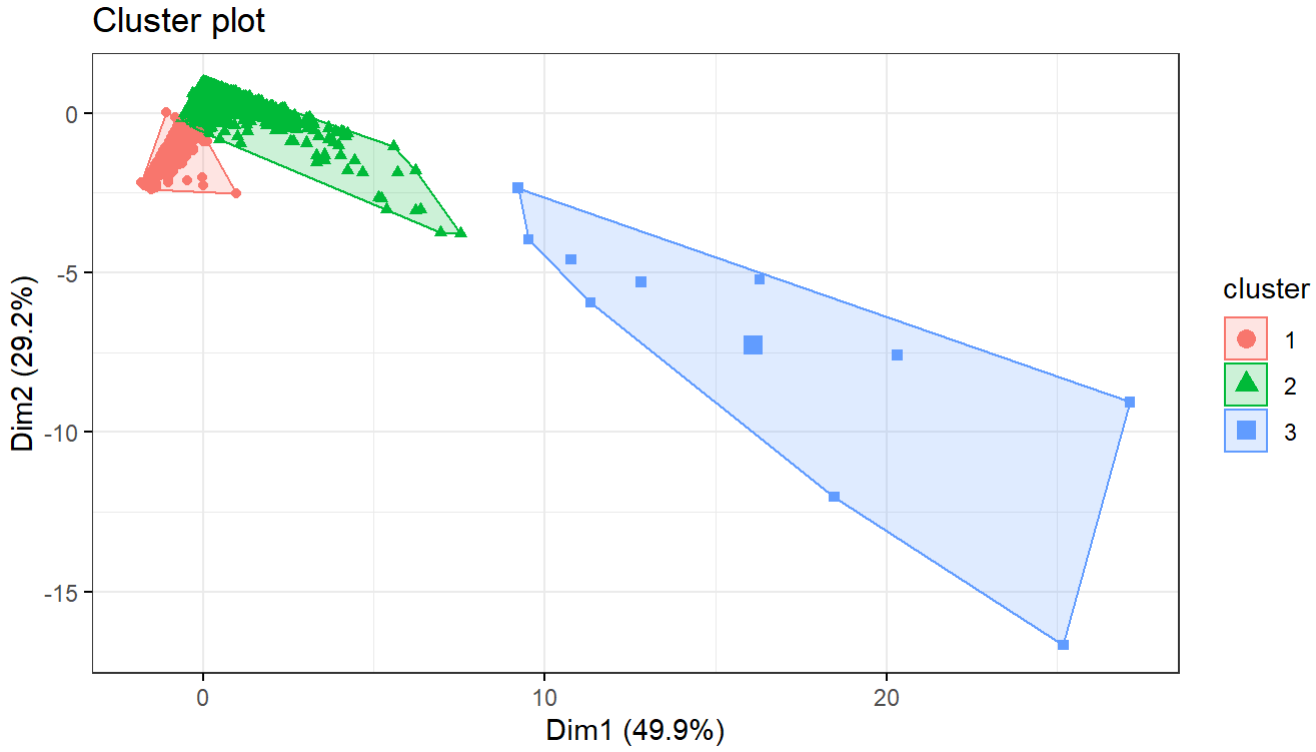
Cluster	Number of Users	Recency Mean	Frequency Mean	Monetary Value Mean	Cluster Revenue
1	1004	244	28	431	432667.341
2	2936	39	104	1872	5497244.713
3	10	2	2,838	83796	837961.34

Showing 1 to 3 of 3 entries Previous 1 Next

```
ggplot(KMeans_Results, aes(Cluster, `Number of Users`)) +
 geom_text(aes(label = `Number of Users`), vjust = -0.3) +
        geom_bar(aes(fill=Cluster), stat='identity') +
 ggtitle('Number of Users per Cluster') +
 xlab("Cluster Number") +
 theme_classic()
```




```
fviz_cluster(clusters, data = Users_RFM[,2:4],
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```



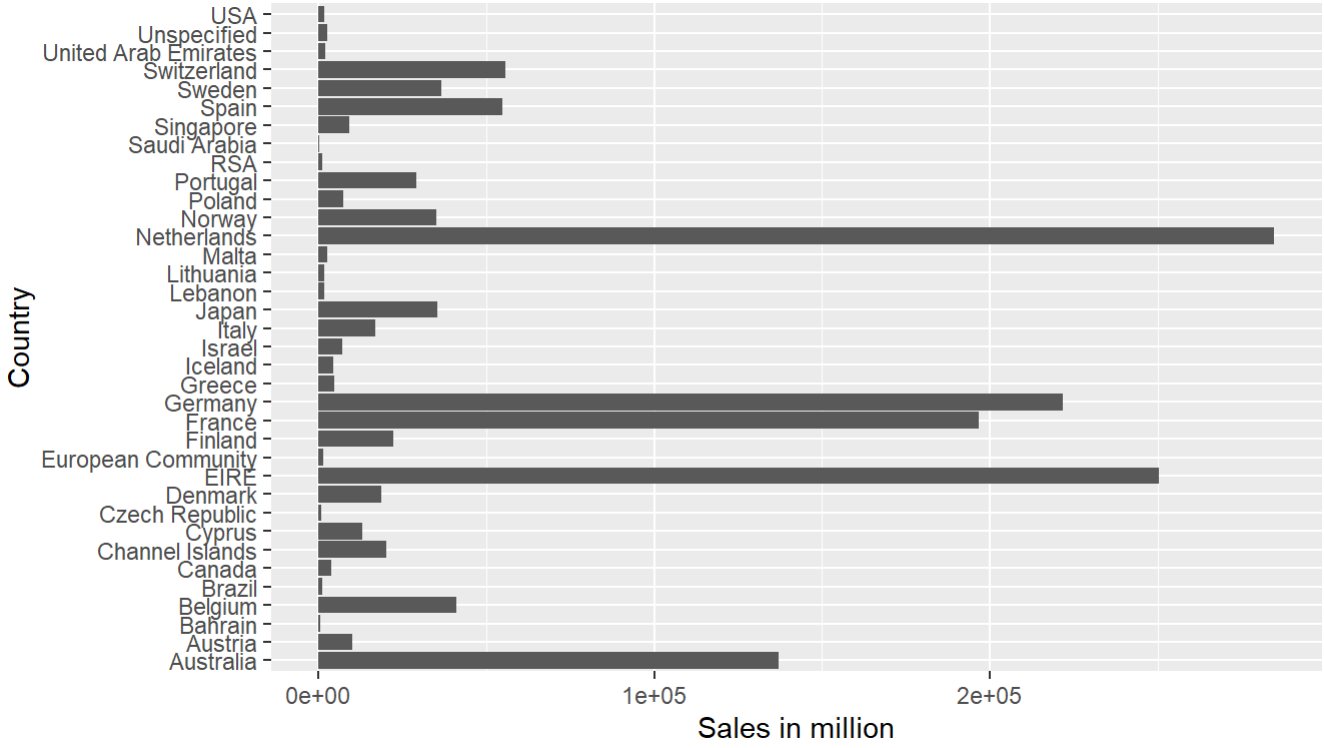
3 clusters have been constructed based on recency, frequence and monetary value.These clusters are based on the customers' behaviour with the store. High value cust, medium val cust and low val customer. the cluster with 1004 cust is low value with avg 244 days of in activity, 27 avg purchases per

year and a monetary value of 431. Medium value consists is the one with 2936 customers and the high value is with only 10 customers.

#Dropping UK and plotting the rest of the countries

```
data %>% filter(Country != "United Kingdom") %>% group_by(Country) %>%
summarise(revenue = sum(sale)) %>%
ggplot(aes(y=revenue, x=Country)) + geom_bar(stat="identity") + coord_flip() +
labs(x="Country", y="Sales in million")
```

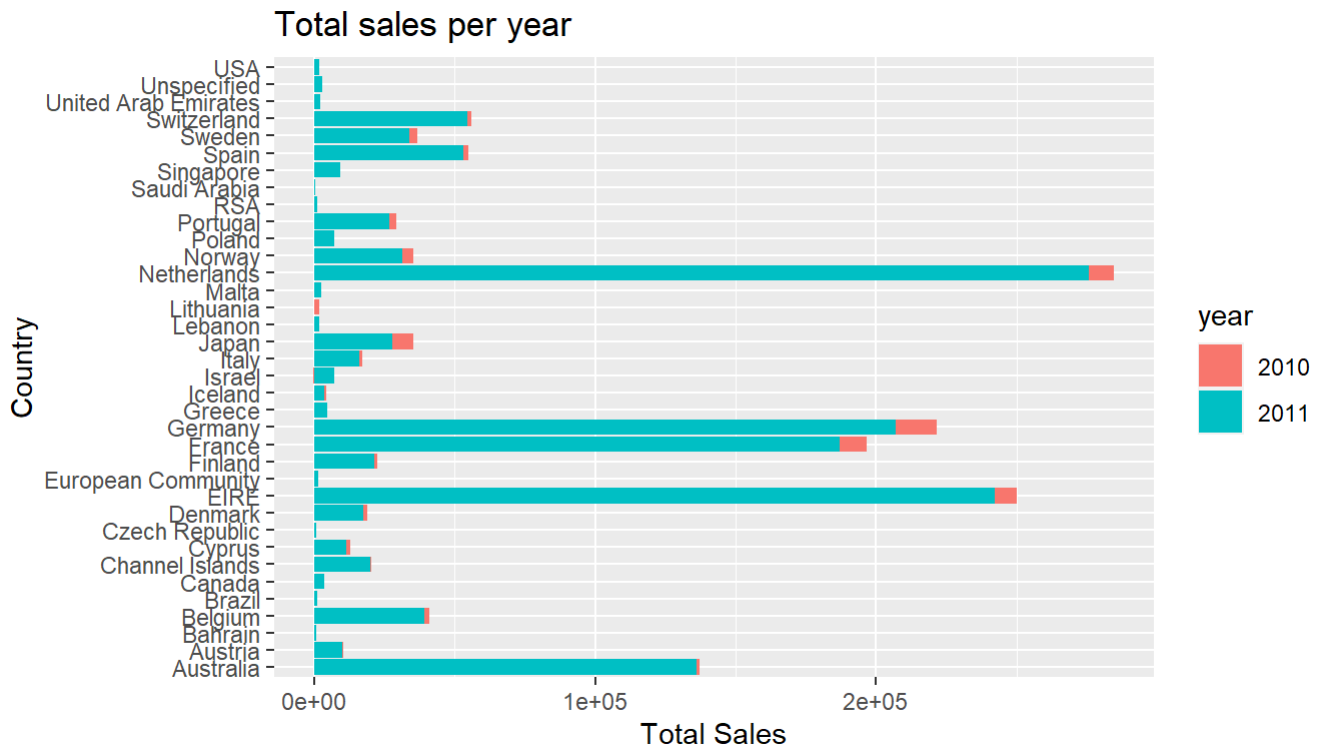
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



#fill=year

```
data %>% filter(Country != "United Kingdom") %>%
group_by(year, Country) %>% summarise(revenue = sum(sale)) %>%
ggplot(aes(x=Country, y=revenue, fill=year)) + geom_bar(stat="identity") + coord_flip() +
labs(x= "Country", y="Total Sales", title = "Total sales per year")
```

```
## `summarise()` regrouping output by 'year' (override with `.groups` argument)
```



#Top five countries

```
topFiveCountries <- data %>%
  filter(Country == 'Netherlands' | Country == 'EIRE' | Country == 'Germany' | Country == 'Franc
  e' | Country == 'Australia')

topFiveCountrySummary <- topFiveCountries %>%
  group_by(Country, date) %>%
  summarise(revenue = sum(sale), transactions = n_distinct(InvoiceNo), customers = n_distinct(Cu
  stomerID)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))
```

```
## `summarise()` regrouping output by 'Country' (override with `.groups` argument)
```

```
head(topFiveCountrySummary)
```

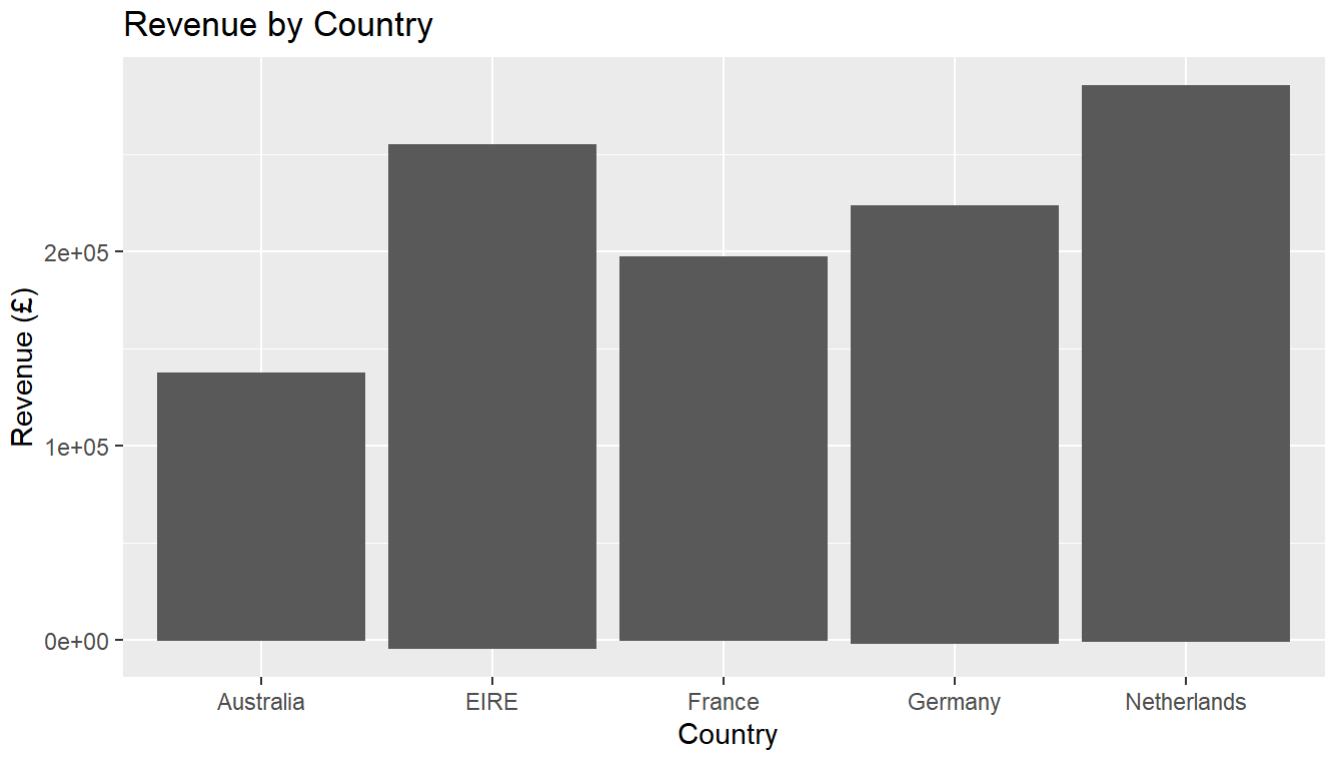
```
## # A tibble: 6 x 6
##   Country     date       revenue transactions customers aveOrdVal
##   <fct>       <date>       <dbl>        <int>     <int>     <dbl>
## 1 Netherlands 2011-10-20  25834.            3         1     8611.
## 2 Australia   2011-06-15  23427.            2         1    11713.
## 3 Australia   2011-08-18  21880.            1         1    21880.
## 4 Netherlands 2011-08-11  19151.            1         1    19151.
## 5 Netherlands 2011-02-21  18279.            2         1     9140.
## 6 Netherlands 2011-03-29  18248.            2         1     9124.
```

```
head(countryCustSummary, n = 10) #just for reference
```

```
## # A tibble: 10 x 4
##    Country         revenue customers aveCustVal
##    <fct>             <dbl>     <int>      <dbl>
##  1 United Kingdom 6767873.      3950      1713.
##  2 Netherlands     284662.         9     31629.
##  3 EIRE            250285.         3     83428.
##  4 Germany         221698.        95      2334.
##  5 France          196713.        87      2261.
##  6 Australia       137077.         9     15231.
##  7 Switzerland      55739.        21      2654.
##  8 Spain            54775.        31      1767.
##  9 Belgium          40911.        25      1636.
## 10 Sweden           36596.         8      4574.
```

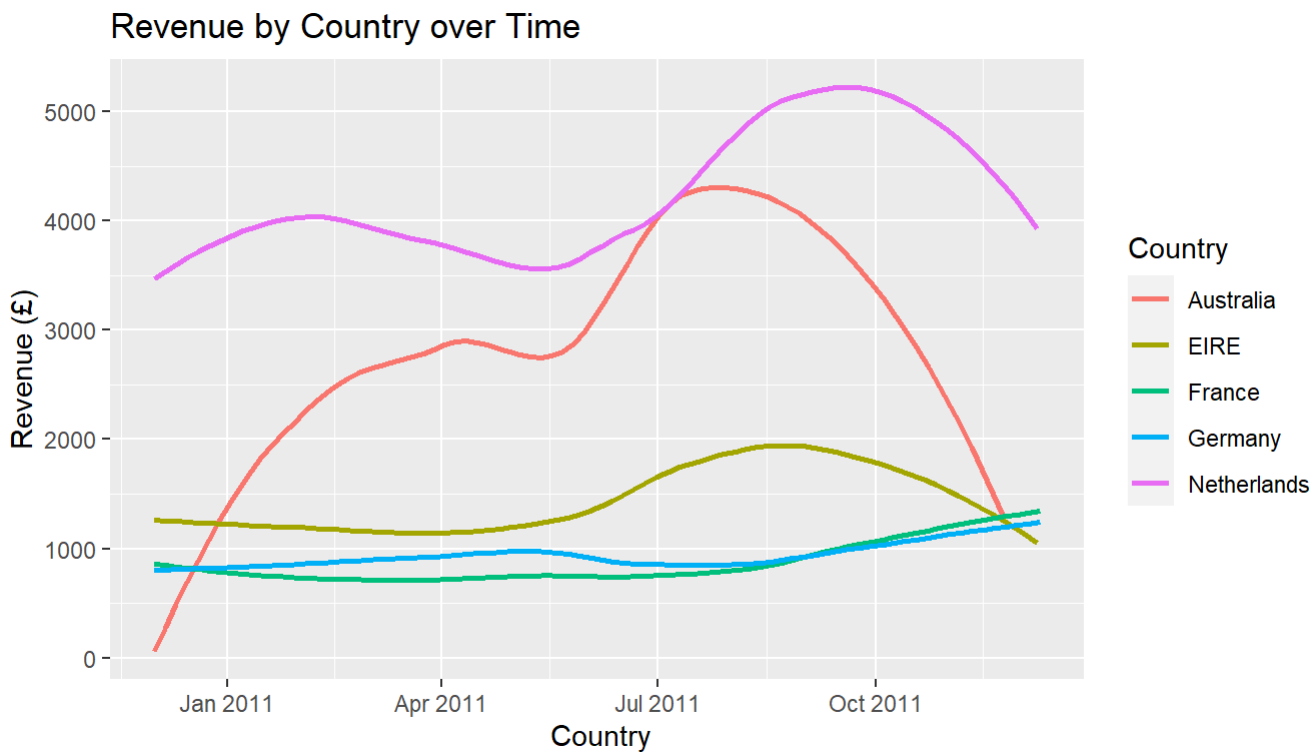
#Visualising top5 countries

```
ggplot(topFiveCountrySummary, aes(x = Country, y = revenue)) + geom_col() + labs(x = ' Country',
  y = 'Revenue (£)', title = 'Revenue by Country')
```




```
ggplot(topFiveCountrySummary, aes(x = date, y = revenue, colour = Country)) + geom_smooth(method
  = 'auto', se = FALSE) + labs(x = ' Country', y = 'Revenue (£)', title = 'Revenue by Country ov
  er Time')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

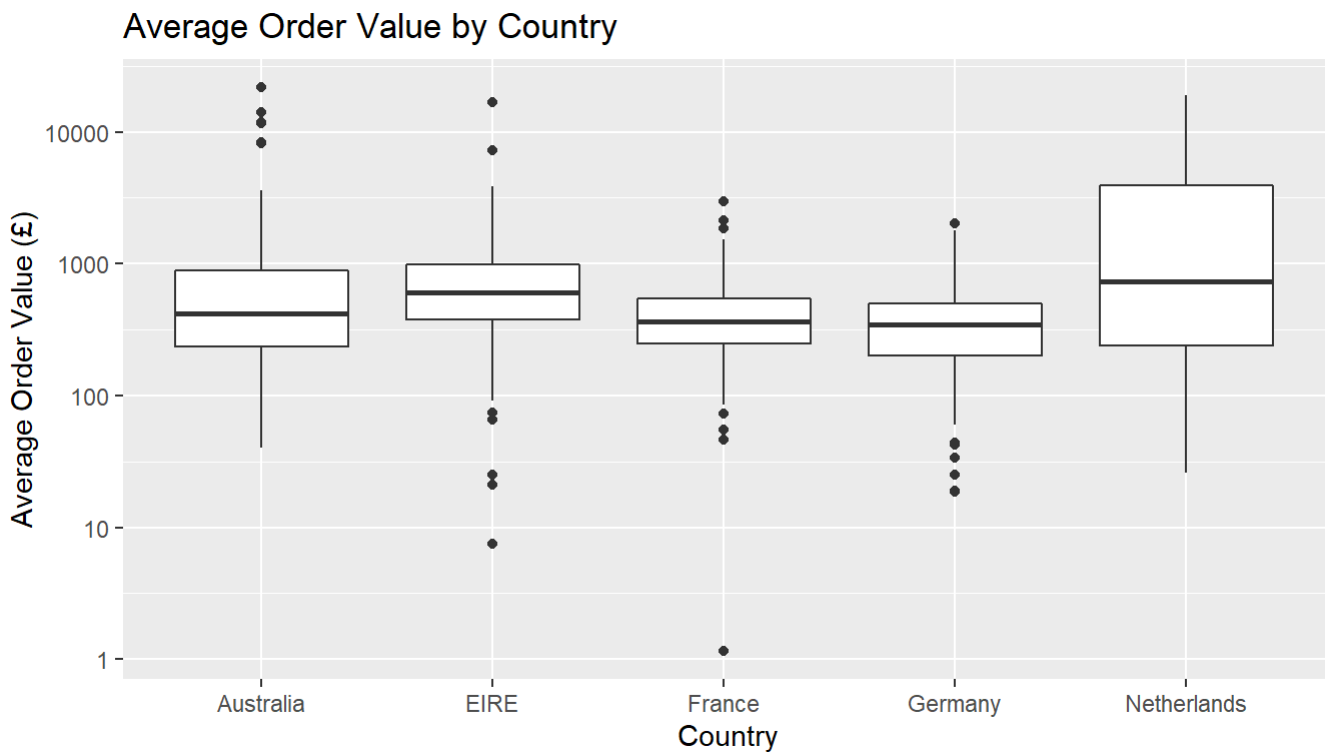



```
ggplot(topFiveCountrySummary, aes(x = Country, y = aveOrdVal)) + geom_boxplot() + labs(x = ' Cou
  ntry', y = 'Average Order Value (£)', title = 'Average Order Value by Country') + scale_y_log1
  0()
```

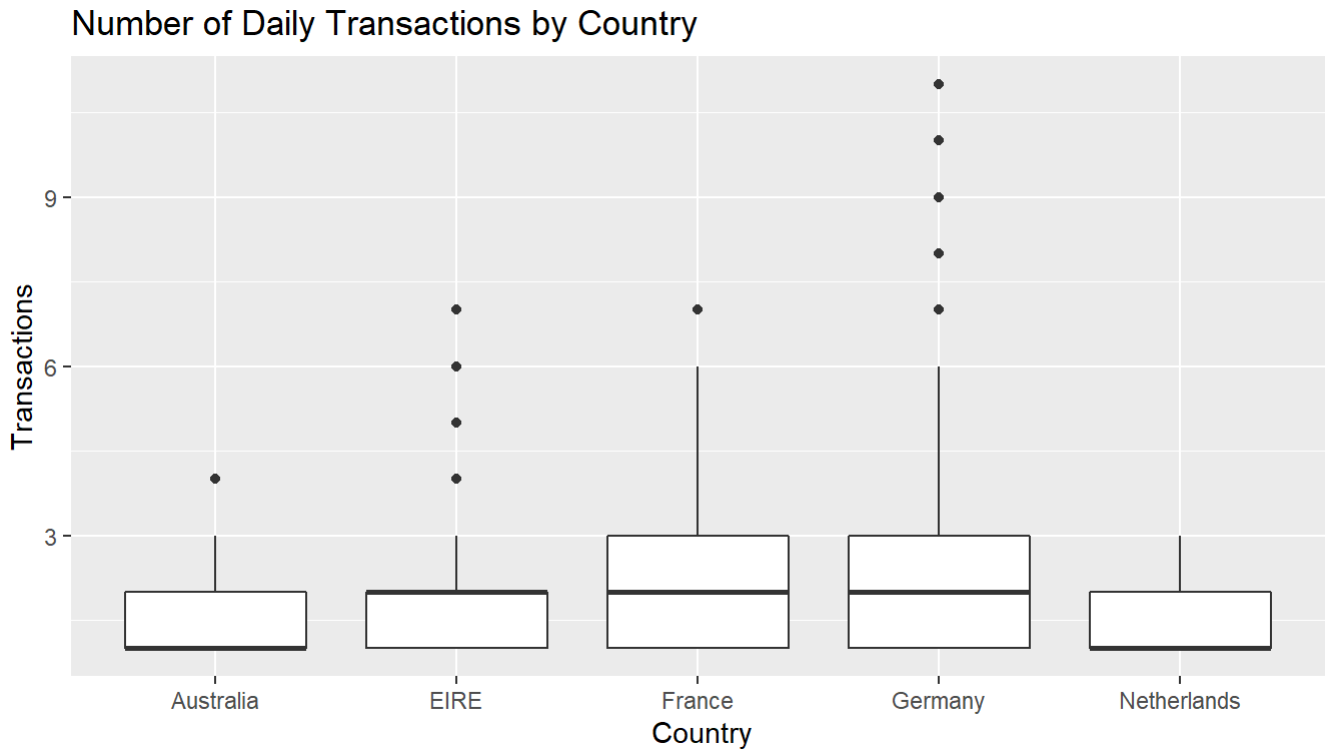
```
## Warning in self$trans$transform(x): NaNs produced
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 78 rows containing non-finite values (stat_boxplot).
```




```
ggplot(topFiveCountrySummary, aes(x = Country, y = transactions)) + geom_boxplot() + labs(x = '
   Country', y = 'Transactions', title = 'Number of Daily Transactions by Country')
```



Revenue is EIRE is solely driven by 3 customers, who seems to be buying regularly and have a good average order value, but revenue has been declining lately. An email or any other form of promotion can help us buy them back because the revenue is really good there and the case is same for netherlands. netherlands has also been a significant source of revenue, and that has been declining lately too. A good market is present in that region. France and Germany has an increasing trend which is a good thing. This can be seen in our boxplots.

###customer segmentation #starting with avgorder value

```
custSummary <- data %>%
  group_by(CustomerID) %>%
  summarise(revenue = sum(sale), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(desc(revenue))
```

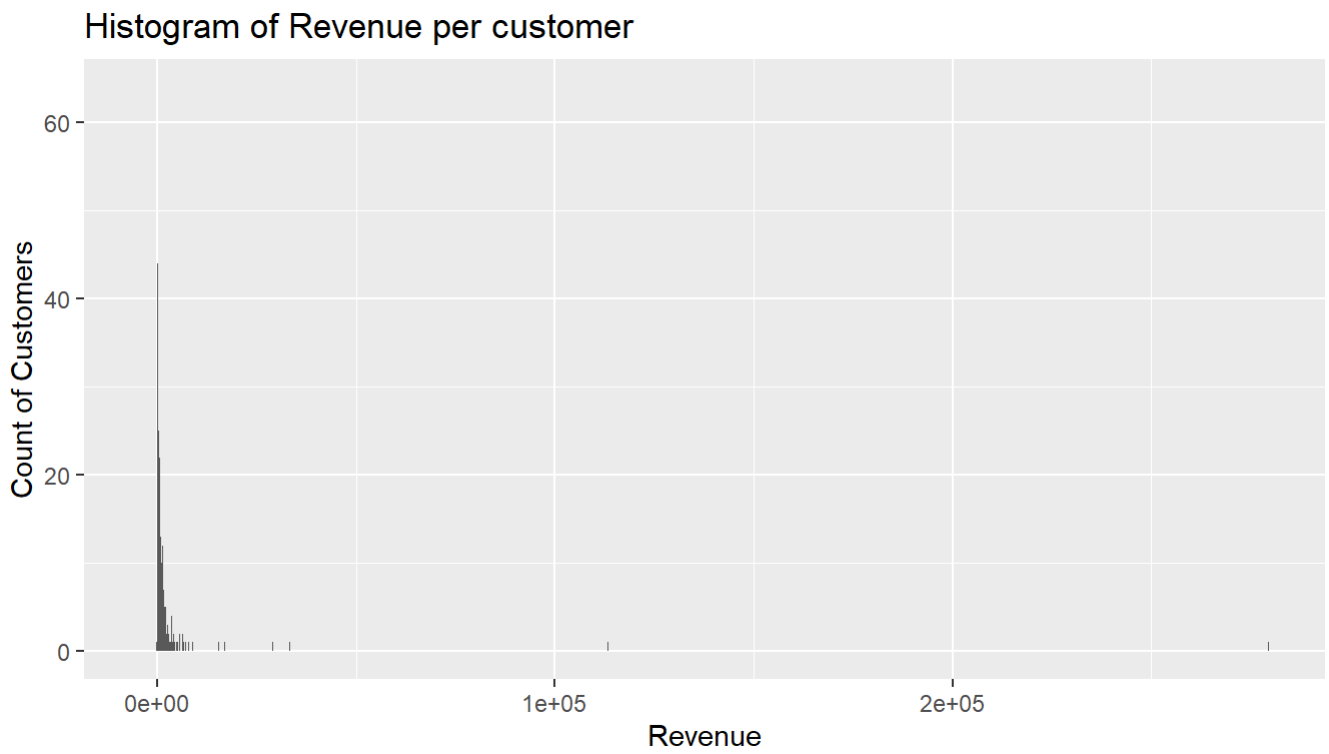
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(custSummary, n = 10)
```

```
## # A tibble: 10 x 4
##    CustomerID revenue transactions aveOrdVal
##         <int>   <dbl>        <int>     <dbl>
##  1      14646 279489.           77     3630.
##  2      18102 256438.           62     4136.
##  3      17450 187482.           55     3409.
##  4      14911 132573.          248      535.
##  5      12415 123725.           26     4759.
##  6      14156 113384.           66     1718.
##  7      17511  88125.           46     1916.
##  8      16684  65892.           31     2126.
##  9      13694  62653.           60     1044.
## 10      15311  59419.          118      504.
```

we can build a nice dataframe from this point with recency, frequency and revenue

```
#revenue vs customers
ggplot(custSummary, aes(revenue)) + geom_histogram(binwidth = 10) + labs(x = 'Revenue', y = 'Cou
  nt of Customers', title = 'Histogram of Revenue per customer')
```



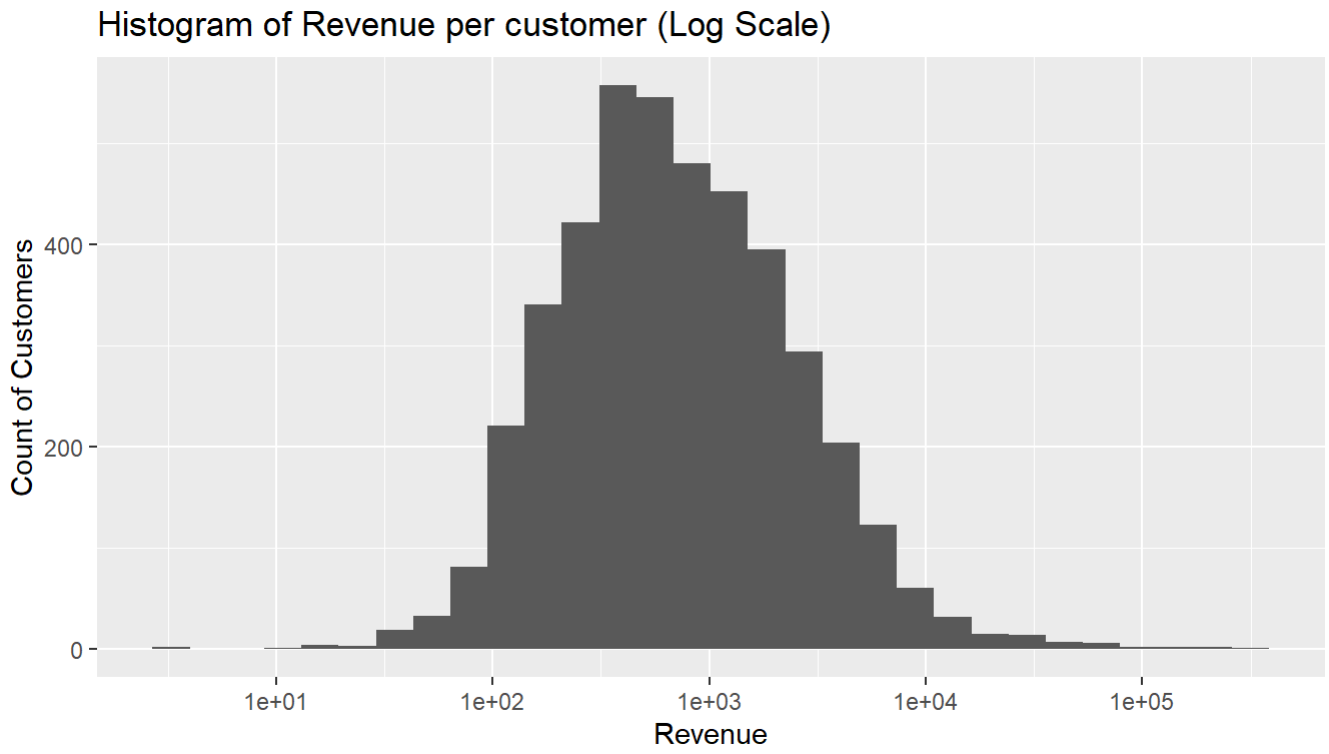

```
#same in log scale
ggplot(custSummary, aes(revenue)) + geom_histogram() + scale_x_log10() + labs(x = 'Revenue', y =
  'Count of Customers', title = 'Histogram of Revenue per customer (Log Scale)')
```

```
## Warning in self$trans$transform(x): NaNs produced
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

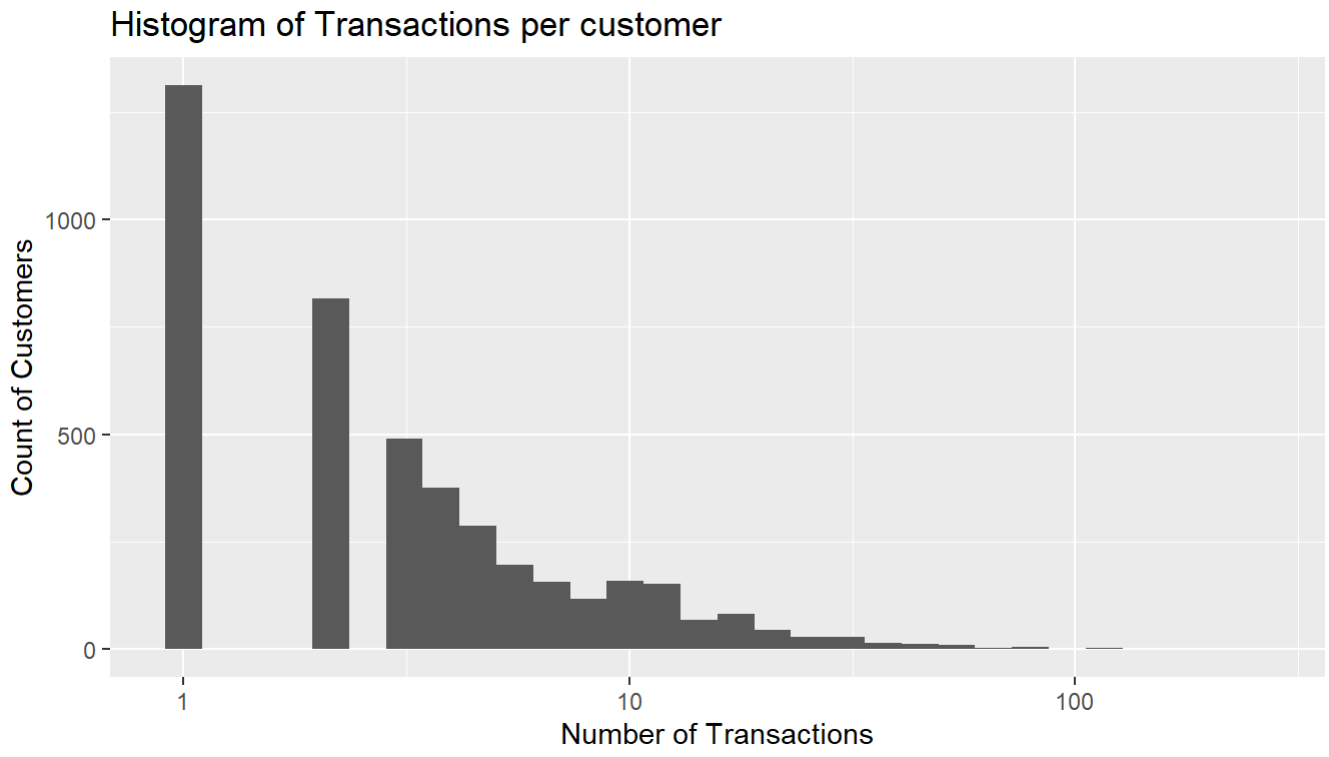
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 55 rows containing non-finite values (stat_bin).
```




```
#tranpercust
ggplot(custSummary, aes(transactions)) + geom_histogram() + scale_x_log10() + labs(x = 'Number o
  f Transactions', y = 'Count of Customers', title = 'Histogram of Transactions per customer')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
custSummaryB <- data %>%
  group_by(CustomerID, InvoiceNo) %>%
  summarise(revenue = sum(sale), transactions = n_distinct(InvoiceNo)) %>%
  mutate(aveOrdVal = (round((revenue / transactions),2))) %>%
  ungroup() %>%
  arrange(revenue) %>%
  mutate(cumsum=cumsum(revenue)) #cumulativesum
```

```
## `summarise()` regrouping output by 'CustomerID' (override with `.groups` argument)
```

```
print.data.frame(head(custSummaryB, n =10))
```

```
##    CustomerID InvoiceNo    revenue transactions  aveOrdVal    cumsum
## 1       16446   C581484 -168469.60            1 -168469.60 -168469.6
## 2       12346   C541433  -77183.60            1  -77183.60 -245653.2
## 3       15098   C556445  -38970.00            1  -38970.00 -284623.2
## 4       15749   C550456  -22998.40            1  -22998.40 -307621.6
## 5       16029   C570556  -11816.64            1  -11816.64 -319438.2
## 6       12536   C573079   -8322.12            1   -8322.12 -327760.4
## 7       16029   C551685   -8142.75            1   -8142.75 -335903.1
## 8       16029   C551699   -6930.00            1   -6930.00 -342833.1
## 9       12744   C571750   -6068.06            1   -6068.06 -348901.2
## 10      14911   C562375   -4345.10            1   -4345.10 -353246.3
```

##going through this table we can see a lot of refunds and returns. let's take a particular ID as an example

```
data %>% filter(CustomerID == 16446)
```

```
##   InvoiceNo StockCode                Description Quantity    InvoiceDate
## 1    553573     22980      PANTRY SCRUBBING BRUSH        1 5/18/2011 9:52
## 2    553573     22982         PANTRY PASTRY BRUSH        1 5/18/2011 9:52
## 3    581483     23843 PAPER CRAFT , LITTLE BIRDIE    80995 12/9/2011 9:15
## 4   C581484     23843 PAPER CRAFT , LITTLE BIRDIE   -80995 12/9/2011 9:27
##   UnitPrice CustomerID        Country       date time month year hourOfDay
## 1      1.65      16446 United Kingdom 2011-05-18 9:52     5 2011         9
## 2      1.25      16446 United Kingdom 2011-05-18 9:52     5 2011         9
## 3      2.08      16446 United Kingdom 2011-12-09 9:15    12 2011         9
## 4      2.08      16446 United Kingdom 2011-12-09 9:27    12 2011         9
##   dayOfWeek       sale
## 1       Wed       1.65
## 2       Wed       1.25
## 3       Fri  168469.60
## 4       Fri -168469.60
```

they ordered a scrubbing brush and p brush first and the second order was a mass order woth over 80k paper craft, little birdies and checked out with 168,470 pounds, this was an irresponsible transaction which was refunded 12 min later.

calculating recent days by subtracting max date from the date after the max range i.e 10/12/2011.

```
range(data$date)
```

```
## [1] "2010-12-01" "2011-12-09"
```

```
custSummaryB <- data %>%
  group_by(InvoiceNo, CustomerID, Country, date, month, year, hourOfDay, dayOfWeek) %>%
  summarise(orderVal = sum(sale)) %>%
  mutate(recent=as.numeric(as.Date("2011-12-10")-max(date))) %>%
  ungroup()
```

```
## `summarise()` regrouping output by 'InvoiceNo', 'CustomerID', 'Country', 'date', 'month', 'ye
ar', 'hourOfDay' (override with `.groups` argument)
```

```
custSummaryB$recent <- as.character(custSummaryB$recent)
custSummaryB$recentDays <- sapply(custSummaryB$recent, FUN = function(x) {strsplit(x, split = '[
  ]')[[1]][1]})

custSummaryB$recentDays <- as.integer(custSummaryB$recentDays) #recentdays

print.data.frame(head(custSummaryB, n = 10))
```

```
##    InvoiceNo CustomerID        Country       date month year hourOfDay
## 1     536365      17850 United Kingdom 2010-12-01    12 2010         8
## 2     536366      17850 United Kingdom 2010-12-01    12 2010         8
## 3     536367      13047 United Kingdom 2010-12-01    12 2010         8
## 4     536368      13047 United Kingdom 2010-12-01    12 2010         8
## 5     536369      13047 United Kingdom 2010-12-01    12 2010         8
## 6     536370      12583         France 2010-12-01    12 2010         8
## 7     536371      13748 United Kingdom 2010-12-01    12 2010         9
## 8     536372      17850 United Kingdom 2010-12-01    12 2010         9
## 9     536373      17850 United Kingdom 2010-12-01    12 2010         9
## 10    536374      15100 United Kingdom 2010-12-01    12 2010         9
##    dayOfWeek orderVal recent recentDays
## 1        Wed   139.12    374        374
## 2        Wed    22.20    374        374
## 3        Wed   278.73    374        374
## 4        Wed    70.05    374        374
## 5        Wed    17.85    374        374
## 6        Wed   855.86    374        374
## 7        Wed   204.00    374        374
## 8        Wed    22.20    374        374
## 9        Wed   259.86    374        374
## 10       Wed   350.40    374        374
```

adding all the required columns to the frame

```
customerBreakdown <- custSummaryB %>%
  group_by(CustomerID, Country) %>%
  summarise(orders = n_distinct(InvoiceNo), revenue = sum(orderVal), meanRevenue = round(mean(or
  derVal), 2), medianRevenue = median(orderVal),
            mostDay = names(which.max(table(dayOfWeek))), mostHour = names(which.max(table(hourO
  fDay))),
            recency = min(recentDays))%>%
  ungroup()
```

```
## `summarise()` regrouping output by 'CustomerID' (override with `.groups` argument)
```

```
print.data.frame(head(customerBreakdown))
```

```
##   CustomerID        Country orders revenue meanRevenue medianRevenue mostDay
## 1      12346 United Kingdom      2    0.00        0.00          0.00     Tue
## 2      12347        Iceland      7 4310.00      615.71        584.91     Tue
## 3      12348        Finland      4 1797.24      449.31        338.50     Tue
## 4      12349          Italy      1 1757.55     1757.55       1757.55     Mon
## 5      12350         Norway      1  334.40      334.40        334.40     Wed
## 6      12352         Norway     11 1545.41      140.49        160.33     Tue
##   mostHour recency
## 1       10     326
## 2       14       3
## 3       10      76
## 4        9      19
## 5       16     311
## 6       14      37
```

```
custBreakSum <- customerBreakdown %>%
  filter(orders > 1, revenue > 50)

print.data.frame(head(custBreakSum))
```

```
##   CustomerID  Country orders revenue meanRevenue medianRevenue mostDay mostHour
## 1      12347  Iceland      7 4310.00      615.71        584.91     Tue       14
## 2      12348  Finland      4 1797.24      449.31        338.50     Tue       10
## 3      12352   Norway     11 1545.41      140.49        160.33     Tue       14
## 4      12356 Portugal      3 2811.43      937.14        481.46     Tue       12
## 5      12358  Austria      2 1168.06      584.03        584.03     Tue       10
## 6      12359   Cyprus      6 6245.53     1040.92        828.41     Wed       12
##   recency
## 1       3
## 2      76
## 3      37
## 4      23
## 5       2
## 6       8
```

```
dim(custBreakSum)
```

```
## [1] 3032    9
```

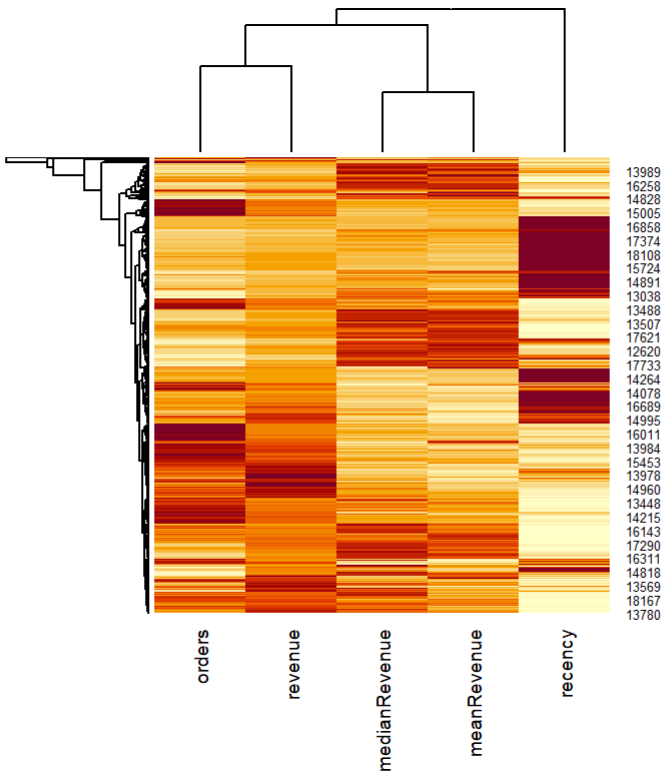
it shows how many repeat cutomers are present with their countries, order details, total revenue and avg order for day, week etc, etc.

#Heat map visualisation

```
custMat <- custBreakSum %>%
  select(recency, revenue, meanRevenue, medianRevenue, orders) %>%
  as.matrix()

rownames(custMat) <- custBreakSum$CustomerID

head(custMat)
```

```
##       recency revenue meanRevenue medianRevenue orders
## 12347       3 4310.00      615.71        584.91      7
## 12348      76 1797.24      449.31        338.50      4
## 12352      37 1545.41      140.49        160.33     11
## 12356      23 2811.43      937.14        481.46      3
## 12358       2 1168.06      584.03        584.03      2
## 12359       8 6245.53     1040.92        828.41      6
```

```
heatmap(scale(custMat), cexCol = 0.7)
```



The revenue clusters with the number of orders as we would expect, then the median and mean revenue are clustering together. Recency has its own group.

Conclusion: The dataset offers innumerable options to study the trends and perform various operartions. We first cleaned the dataset and posted some graph to notice the retail store trends and further analysed by clustering our customers. We first pulled out UK set and performed k means clustering which gave us a good insight as to where we have to improve. We further studied the next top 5 countries(dopping UK) and concluded for the same. There is a decline in revenue which can be corrected by some mail or online ads and with respect to UK the top tier customers can be awarded with some elite membership card, vouchers and exciting discount offers, mid tier with some discount and we need to work on our low tier profiles. A campaign with any form of ads can bring them back to us. There are lots of negative revenue which indicates there are some returns, we can futher analyse them and draw some conclusions.Futher analysis can be conducted on clusters to identiy more narrowed charactersitcs of the customers, understand the relationship between cluster and types of product purchased.We can build a new model using heriarchial clustering and compare both the models.