IXIS Data Science Challenge

Patrick Hearin

2022-08-28

Initial Setup

Load the libraries that will be used in this assignment.

```
library(readr)            # Read csv files.
library(dplyr)            # Manipulate data.
```

```
##
## 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(xlsx)             # Work with Excel spreadsheets.
library(lubridate)        # Work with dates.
```

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

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

```
library(ggplot2)          # Create visualizations.
library(gganimate)        # Animate visualizations.
library(gifski)           # Render animations at different frame rates.
library(png)              # Help with rendering.
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(stringr)
```

Read in the data.

```
adds_to_cart <- read_csv("DataAnalyst_Ecom_data_addsToCart.csv")       # Load the adds to cart csv.
```

```
## Rows: 12 Columns: 3
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): dim_year, dim_month, addsToCart
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
session_counts <- read_csv("DataAnalyst_Ecom_data_sessionCounts.csv")  # Load the sessions csv.
```

```
## Rows: 7734 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): dim_browser, dim_deviceCategory, dim_date
## dbl (3): sessions, transactions, QTY
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

View the full data set in a seperate tab.

```
View(adds_to_cart)    # Keep a reference to the adds to cart and sessions data.
View(session_counts)
```

Look at the structure of the data and the types.

```
str(session_counts) # Look at the types of variables.
```

```
## spec_tbl_df [7,734 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ dim_browser       : chr [1:7734] "Safari" "Internet Explorer" "Chrome" "Amazon Silk" ...
##  $ dim_deviceCategory: chr [1:7734] "tablet" "desktop" "tablet" "tablet" ...
##  $ dim_date          : chr [1:7734] "7/1/12" "7/1/12" "7/1/12" "7/1/12" ...
##  $ sessions          : num [1:7734] 2928 1106 474 235 178 ...
##  $ transactions      : num [1:7734] 127 28 3 4 6 7 0 0 0 0 ...
##  $ QTY               : num [1:7734] 221 0 13 5 11 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   dim_browser = col_character(),
##   ..   dim_deviceCategory = col_character(),
##   ..   dim_date = col_character(),
##   ..   sessions = col_double(),
##   ..   transactions = col_double(),
##   ..   QTY = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

View the first few columns of a data set.

```
head(session_counts)  # Look at the first couple rows to get to know the data.
```

```
## # A tibble: 6 x 6
##   dim_browser       dim_deviceCategory dim_date sessions transactions   QTY
##   <chr>             <chr>              <chr>       <dbl>        <dbl> <dbl>
## 1 Safari            tablet             7/1/12       2928          127   221
## 2 Internet Explorer desktop            7/1/12       1106           28     0
## 3 Chrome            tablet             7/1/12        474            3    13
## 4 Amazon Silk       tablet             7/1/12        235            4     5
## 5 Internet Explorer mobile             7/1/12        178            6    11
## 6 Internet Explorer tablet             7/1/12        120            7     0
```

Data Cleaning and Wrangling.

Caculate is the data set

```
any(is.na(session_counts)) # Check the data for missing values.
```

```
## [1] FALSE
```

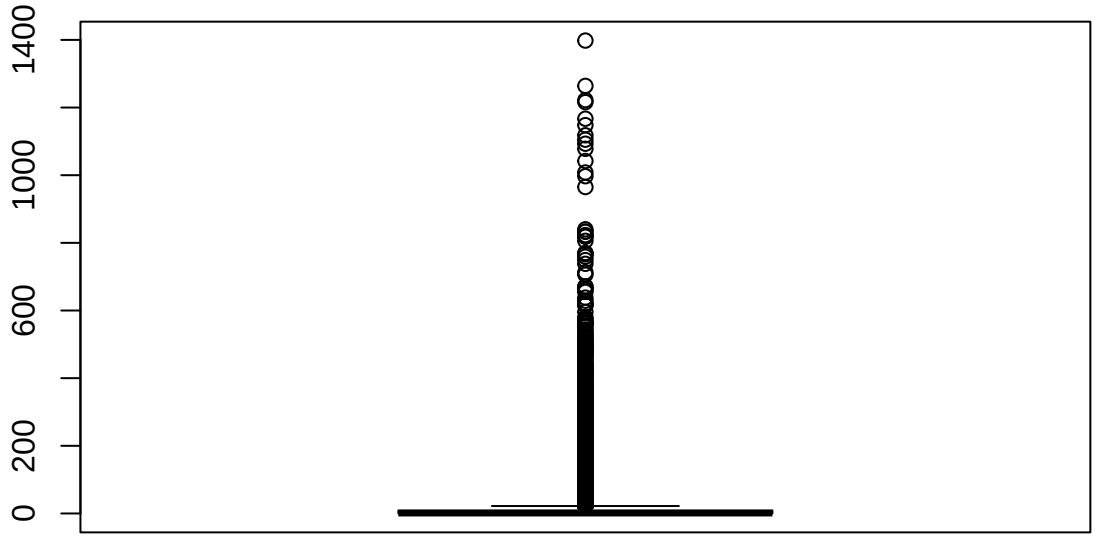
Next check if there is any duplicate data points.

```
any(duplicated(session_counts)) # Check the data for duplicates.
```

```
## [1] FALSE
```

Next the outliers are calculated. First the boxplot is visualized for the transactions feature. This often yields far to many outliers so another method is applied.

```
outlier_box_plot <- boxplot(session_counts$transactions) # Graph the boxplot for the transaction featur
```



The previous plot has to many outliers. These are printed out using the following script.

```
boxplot.stats(session_counts$transactions)$out  # Print all the outliers from the boxplot.
```

```
##    [1]  127   28  261  120  188  294  160  203  151  188  563  107   55  385
##   [15]  152  227  146   31  304  620   64   40  352   54   76  206  186  277
##   [29]  159   34   97  209   99   29   68   77   39   29  355   41  148   50
##   [43]  111   47   55   65  265  270  200  122   42   88   30  769   96  191
##   [57]   27  101  296  228  369   73   27   32  479  144   71   75  185  219
##   [71]  116   84   80   98   80  378  107   24   68   59 1148   90   28   29
##   [85]   47  428   62   67   87  225   50   35   30  551   25   29  116   40
##   [99]   31   27  253  184   28   40  106  202   76   55  162  297   48   27
```

```
##  [113]  103  104   38  343   75  146  136   67  372   56   25   35  223  171
##  [127]  109   37   32  150  199  239   23   53   29   59  162  205   63   55
##  [141]   72  127  381   82   92  247  174  133   23  429  178   38   91   55
##  [155]  818  240   34   86   84   82  187  121  403  115  126  263  108   82
##  [169]   57  214  567  192   59  147   36   38  348  203   35  287  239  257
##  [183]   87   27   90  209   35  130  116  105   83  310  129   30  125   27
##  [197]  713   75   72   77  128  133  449  128   30   38  243  145   46   62
##  [211]   38   49  144  102  213  131   30  127  407   52   29   29  412  245
##  [225]   97  182   29   39  203  240  476  327   40   32  179  134   96  173
##  [239]  105   69   43   53  105   85   98   46   45  268   59   39  253  326
##  [253]  113   65  182   74   74  108   56   58   78   96   68   24  339   54
##  [267]  163   50  361  436  260   89   42   84  252  531  148  196   27  246
##  [281]   97   78   44   81  185   27  283  130   31   37   64  343  105  164
##  [295]   75   58  240   27   49  241  126   98   58   26  303  194  132  217
##  [309]  250   78   25  237  227   50  123   47  219  301   82  544   76   29
##  [323]   88  117  208   53  438  152   33   58  156   84  107   44   37  425
##  [337]  382  221   25   24  339  195   90   45   82  112   24  317  110   62
##  [351]  154  275   76  126  111   25  266  114   26  253   61   60   90   38
##  [365]  126  155   32   81   32  125  142  162   64   42   32  469  137   43
##  [379]   44  166  155   32  484  111   45   70   75  173   30  260   63  169
##  [393]   57  136  361  425   87   25   26   72   34  162   96   50  156  114
##  [407]  319  120  303  108   70   55  146  295   66   31   60   80   40  159
##  [421]  662   67   82   41  206  178   71  145   89   81   23  112  132  126
##  [435]   60  189  206  207  102   55  110  115   32  129  281  250  374  109
##  [449]   38   37  202  197   23  112  100   24  372  522   81   31   70  154
##  [463]  147  237   68   23   35  189   39  212  132  165   23  399   66  177
##  [477]  118   27  430   80   23   57  226   61   49  285  191  248   73   47
##  [491]   35   65   34  217   36   50   37   35  192  580  274   63   37  194
##  [505]   65   30  201   40   66  303  125   82   61   44  497  103   49   38
##  [519]  180   92   99   35   55  310   41   23   41   85   85   40  137   62
##  [533]   49  175  102   92   51   68  343  188   50   59  500  106   65   30
##  [547]  218   91  116  105   37   32  347   36   24  259 1167  141   98  236
##  [561]   23  182   28  167  343  223  153   25  474   41  423  142   50   25
##  [575]   23  249  201   73  170  749   50  149   56   43   31   32   72   69
##  [589]   84  107   81  106  146   30   85  185   59   31  427  473   49   83
##  [603]   39   80  216   28   57   27   49  272  122   67  254  329   62   83
##  [617]   40  315   64   44   44   23  134   23   26  164  167   72  297  124
##  [631]   72   29  532  423  575  174  189   30   66   26  489   38  561  138
##  [645]   88  169  127   28  137   76   31   98   64   42  120  213  109   57
##  [659]  160  209  210  108  234   38   25  374  124  234   76   36 1118   76
##  [673]   27  153   29  196  402  191   27   34  403  113  389   48   27  109
##  [687]  427  159  182  149   61  105   64   56  143   74   38  385   64   55
##  [701]  209  150  107   36   36  820  287  231   45  380  178   40 1222 1105
##  [715]  105   83   38  358  128   34   74   52   28  306  373  213   87   34
##  [729]   68  197  129  108  155  124   24   51  347  351   43  661  165  162
##  [743]   44   31   37  175   36   39  147  125  155   44   41  121   81  136
##  [757]   86   82   31  236  120  137  233  291   90  232  186   81  840  331
##  [771]   73  211   98   86  409   88   96  478   71   23   32  263  244   30
##  [785]  130  316   61  119  182   68  327  215  119  145  100   55   23   35
##  [799]  200   90  197   63   72   60  265   30   55   64  179  288   48   31
##  [813]  115  357   31  385  343   73  519   31   64  444   64   71   32   25
##  [827]   99  476   86   30   25   34   47  206   75  136   76  468  164   72
##  [841]   89   36  113  162   94   98   72  291  117  833   63  310  137  224
##  [855]   79   43  136  208   51  344  214  239  171  158   52   98   24   39
```

```
##  [869]  138  230  224   78  155   35   49   26   69  289   57   58  107   86
##  [883]   39  357   68   33  347  139  120  103  571   87   43  226  102   37
##  [897]   92   39   29  472   45  361  139  149  116  353  227   58  481  141
##  [911]   43   25  108  231  108   93   70   41  125  167   91   91   52  271
##  [925]   68  130   28   60   98   41  377  216  106  262  128   56  493  121
##  [939]  118   53   27  205  227   28  234  196   79   33  261  328  116  122
##  [953]   61   94   64   62   30  216  137   45   23  110   66   83   51  329
##  [967]  163   30   79   45  616  214  132  130  135  418  263  469  121  212
##  [981]   69   48   57  129  157  134   27  280  142  122   54  169  198  231
##  [995]   49   23  186   80   81  117  766   68   71   90  161  482   46   70
## [1009]   54   56  490  329  425  196   39   66   27   64  226  193  134  181
## [1023]  124   52   25   27  374  214  179   24   36   69   44  541  211  148
## [1037]  412  125   55   30  315  342  443  334  154   29   97   29  515  161
## [1051]   34  126   88   89   24  507  833  144   82   62   96   25  349   86
## [1065]  135   39   70   37   23   55 1264  351   72  168  138   26   30  277
## [1079]  141  486  129   53   28   38  655  260  186   48   25   37  182  210
## [1093]  121   32   68  638  241  485   63   40   23  493  629  214  140  134
## [1107]   43   65  349  287  164   30  396   25   62   83   71   45  226  210
## [1121]  193  111  286   38  120   69  128  242  806  247   73   89   26   58
## [1135]   47   36  107  156   93  267   55  113   73  270   73  412  119   63
## [1149]   87  217  172   94  118   54   37  239   60  596   94   25   31   30
## [1163]  290   69   74   45  180  114  123   61   37   46  205  318  119   76
## [1177]   38  238   53   42   83   33  203  137   40   35  271  177   45   31
## [1191]  120  434  171   43   49  287   89  159  427  108  168   23  225  531
## [1205]  108   81   26  404   56   53   34   25  131  142   26   70   66  178
## [1219]   96   69   35  965  496   45   53  130   57   50   44  340  315  380
## [1233]  351   60   98   33  387  159  291  320   37   48  129  146  161   24
## [1247]  820  997  303  112  113  107   74   33  159  159   56   32   26  314
## [1261]  195   58   43  207  236   82  248   48   30  277   66  324   79   52
## [1275] 1042  770  343  369   71  157  208  145   59  162  177  368   24   49
## [1289]   40  102  295   56   24   41   80   73   28  556 1078  202  146  146
## [1303]   31   37   57  201  360  150   89   31 1008  420   32   28   41   74
## [1317]   40  299  706  362  352  112   68   25  442  273  217  169   44   86
## [1331]   28  403  347   78   36  269  164   88   30  304  517   32  115   99
## [1345]   99   68  615   54  185  140   81   26  107   96   84   54   60  416
## [1359] 1398 1216  508   37   62  261  563  310  194   61   28   26  193  563
## [1373]  368   92   40   90   45   46   25   67   41   59   47   50   33   39
## [1387]  236  670  217   93   24  458  671  252   39  218  114  109   90  103
## [1401]   23  458  159  137   77   63  410  145  135   62   60  231  325  759
## [1415]  110   25  490  738  230  159   42   55   48   31 1093  436  216  223
## [1429]   98   71   28  130  200  493  125   49   54   24  264  621  430  199
## [1443]   47  130   25   61  131  213   33   42  350  504   93   64  223  456
## [1457]   53   92   38   29  382  138  244   37   33  239   42   32  118  236
## [1471]   85   69   38  823  180   37   35   27   84  229  204   74   89   54
## [1485]   46
```

Since there is far to many outliers from the boxplot calculation quantiles will be used. Setting a small value for the quantile yields the worst outliers in the data.

```
lower_bound <-  quantile(session_counts$transactions, 0.001)  # Set the lower bound for the data.

upper_bound <- quantile(session_counts$transactions, 0.999) # Set the upper bound for the data.

# Calculate the rows that are outside the bounds.
```

```
outlier <- which(session_counts$transactions < lower_bound | session_counts$transactions > upper_bound)

# Print out the rows.

outlier
```

```
## [1]  408 2934 3542 3764 3765 5649 7123 7124
```

```
# Remove these rows from the data set.

session_counts_clean <- session_counts[-outlier, ]
```

```
print(dim(session_counts)) # Print out the original data length.
```

```
## [1] 7734    6
```

```
print(dim(session_counts_clean)) # Print out length of the data without outliers.
```

```
## [1] 7726    6
```

The calculation is executed for the sessions feature.

```
lower_bound1 <-  quantile(session_counts_clean$sessions, 0.001) # Set the lower bound for the data.

upper_bound1 <- quantile(session_counts_clean$sessions, 0.999) # Set the upper bound for the data.

# Calculate the rows that are outliers.

outlier1 <- which(session_counts_clean$sessions < lower_bound1 | session_counts_clean$sessions > upper_

#print out the rows.

outlier1
```

```
## [1]   89  140 3517 3740 5585 6703 6704 6812
```

```
# Remove the outlier rows from the data.

session_counts_clean1 <- session_counts_clean[-outlier1, ]
```

```
# Compare the lengths to make sure the data was removed.

print(dim(session_counts_clean))
```

```
## [1] 7726    6
```

```
print(dim(session_counts_clean1))
```

```
## [1] 7718    6
```

Finally, the calculation is done for the quantity feature.

```
lower_bound2 <-  quantile(session_counts_clean1$QTY, 0.001) # Set the lower bound for the data.

upper_bound2 <- quantile(session_counts_clean1$QTY, 0.999) # Set the upper bound for the data.

# Calculate the rows that are outliers.

outlier2 <- which(session_counts_clean1$QTY < lower_bound2 | session_counts_clean1$QTY > upper_bound2)
```

```
# Print the rows.

outlier2
```

```
## [1]  768 3720 4507 6542 6805 6866 7373 7675
```

Next remove the outliers from the data and define the final clean data.

```
# Remore outliers for the final clean data set.

session_counts_clean_final <- session_counts_clean1[-outlier2, ]
```

Compare the length of the final data set.

```
print(dim(session_counts_clean1))
```

```
## [1] 7718    6
```

```
print(dim(session_counts_clean_final))
```

```
## [1] 7710    6
```

Use the strsplit function to split the data at the forward slash. Then add those columns to the data set.

```
date_split <-  strsplit(session_counts_clean_final$dim_date, "/") # Split the data on the forward slash

date_matrix <- matrix(unlist(date_split),ncol = 3,byrow=T) # Create a matrix from the split data.

date_df <- data.frame(date_matrix) # Make a data frame from the matrix.


# Add the data to the original matrix.


session_counts_clean_final$Month <-  date_df$X1

session_counts_clean_final$Day <-  date_df$X2

session_counts_clean_final$Year <-  date_df$X3
```

Calculate the ECR and add it to the dataframe.

```
session_counts_clean_final$ECR <- session_counts_clean_final$transactions/session_counts_clean_final$ses
```

Ca

Make the dim_data column into data type.

```
# Use the mdy function to covert the dim_date feature to the date type.

session_counts_clean_final$dim_date <- mdy(session_counts_clean_final$dim_date)
```

Calculations

The first deliverable, the month device, deliverable will be calculated.

Get rid of all the data that isn't going to be aggregated.

```
df1 <-session_counts_clean_final[c(2,4:7,10)] # Use the subset of the columns in the vector.
```

View the first couple rows of the data to make sure that it is acceptable.

```
# Display the data.

head(df1)
```

```
## # A tibble: 6 x 6
##   dim_deviceCategory sessions transactions   QTY Month     ECR
##   <chr>                 <dbl>        <dbl> <dbl> <chr>   <dbl>
## 1 tablet                 2928          127   221 7     0.0434
## 2 desktop                1106           28     0 7     0.0253
## 3 tablet                  474            3    13 7     0.00633
## 4 tablet                  235            4     5 7     0.0170
## 5 mobile                  178            6    11 7     0.0337
## 6 tablet                  120            7     0 7     0.0583
```

Calculate the sum aggregation grouping by month and device type.

```
# Calculate the aggregation by grouping the month and device category variables.

df_month_device <- df1 %>%
  group_by(Month, dim_deviceCategory) %>%
  summarise_all(list(sum), na.rm=TRUE)

# Order by the month.

df_month_device[order(df_month_device$Month),]
```

```
## # A tibble: 36 x 6
## # Groups:   Month [12]
##    Month dim_deviceCategory sessions transactions   QTY     ECR
##    <chr> <chr>                 <dbl>        <dbl> <dbl>   <dbl>
##  1 1     desktop              349075        11466 21336   4.97
##  2 1     mobile               301423         3980  6523   0.806
##  3 1     tablet               141483         2587  4501   2.73
##  4 10    desktop              302682         9373 17675   3.45
##  5 10    mobile               238849         2418  4446   0.711
##  6 10    tablet               107108         2484  4505   2.38
##  7 11    desktop              320717        10350 18778   3.56
##  8 11    mobile               178828         1994  3407   1.09
##  9 11    tablet               138235         3183  5947   1.63
## 10 12    desktop              274877         9328 16194 Inf
## # ... with 26 more rows
```

The data frame df_month_device is the first deliverable asked to be calculated.

Next the second deliverable will be calculated.

Find the maximum data in the data.

```
# Use the date type to calculate the last date.

max(session_counts_clean_final$dim_date)
```

```
## [1] "2013-06-30"
```

Since the maximum year is 2013 and the maximum month is June use a Boolean to calculate the data that is from the last month and the one previous.

```
# Make a subset with the last month and the one before it using a Boolean.

df_last_month <-session_counts_clean_final[(session_counts_clean_final$Month == "6" & session_counts_cl
```

Take the data needed to compute the final table.

```
# Use a vector to compute the subset needed.

df_last_month_final <-  df_last_month[c(4:7,10)]
```

Compute the sum of all the data grouping by month.

```
# Aggregate all the data during those months.

df_comparison <- df_last_month_final %>%
  group_by(Month) %>%
  summarise_all(list(sum), na.rm=TRUE)

# Order by the month.

df_comparison[order(df_comparison$Month),]
```

```
## # A tibble: 2 x 5
##   Month sessions transactions   QTY   ECR
##   <chr>    <dbl>        <dbl> <dbl> <dbl>
## 1 5       976182        22938 41253  13.3
## 2 6      1307855        30342 54178  10.9
```

Use the same Boolean to extract them months needed from the adds to cart data frame.

```
# Use the same boolean to calculate the months required from the adds_to_cart data set.

df_cart <- adds_to_cart[(adds_to_cart$dim_month == "6" & adds_to_cart$dim_year == "2013") | (adds_to_ca
```

Rename the dim_month column.

```
# Change the column name.

df_cart$Month <-df_cart$dim_month
```

Take only the data needed.

```
# Use a vector to take a subset of the data.

df_cart_final <- df_cart[c(3,4)]
```

Put the add to cart column in the comparison data frame.

```
# Add the final column to the data set.

df_comparison$addsToCart <- df_cart_final$addsToCart
```

Check the final data frame.

```
head(df_comparison)
```

```
## # A tibble: 2 x 6
##   Month sessions transactions   QTY   ECR addsToCart
##   <chr>    <dbl>        <dbl> <dbl> <dbl>      <dbl>
## 1 5       976182        22938 41253  13.3     136720
```

```
## 2 6      1307855        30342 54178  10.9     107970
```

The df_comparison data_frame is the final result for the second deliverable.

Now the two calculation will be exported to a single Excel file with two worksheets.

```
# Convert the Tibble to a data_frame.

df_month_device_final <- data.frame(df_month_device)
```

Use the xlsx2 function from the xlsx package to write the excel spreadsheet.

```
# Write to the Excel file the two worksheets.

write.xlsx2(df_month_device_final, file = "client_deliverable.xlsx", sheetName = "Month_Device_Aggregati

write.xlsx2(df_comparison, file = "client_deliverable.xlsx", sheetName = "Month_Comparison", append = TR
```

EDA

In this section I dive into some EDA to try to find a proper story from the data. I could have created other files but I will end the analysis at what was requested and proceed to present my data visualization skills with R.

The following plots are from the comparison of the consecutive months data frame. Below is a comparison of the sessions.

First replace the numbers with the month name.

```
# Replace the numberic month with the name.

df_comparison['Month'][df_comparison['Month'] == 5] <- "May"


df_comparison['Month'][df_comparison['Month'] == 6] <- "June"
```

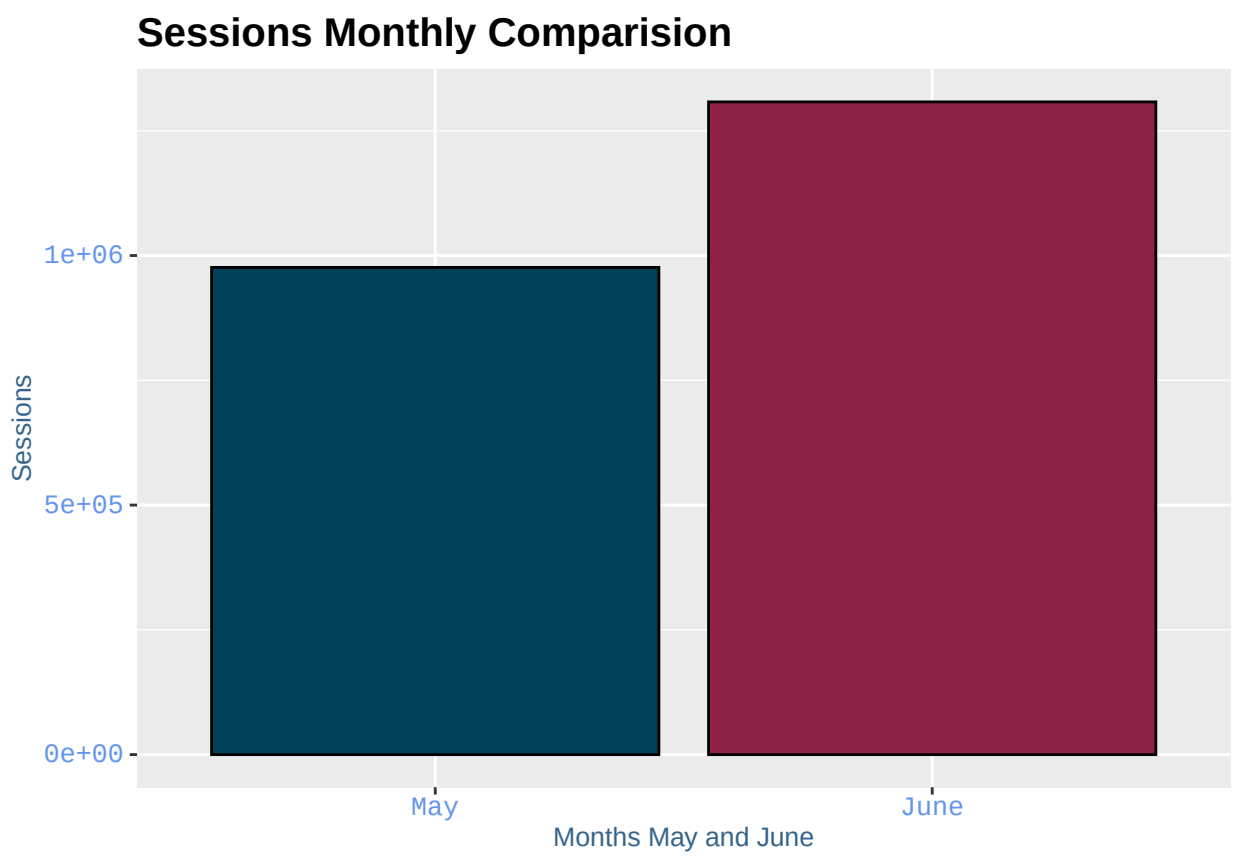
Next define the theme for the visualizations.

```
myTheme <- theme(
  plot.title = element_text(family = "Helvetica", face = "bold", size = (15)),
  legend.title = element_text(colour = "steelblue", face = "bold.italic", family = "Helvetica"),
  legend.text = element_text(face = "italic", colour = "steelblue4", family = "Helvetica"),
  axis.title = element_text(family = "Helvetica", size = (10), colour = "steelblue4"),
  axis.text = element_text(family = "Courier", colour = "cornflowerblue", size = (10))
)
```

```
# Plot the graph reorder such that the months are sequential.
# Fill based on the month to have two different colors.

fig_comparision <- ggplot(df_comparison, aes(x=reorder(Month, sessions), y=sessions, fill = Month)
               )+geom_col(show.legend = TRUE, alpha=1, colour = "black")+ scale_colour_viridis_b() +
                scale_fill_manual(values = c("#902147", "#004159")) +
                xlab("Months May and June")+
                ylab("Sessions")+
                ggtitle("Sessions Monthly Comparision")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
print(fig_comparision+myTheme)
```

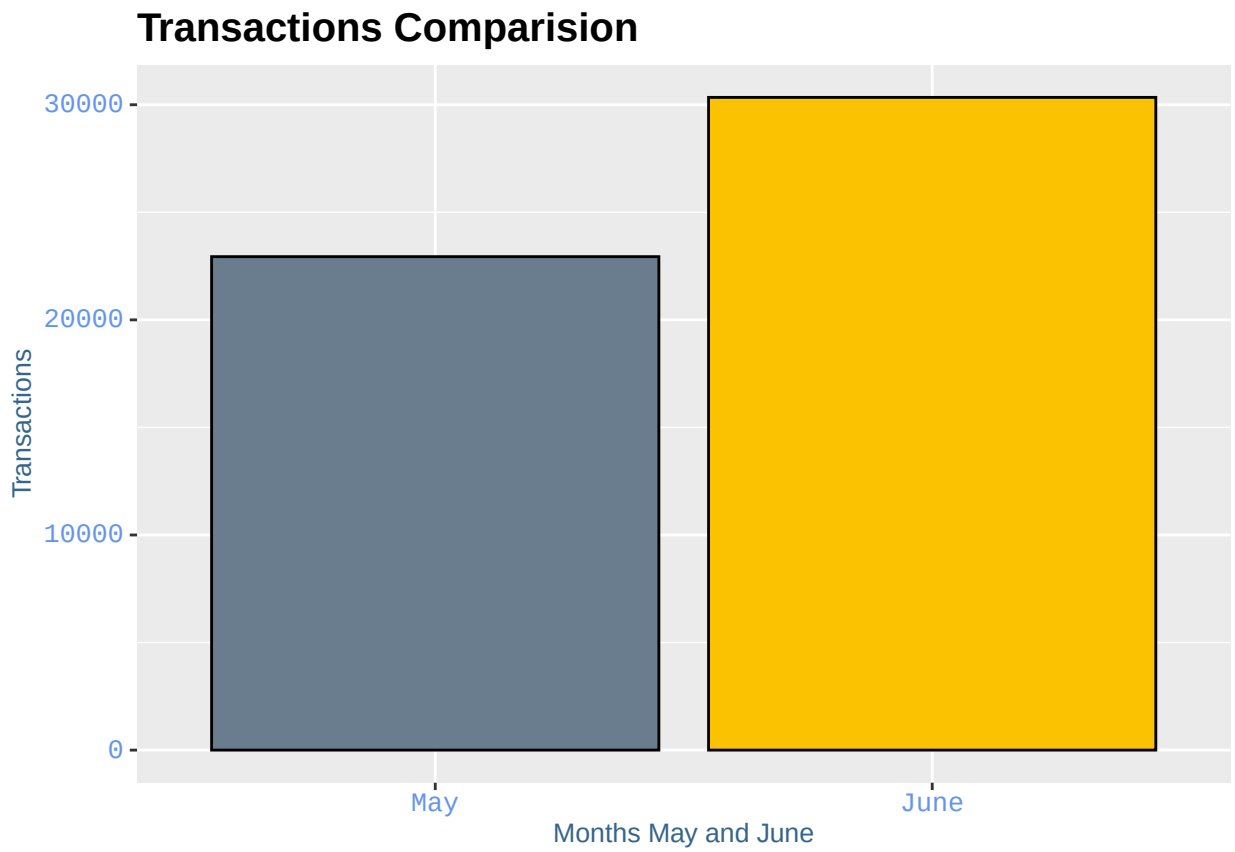


The next consecutive month comparison is over the transactions.

```
# Plot the graph reorder such that the months are sequential.
# Fill based on the month to have two different colors.

fig_comparision2 <- ggplot(df_comparison, aes(x=reorder(Month, transactions), y=transactions, fill = Mo
               )+geom_col(show.legend = TRUE, alpha=1, colour = "black")+ scale_colour_viridis_b() +
                scale_fill_manual(values = c("#FAC200", "#6A7D8E")) +
                xlab("Months May and June")+
                ylab("Transactions")+
                ggtitle("Transactions Comparision")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
print(fig_comparision2+myTheme)
```

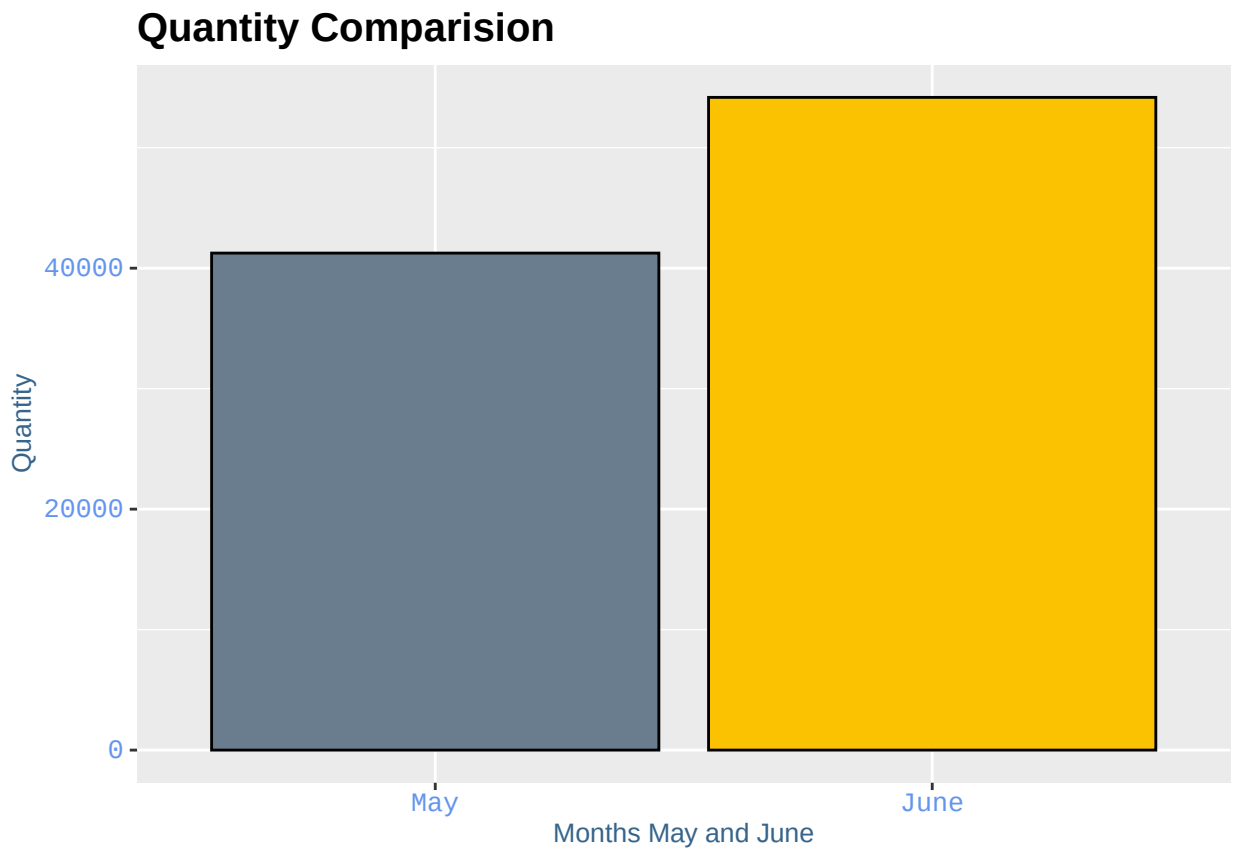


The next consecutive month comparison is over the quantity.

```
# Plot the graph reorder such that the months are sequential.
# Fill based on the month to have two different colors.

fig_comparision3 <- ggplot(df_comparison, aes(x=reorder(Month, QTY), y=QTY, fill = Month)
               )+geom_col(show.legend = TRUE, alpha=1, colour = "black")+ scale_colour_viridis_b()+
                scale_fill_manual(values = c("#FAC200", "#6A7D8E")) +
                xlab("Months May and June")+
                ylab("Quantity")+
                ggtitle("Quantity Comparision")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
print(fig_comparision3+myTheme)
```

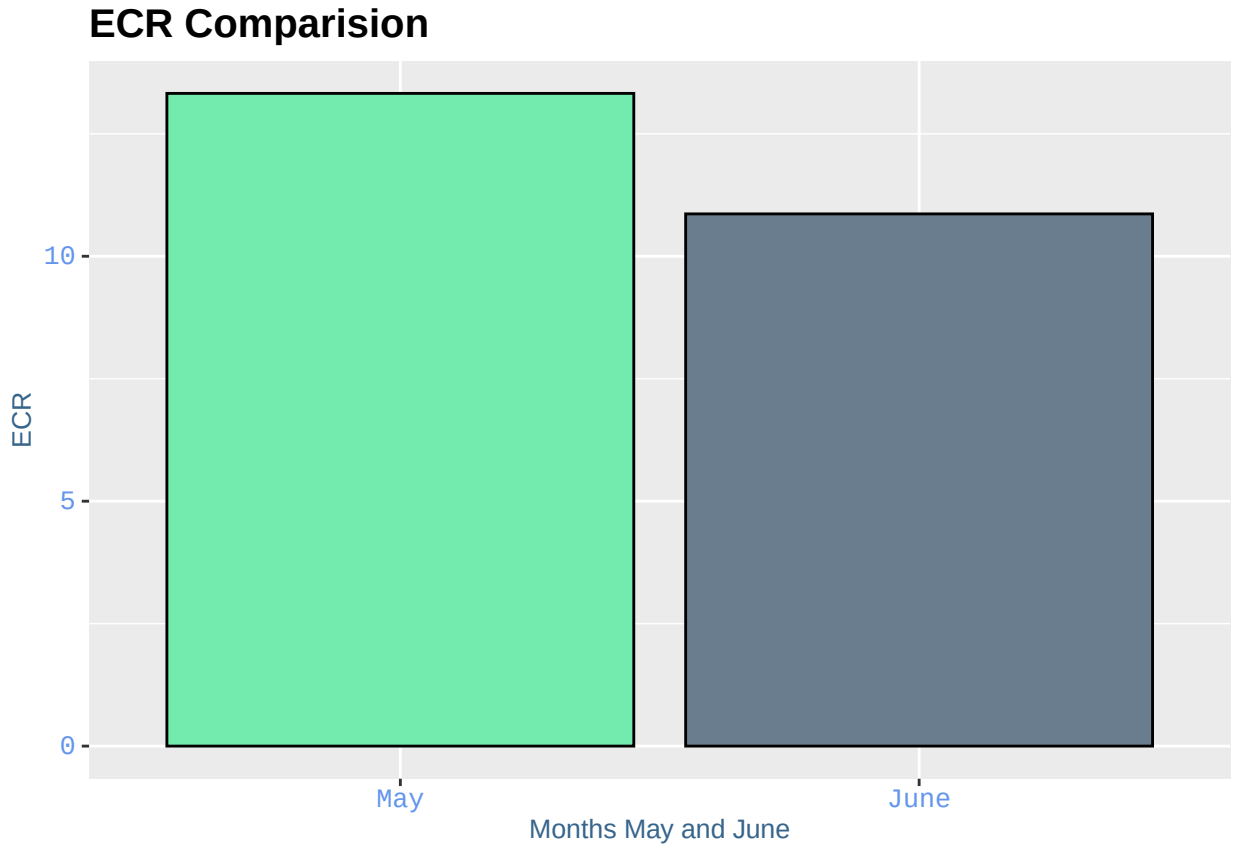


The next consecutive month comparison is over ECR.

```
# Plot the graph reorder such that the months are sequential.
# Fill based on the month to have two different colors.

fig_comparision4 <- ggplot(df_comparison, aes(x=reorder(Month,-ECR), y=ECR, fill = Month)
               )+geom_col(show.legend = TRUE, alpha=1, colour = "black")+ scale_colour_viridis_b()+
                scale_fill_manual(values = c("#6A7D8E", "#73EBAE")) +
                xlab("Months May and June")+
                ylab("ECR")+
                ggtitle("ECR Comparision")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
print(fig_comparision4+myTheme)
```

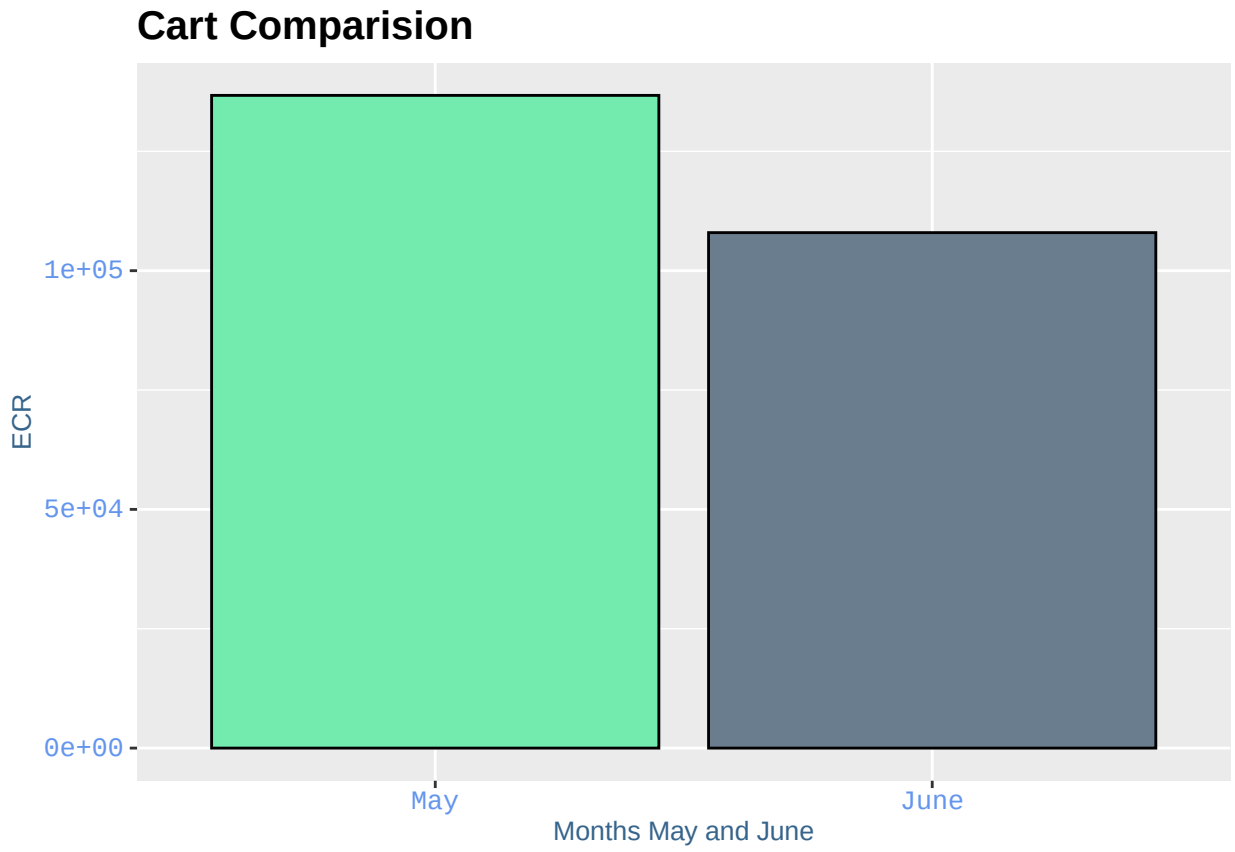


Finally the consecutive month comparison is over the addsToCart feature.

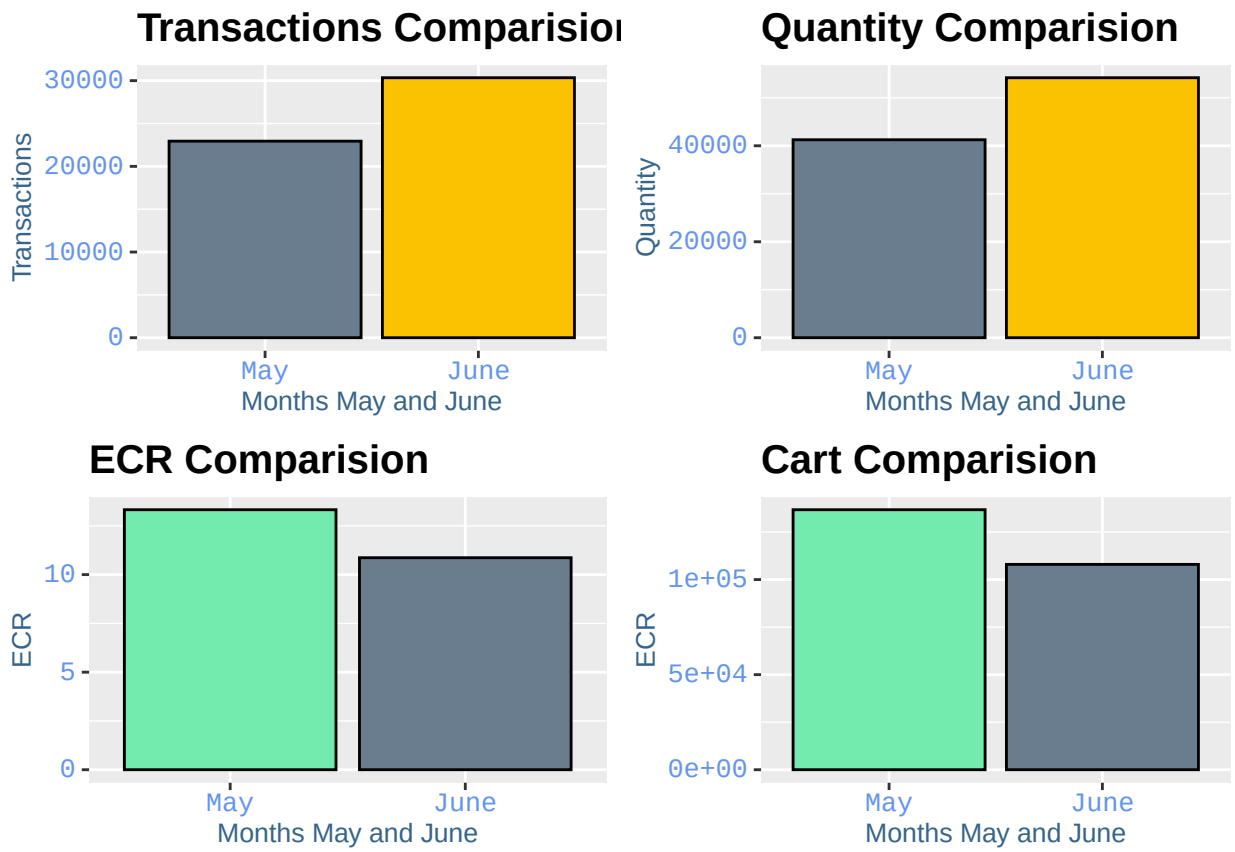
```
# Plot the graph reorder such that the months are sequential.
# Fill based on the month to have two different colors.

fig_comparision5 <- ggplot(df_comparison, aes(x=reorder(Month,-addsToCart), y=addsToCart, fill = Month)
               )+geom_col(show.legend = TRUE, alpha=1, colour = "black")+ scale_colour_viridis_b()+
                 scale_fill_manual(values = c("#6A7D8E", "#73EBAE")) +
                xlab("Months May and June")+
                ylab("ECR")+
                ggtitle("Cart Comparision")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
print(fig_comparision5+myTheme)
```




```
grid.arrange(
fig_comparision2+myTheme,
fig_comparision3+myTheme,
fig_comparision4+myTheme,
fig_comparision5+myTheme)
```



Look at the month device aggregation to find to start the next visualization.

```
head(df_month_device_final)
```

```
##   Month dim_deviceCategory sessions transactions   QTY       ECR
## 1     1            desktop   349075        11466 21336 4.9682245
## 2     1             mobile   301423         3980  6523 0.8058093
## 3     1             tablet   141483         2587  4501 2.7322455
## 4    10            desktop   302682         9373 17675 3.4530783
## 5    10             mobile   238849         2418  4446 0.7105307
## 6    10             tablet   107108         2484  4505 2.3792474
```

Concatenate the month and device category to graph them.

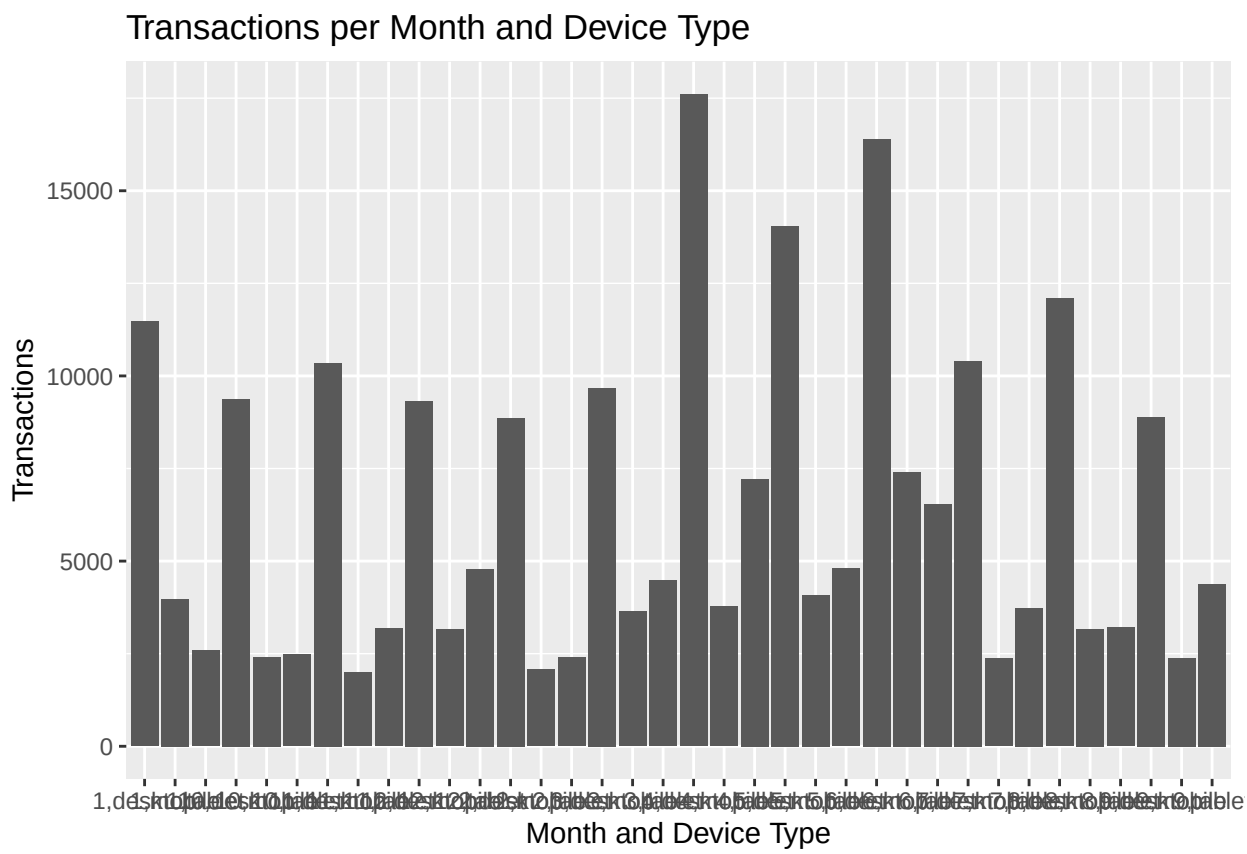
```
# Use the str_c function from the stringr library.

df_month_device_final$Month_Device1 = str_c( df_month_device_final$Month,",",df_month_device_final$dim_
```

Visualize the number of transactions per month.

```
fig_month_device <- ggplot(df_month_device_final, aes(x=Month_Device1, y=transactions, full=Month_Devic
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+
                scale_fill_manual(values = c("#6A7D8E", "#73EBAE")) +
                xlab("Month and Device Type")+
                ylab("Transactions")+
                ggtitle("Transactions per Month and Device Type")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
fig_month_device
```



Find the largest amount of transactions per month and device type.

```
which(df_month_device_final$transactions > 12000)
```

```
## [1] 19 22 25 31
```

Print out the rows that have the maximum transactions.

```
print(df_month_device_final[19,]$Month_Device1)
```

```
## [1] "4,desktop"
```

```
print(df_month_device_final[22,]$Month_Device1)
```

```
## [1] "5,desktop"
```

```
print(df_month_device_final[25,]$Month_Device1)
```

```
## [1] "6,desktop"
```

```
print(df_month_device_final[31,]$Month_Device1)
```

```
## [1] "8,desktop"
```

Use a Boolean to calculate a subset with only those rows.

```
# Use a Boolean to calculate a subset of the data.

df_month_device_sub <- df_month_device_final[ df_month_device_final$Month_Device1 == "4,desktop" | df_m
```

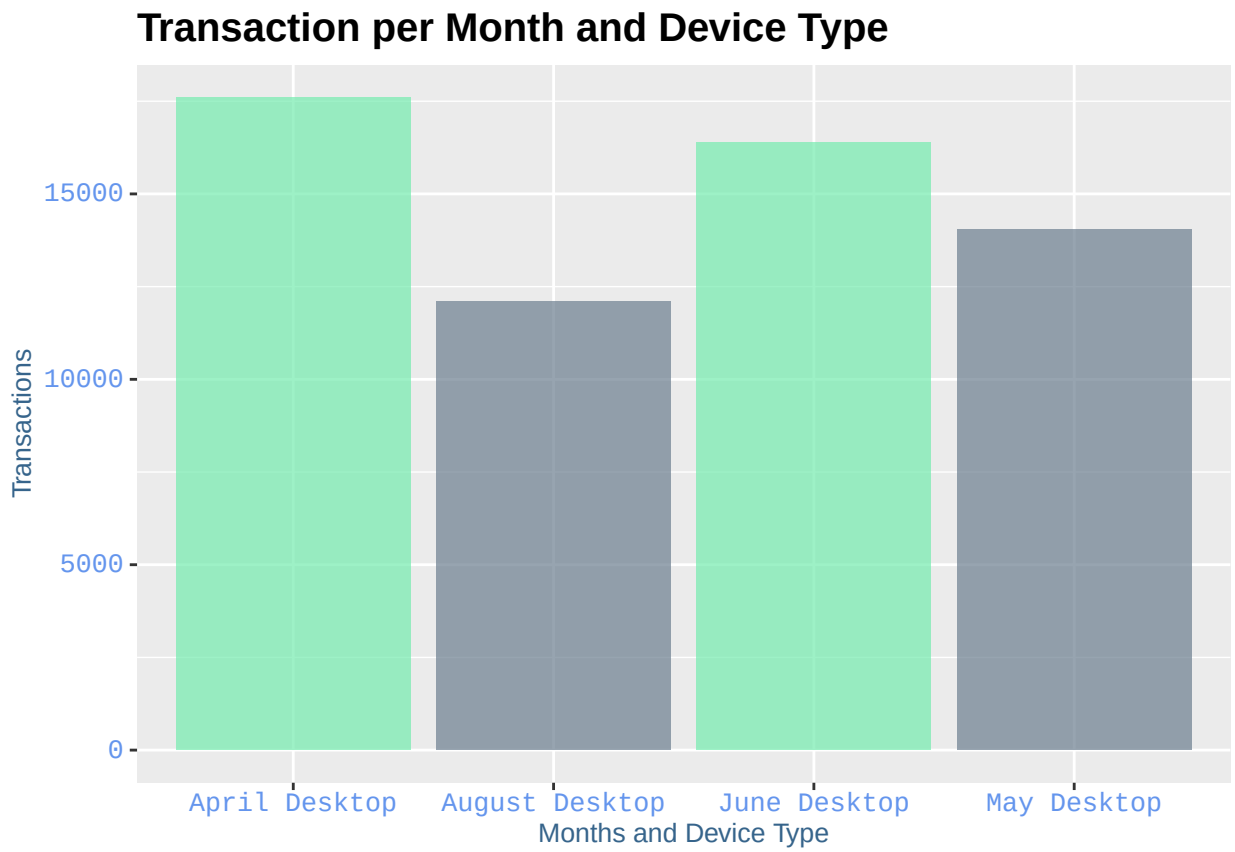
```
# Replace the month and device types with a better name.

df_month_device_sub['Month_Device1'][df_month_device_sub['Month_Device1'] == '4,desktop'] <- "April Desk


df_month_device_sub['Month_Device1'][df_month_device_sub['Month_Device1'] == '5,desktop'] <- "May Deskto


df_month_device_sub['Month_Device1'][df_month_device_sub['Month_Device1'] == '6,desktop'] <- "June Deskt


df_month_device_sub['Month_Device1'][df_month_device_sub['Month_Device1'] == '8,desktop'] <- "August Des
```

Graph those rows.

```
# Make a visualization of the four maximum months.

fig_month_device1 <- ggplot(df_month_device_sub, aes(x=Month_Device1, y=transactions, fill=Month_Device
               )+geom_col(show.legend = FALSE, alpha=0.7)+ scale_colour_viridis_b()+ myTheme +
              scale_fill_manual(values = c("#73EBAE", "#6A7D8E","#73EBAE", "#6A7D8E")) +
                xlab("Months and Device Type")+
                ylab("Transactions")+
                ggtitle("Transaction per Month and Device Type")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
fig_month_device1
```



Next drill down into the data by aggregating month, device type, and browser.

```
df2 <-session_counts_clean_final[c(1,2,4:7,10)] # Use the subset of the columns in the vector.
```

Aggregate over month, device type, and browser.

```
# Calculate the aggregation by grouping the month, device category, and browser variables.

df_month_device_browser <- df2 %>%
  group_by(Month, dim_deviceCategory, dim_browser) %>%
  summarise_all(list(sum), na.rm=TRUE)
```

Check the data frame that was aggregated.

```
head(df_month_device_browser)
```

```
## # A tibble: 6 x 7
## # Groups:   Month, dim_deviceCategory [1]
##   Month dim_deviceCategory dim_browser        sessions transactions   QTY   ECR
##   <chr> <chr>              <chr>                 <dbl>        <dbl> <dbl> <dbl>
## 1 1     desktop            Android Webview           1            0     0 0
## 2 1     desktop            Apple-iPhone7C2           0            0     0 0
## 3 1     desktop            Chrome               131743         4316  8133 0.602
## 4 1     desktop            Coc Coc                   2            0     0 0
## 5 1     desktop            DDG-Android-3.0.11        2            0     0 0
## 6 1     desktop            DDG-Android-3.1.1         1            0     0 0
```

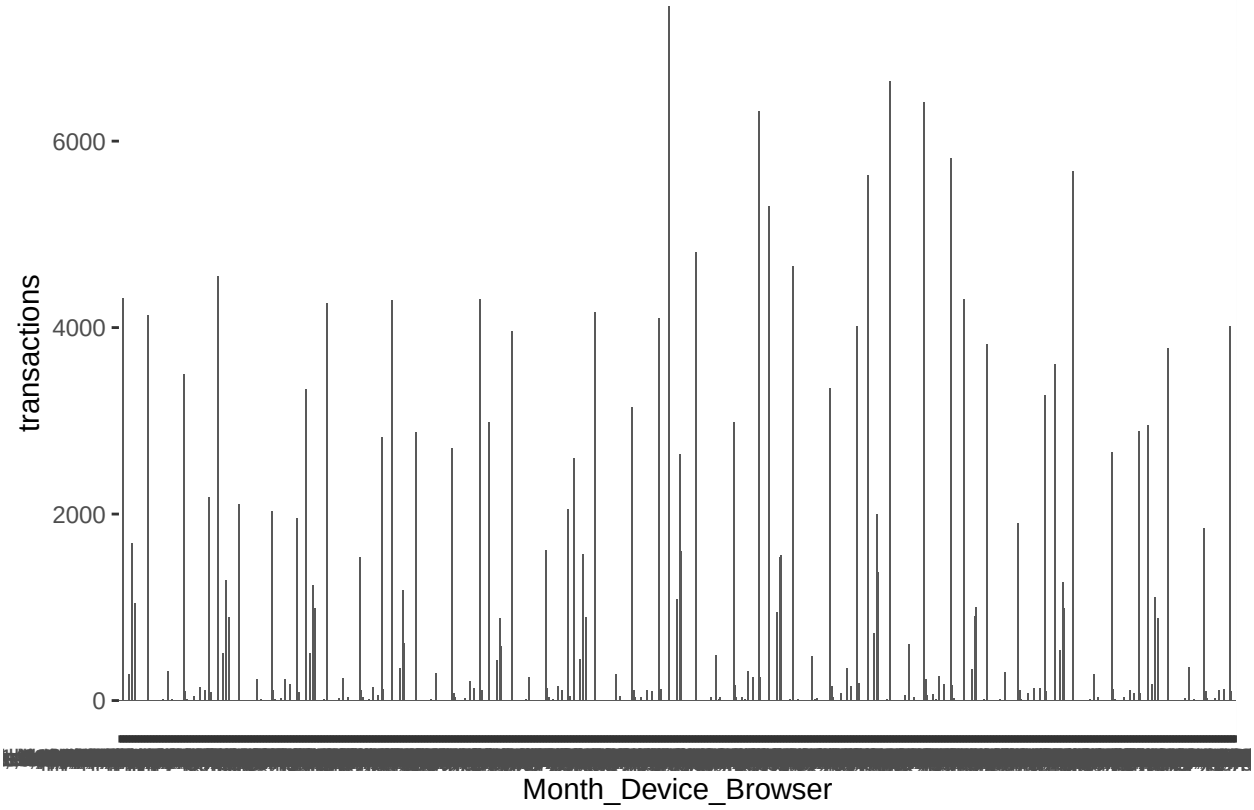
Concatenate the month and device type, Then concatenate that column with browwer.

```
df_month_device_browser$Month_Device = str_c( df_month_device_browser$Month,",",df_month_device_browser$
```

```
df_month_device_browser$Month_Device_Browser = str_c( df_month_device_browser$Month_Device,",",df_month_
```

Graph all of the different aggregations.

```
fig_month_device_browser <- ggplot(df_month_device_browser, aes(x=Month_Device_Browser, y=transactions)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()

fig_month_device_browser
```



Find the maximum aggregations.

```
which(df_month_device_browser$transactions > 6000)
```

```
## [1] 370 430 518 541
```

Print the rows with the largest aggregations.

```
print(df_month_device_browser[370,]$Month_Device_Browser)
```

```
## [1] "4,desktop,Chrome"
```

```
print(df_month_device_browser[430,]$Month_Device_Browser)
```

```
## [1] "4,tablet,Safari"
```

```
print(df_month_device_browser[518,]$Month_Device_Browser)
```

```
## [1] "6,desktop,Safari"
```

```
print(df_month_device_browser[541,]$Month_Device_Browser)
```

```
## [1] "6,mobile,Safari"
```

Use a Boolean to calculate only those rows.

```
# Use a Boolean to calculate a subset of the data.

df_month_device_browser_sub <- df_month_device_browser[ df_month_device_browser$Month_Device_Browser ==
```

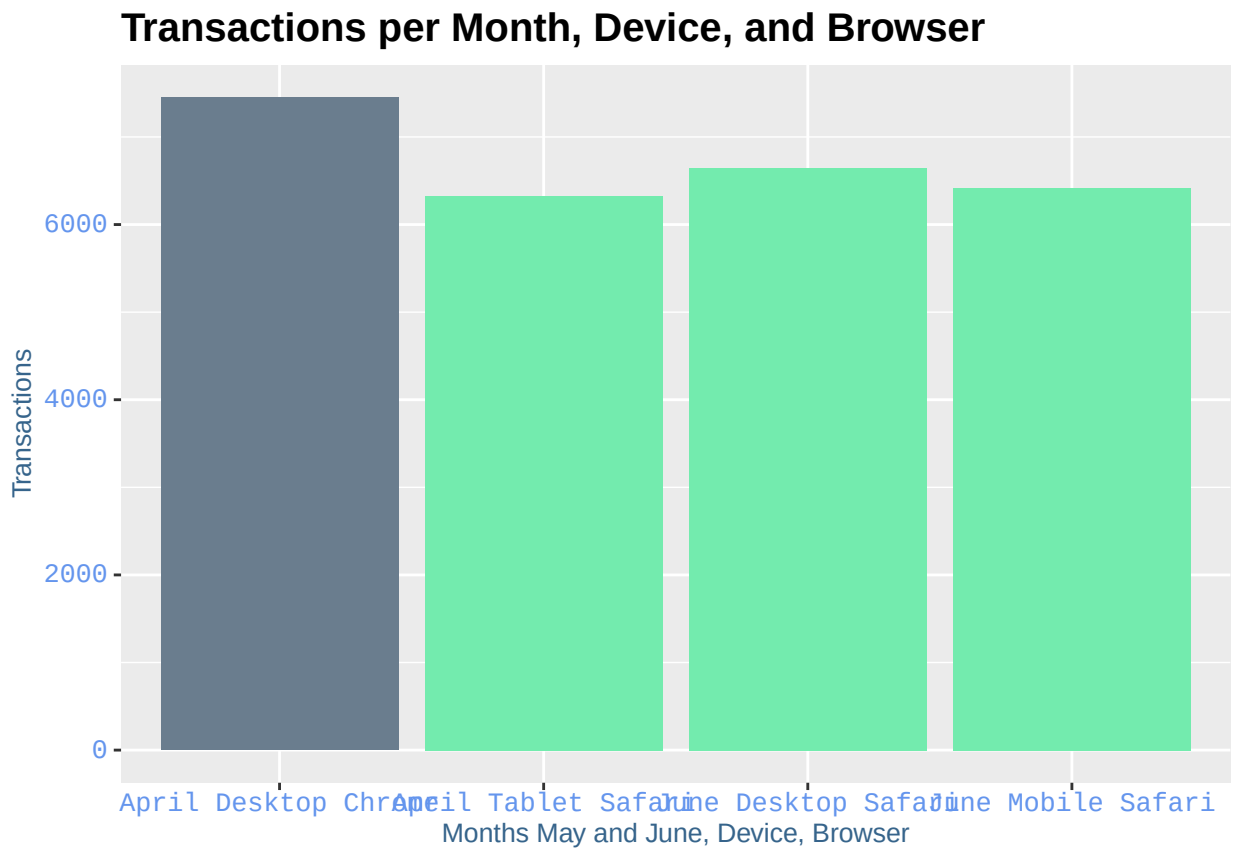
```
# Replace the month and device types with a better name.

df_month_device_browser_sub['Month_Device_Browser'][df_month_device_browser_sub['Month_Device_Browser']


df_month_device_browser_sub['Month_Device_Browser'][df_month_device_browser_sub['Month_Device_Browser']


df_month_device_browser_sub['Month_Device_Browser'][df_month_device_browser_sub['Month_Device_Browser']


df_month_device_browser_sub['Month_Device_Browser'][df_month_device_browser_sub['Month_Device_Browser']
```

Graph the final maximum aggregation.

```
fig_month_device_browser <- ggplot(df_month_device_browser_sub, aes(x=Month_Device_Browser, y=transacti
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+myTheme+
                 scale_fill_manual(values = c("#6A7D8E", "#73EBAE","#73EBAE", "#73EBAE")) +
                xlab("Months May and June, Device, Browser")+
                ylab("Transactions")+
                ggtitle("Transactions per Month, Device, and Browser ")+ guides(fill = FALSE)
```

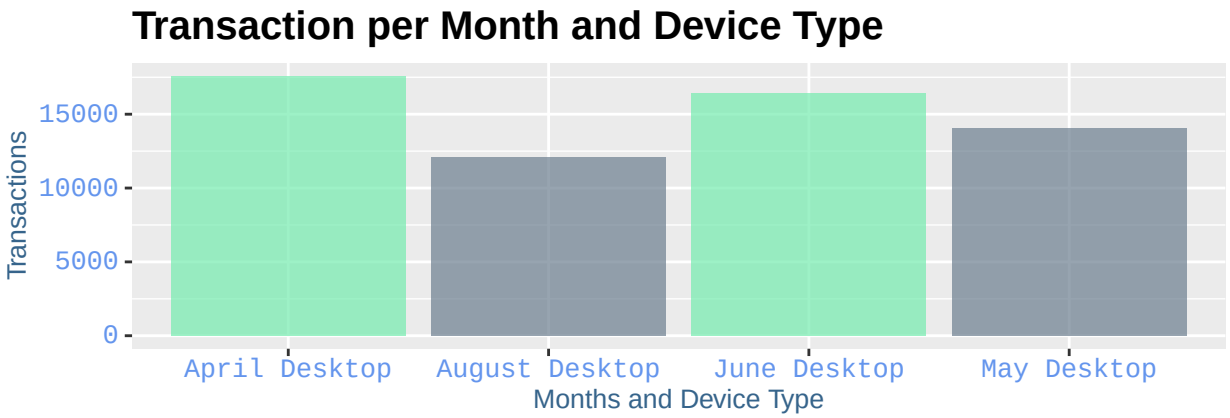
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

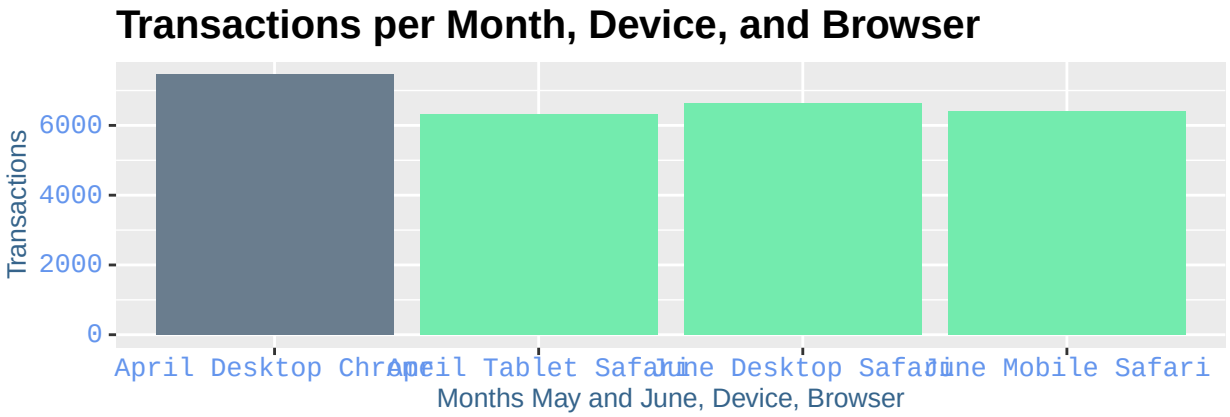
```
fig_month_device_browser
```



Graph the final figure.

```
grid.arrange(
fig_month_device1,
fig_month_device_browser)
```



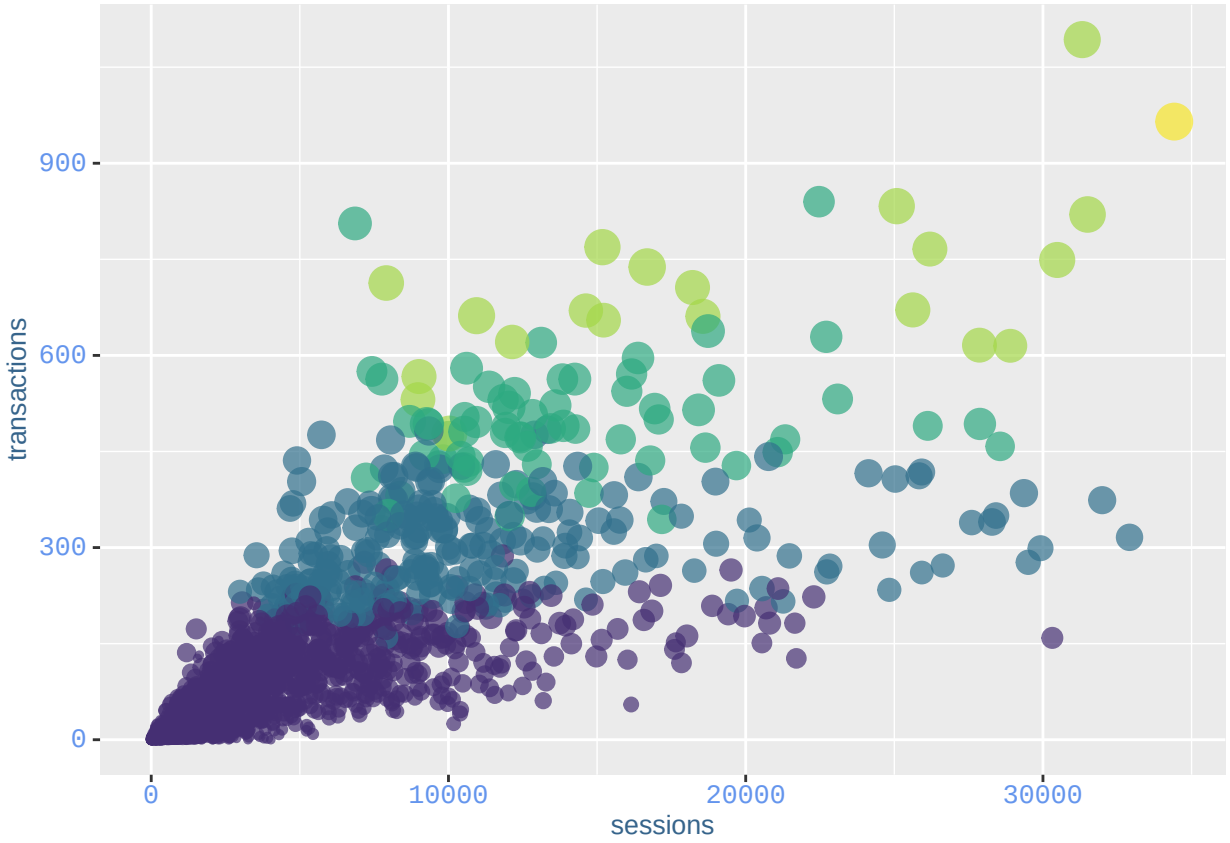


From the visualization it can be seen that the April desktops had the most transactions. Furthermore, from the second figure those trasactions come from Chrome and Safari.

Animations

Next animations will be constructed to try to understand the story of the data.

```
# Make a basic graph to produce an animation.

fig1 <- ggplot(session_counts_clean_final, aes(x=sessions, y=transactions, size=QTY, color=QTY)
              )+geom_point(show.legend = FALSE, alpha=0.7)+ scale_colour_viridis_b()+ myTheme

fig1
```



From the following graph the sessions variable has a linear relasionship with the tranasctions. Furthermore, the animation gives a more comprehesive story.

```
# Animate the previous graph over the time period.

# Finally, use one frame per second to see the animation correctly.

#anim1 <- fig1+transition_time(session_counts_clean_final$dim_date)+labs(title = "Date: {frame_time}, T

#animate(anim1, nframes= 100, fps = 1)
```

Looking at the animation shows that most of the sessions are less than 10000 and transactions are under 300. There are a moderate amount of sessions less than 20000 but greater than 15000 and the transactions are less than 600 but greater than 400. Finally, there are very few sessions greater than 25000 and the transactions are greater than 600.

Next create new variables to take into account the cumulative sum of the data.

```
# Calculate the cumulative sum of the transactions feature.

session_counts_clean_final <- session_counts_clean_final %>%
              group_by(dim_deviceCategory) %>%
              arrange(dim_date) %>%
              mutate(cumulative_trans = cumsum(transactions))
```

```
# Calculate the cumulative sum of the sessions variable.

session_counts_clean_final <- session_counts_clean_final %>%
```

```
              group_by(dim_deviceCategory) %>%
              arrange(dim_date) %>%
              mutate(cumulative_sessions = cumsum(sessions))
```

```
# Calculate the cumulative sum of the quantity.

session_counts_clean_final <- session_counts_clean_final %>%
              group_by(dim_deviceCategory) %>%
              arrange(dim_date) %>%
              mutate(cumulative_QTY = cumsum(QTY))
```

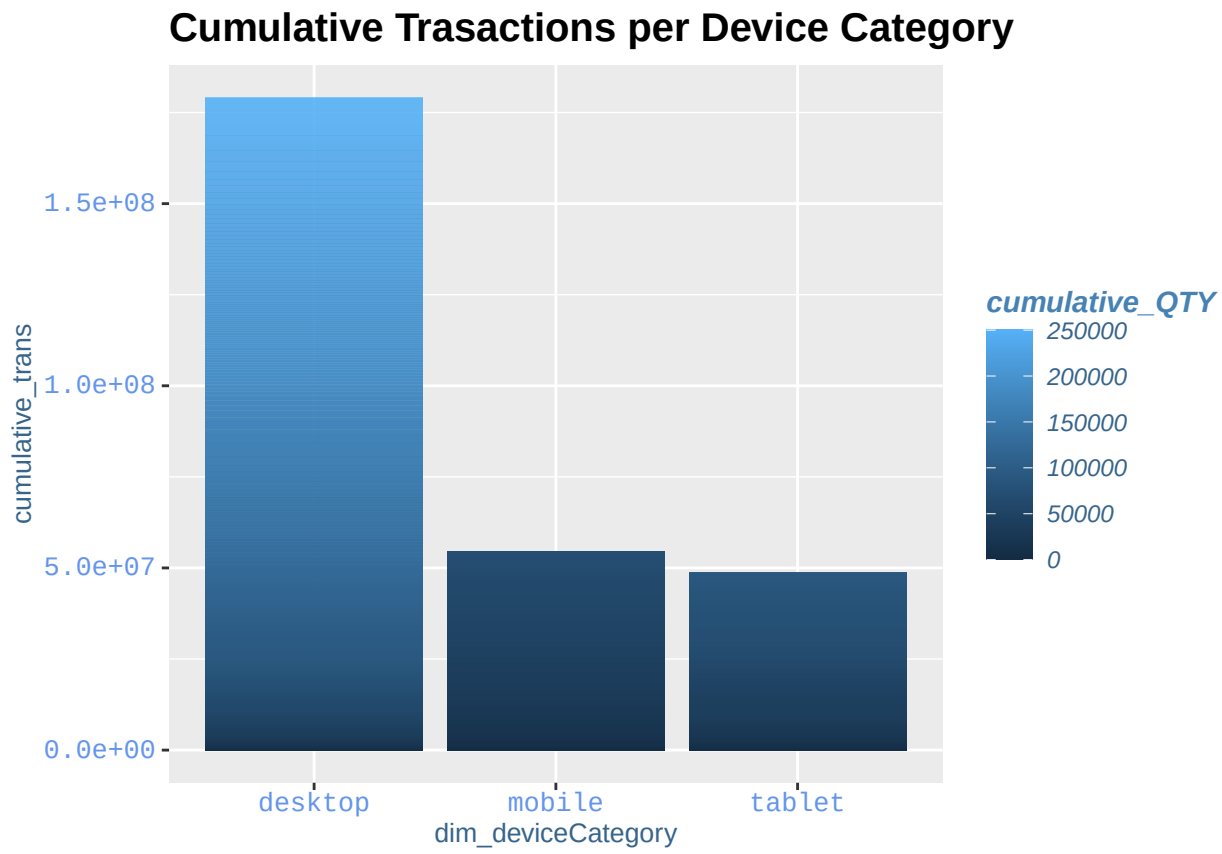
```
# Calculate the cumulative sum of the quantity.

session_counts_clean_final <- session_counts_clean_final %>%
              group_by(dim_deviceCategory) %>%
              arrange(dim_date) %>%
              mutate(cumulative_ECR = cumsum(ECR))
```

Make sure the correct columns where calculated.

```
View(session_counts_clean_final)
```

Next the visualizations will be animated to tell the story better.

```
# Plot the initial graph for animation.


fig2 <- ggplot(session_counts_clean_final, aes(x=dim_deviceCategory, y=cumulative_trans, fill = cumulati
               )+geom_col(show.legend = TRUE, alpha=0.7)+scale_colour_ordinal()+myTheme

fig2 + ggtitle("Cumulative Trasactions per Device Category")
```

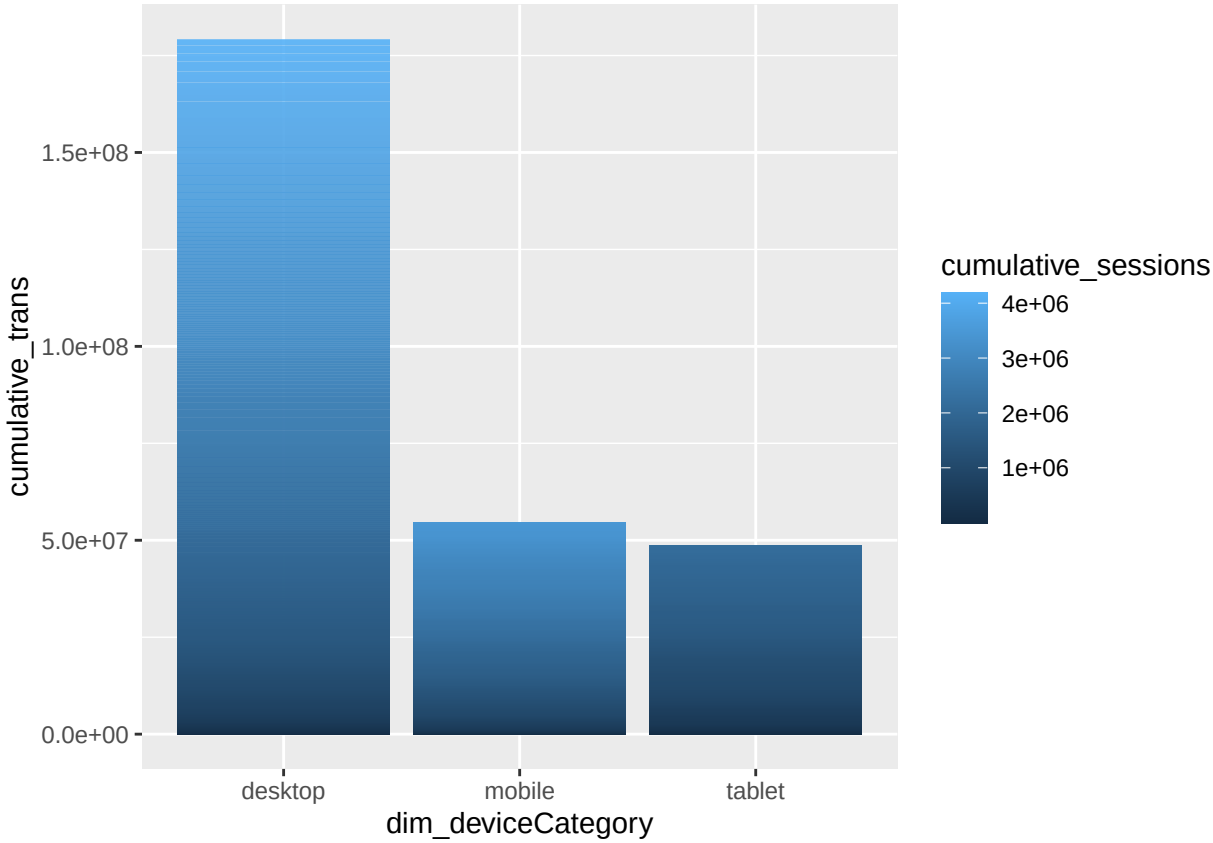


Next animate the column graph.

```
# Animate the previous graph over the time period.

# Finally, use seven frames per second to see the animation correctly.

#anim2 <- fig2+transition_time(session_counts_clean_final$dim_date)+labs(title = "Date: {frame_time}, C

#animate(anim2, nframes= 100, fps = 7)

# Save the animation.

#anim_save("Transaction_Animation_Deliverable.gif", animation = last_animation())
```

The previous animation tells the story that most transactions are on desktop type devices. And the quantity over time becomes much greater in the desktop category.

```
# Plot the initial graph.

fig3 <- ggplot(session_counts_clean_final, aes(x=dim_deviceCategory, y=cumulative_trans, fill = cumulati
              )+geom_col(show.legend = TRUE, alpha=0.7)+scale_colour_ordinal()

fig3
```

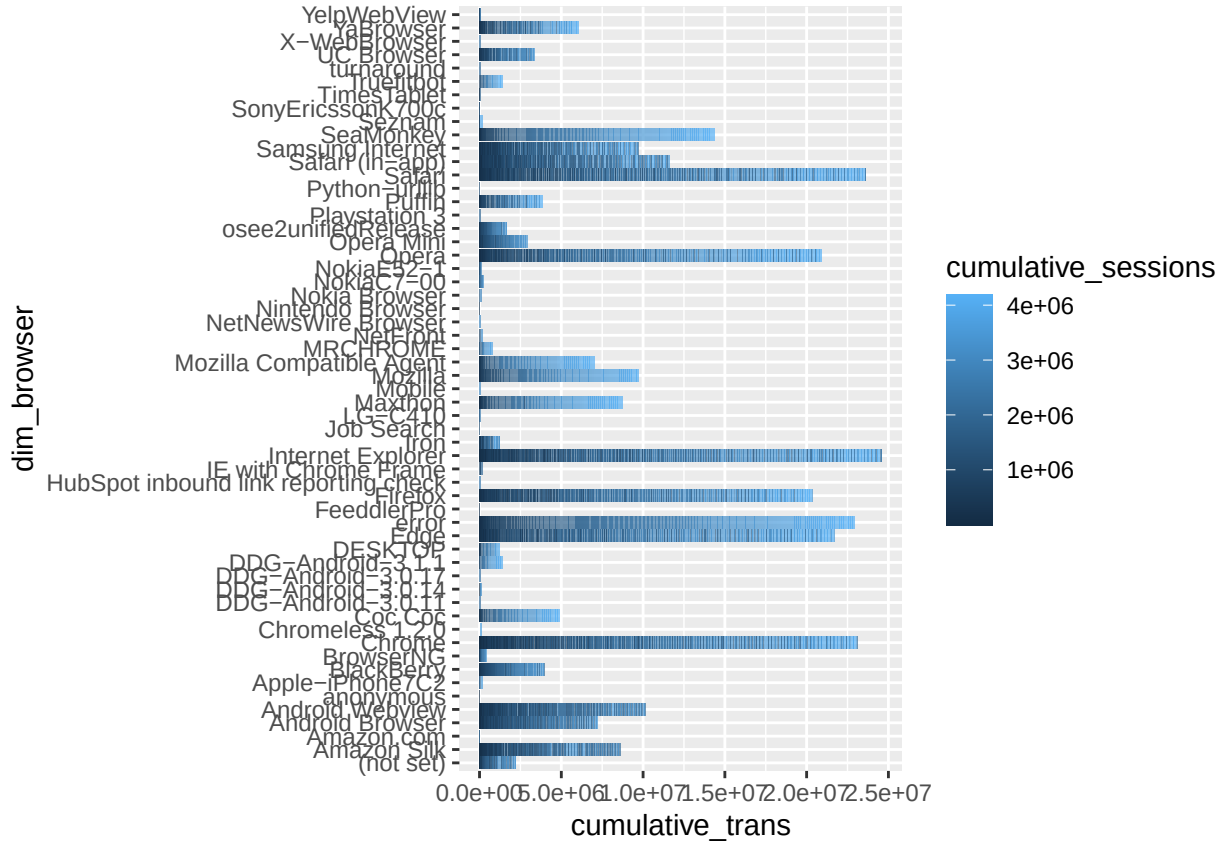



```
# Animate the previous graph over the time period.

# Finally, use seven frames per second to see the animation correctly.

#anim3 <- fig3+transition_time(session_counts_clean_final$dim_date)+labs(title = "Date: {frame_time}")

#animate(anim3, nframes= 100, fps = 7)
```

This animation gives the same story as the previous transaction animation.

Next look at the distribution of device type over transactions.

```
# Graph all the data to understand its distribution.


fig4 <- ggplot(session_counts_clean_final, aes(x=dim_browser, y=cumulative_trans, fill = cumulative_sess
              )+geom_col(show.legend = TRUE, alpha=0.7)+ scale_colour_viridis_b()+coord_flip()

fig4
```

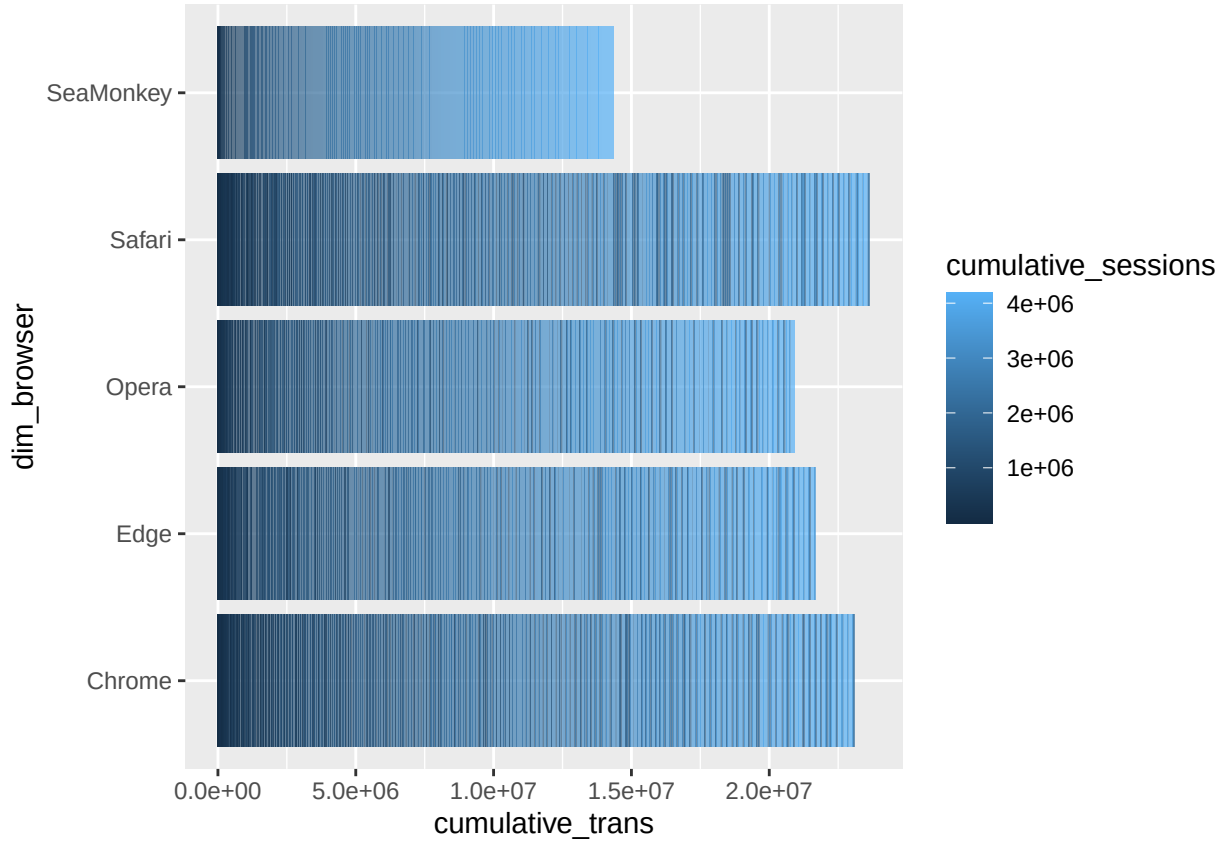


The previous graph is to difficult to discern which browsers have the greatest cumulative transactions. So a subset is used for the six greatest types.

```
# Use a Boolean to calculate a subset of the data.

df_browser <- session_counts_clean_final[session_counts_clean_final$dim_browser == "SeaMonkey" | session
```

Next graph the plot to animate.

```
# Plot a basic graph to animate and see if any interesting knowledge can obtained.

fig5 <- ggplot(df_browser, aes(x=dim_browser, y=cumulative_trans, fill = cumulative_sessions)
              )+geom_col(show.legend = TRUE, alpha=0.7)+ scale_colour_viridis_b()+coord_flip()

fig5
```




```
#anim5 <- fig5+transition_time(df_browser$dim_date)+labs(title = "Date: {frame_time}")

#animate(anim5, nframes= 100, fps = 7)
```

From the previous animation the browsers give that the top five browsers equally share sessions over time

Next compute the total amounts of all the numerical columns.

```
df_subset_totals <- session_counts_clean_final[-c(2,3,7:9,11:14)]
```

Aggregate over the browser.

```
totals <- df_subset_totals %>%
              group_by(dim_browser) %>%
              summarise_all(list(sum), na.rm=TRUE)
```

```
totals <- session_counts_clean_final %>%
              group_by(dim_browser) %>%
              mutate(total_transactions = sum(transactions))
```

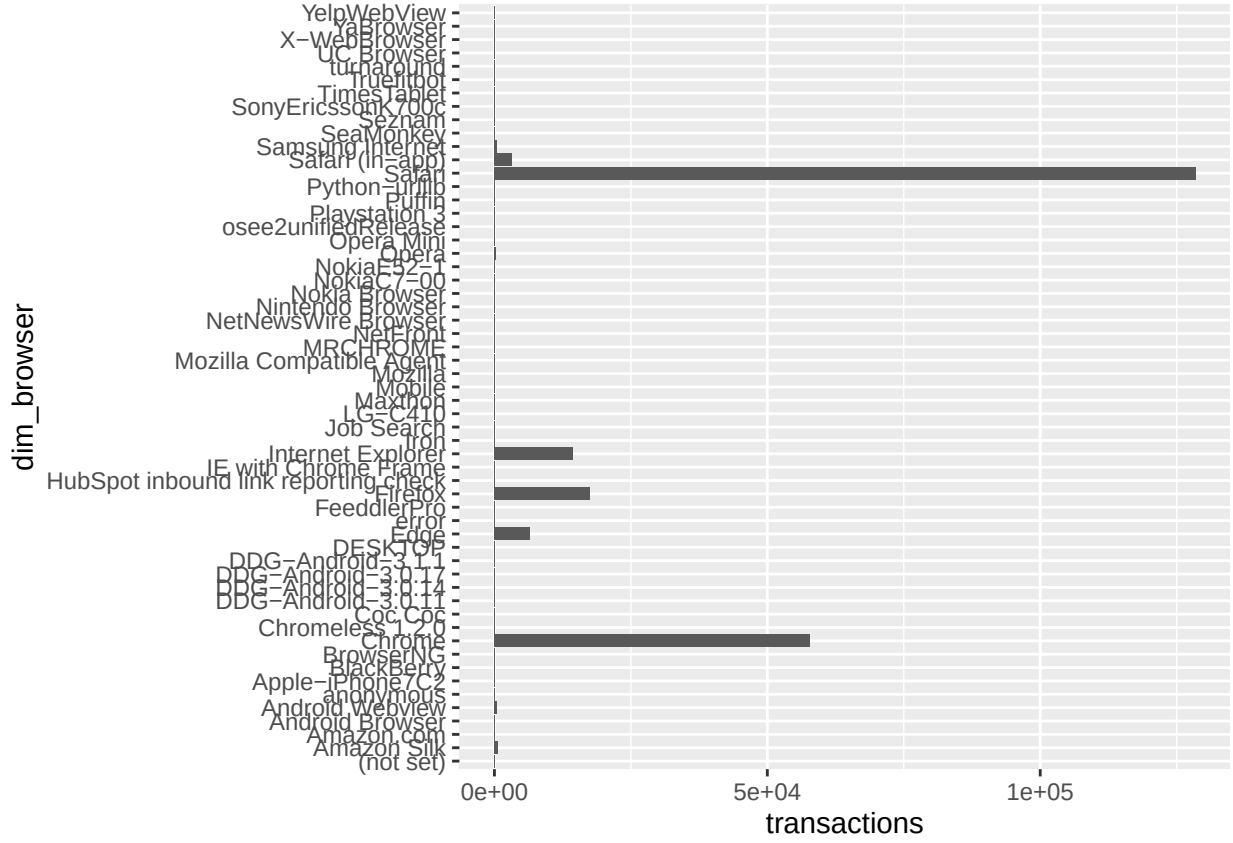
```
totals <- session_counts_clean_final %>%
              group_by(dim_browser) %>%
              mutate(total_sessions = sum(sessions))
```

```
totals <- session_counts_clean_final %>%
              group_by() %>%
              mutate(total_QTY = sum(QTY))
```

```
totals <- session_counts_clean_final %>%
              group_by() %>%
              mutate(total_ECR = sum(ECR))
```

```
View(totals)
```

```
totals_fig1 <- ggplot(totals, aes(x=dim_browser, y=transactions)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+coord_flip()

totals_fig1
```



Cacluate the largest transactions.

```
which(totals$transactions > 15000)
```

```
## integer(0)
```

```
print(totals[10,]$dim_browser)
```

```
## [1] "Opera"
```

```
print(totals[21,]$dim_browser)
```

```
## [1] "Android Webview"
```

```
print(totals[45,]$dim_browser)
```

```
## [1] "Edge"
```

```
df_browser1 <- totals[totals$dim_browser == "Safari" | totals$dim_browser == "Firefox" | totals$dim_brou
```

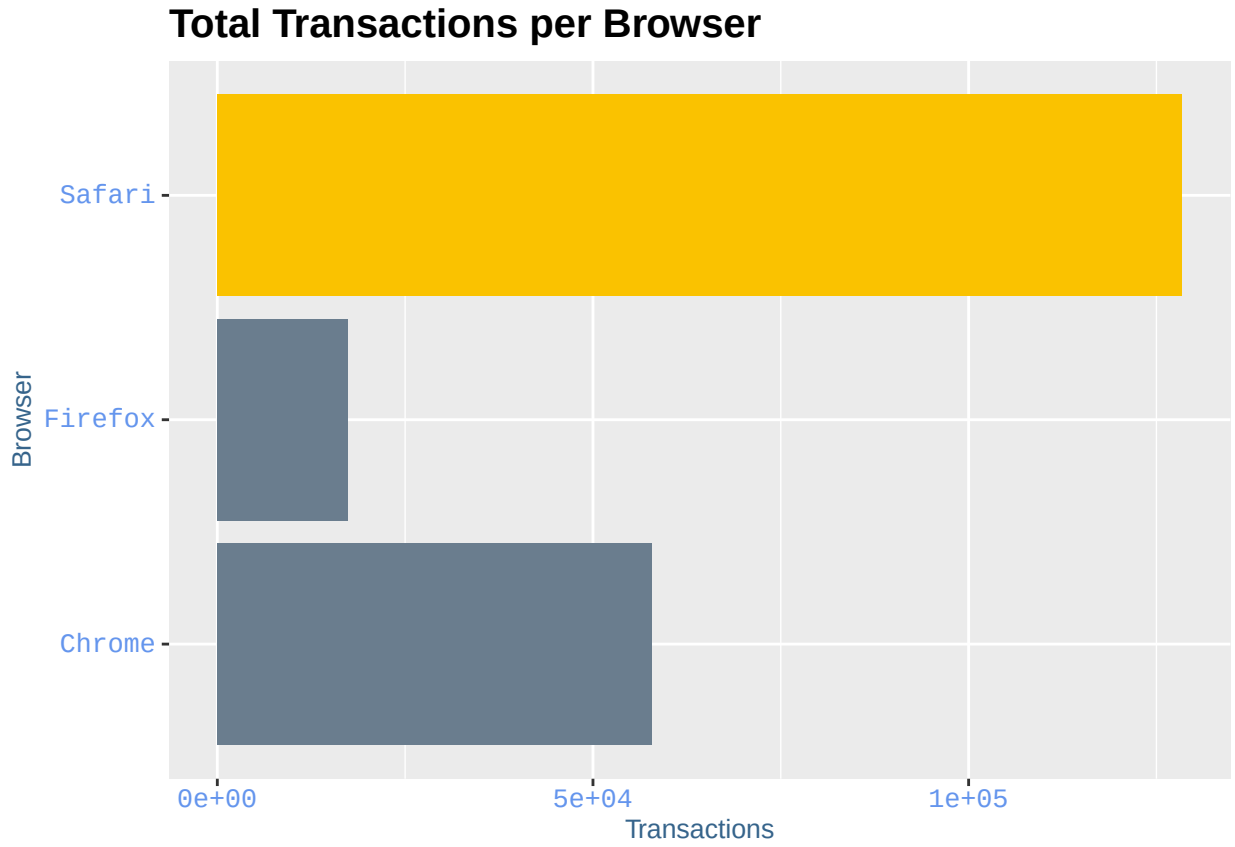
```
head(df_browser1)
```

```
## # A tibble: 6 x 15
##   dim_browser dim_d~1 dim_date   sessi~2 trans~3   QTY Month Day   Year      ECR
##   <chr>       <chr>   <date>       <dbl>   <dbl> <dbl> <chr> <chr> <chr>   <dbl>
## 1 Safari      tablet  2012-07-01    2928     127   221 7     1     12    0.0434
## 2 Chrome      tablet  2012-07-01     474       3    13 7     1     12    0.00633
## 3 Safari      tablet  2012-07-02    6624     261   494 7     2     12    0.0394
## 4 Chrome      desktop 2012-07-02    5953     120   272 7     2     12    0.0202
## 5 Firefox     mobile  2012-07-02      40       0     0 7     2     12    0
## 6 Safari      mobile  2012-07-03   14196     188   311 7     3     12    0.0132
## # ... with 5 more variables: cumulative_trans <dbl>, cumulative_sessions <dbl>,
## #   cumulative_QTY <dbl>, cumulative_ECR <dbl>, total_ECR <dbl>, and
## #   abbreviated variable names 1: dim_deviceCategory, 2: sessions,
## #   3: transactions
```

```
totals_fig1 <- ggplot(df_browser1, aes(x=dim_browser, y=transactions, fill = dim_browser)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+coord_flip()+
                myTheme+scale_fill_manual(values = c("#6A7D8E", "#6A7D8E","#FAC200")) +
                xlab("Browser")+
                ylab("Transactions")+
                ggtitle("Total Transactions per Browser")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
totals_fig1
```



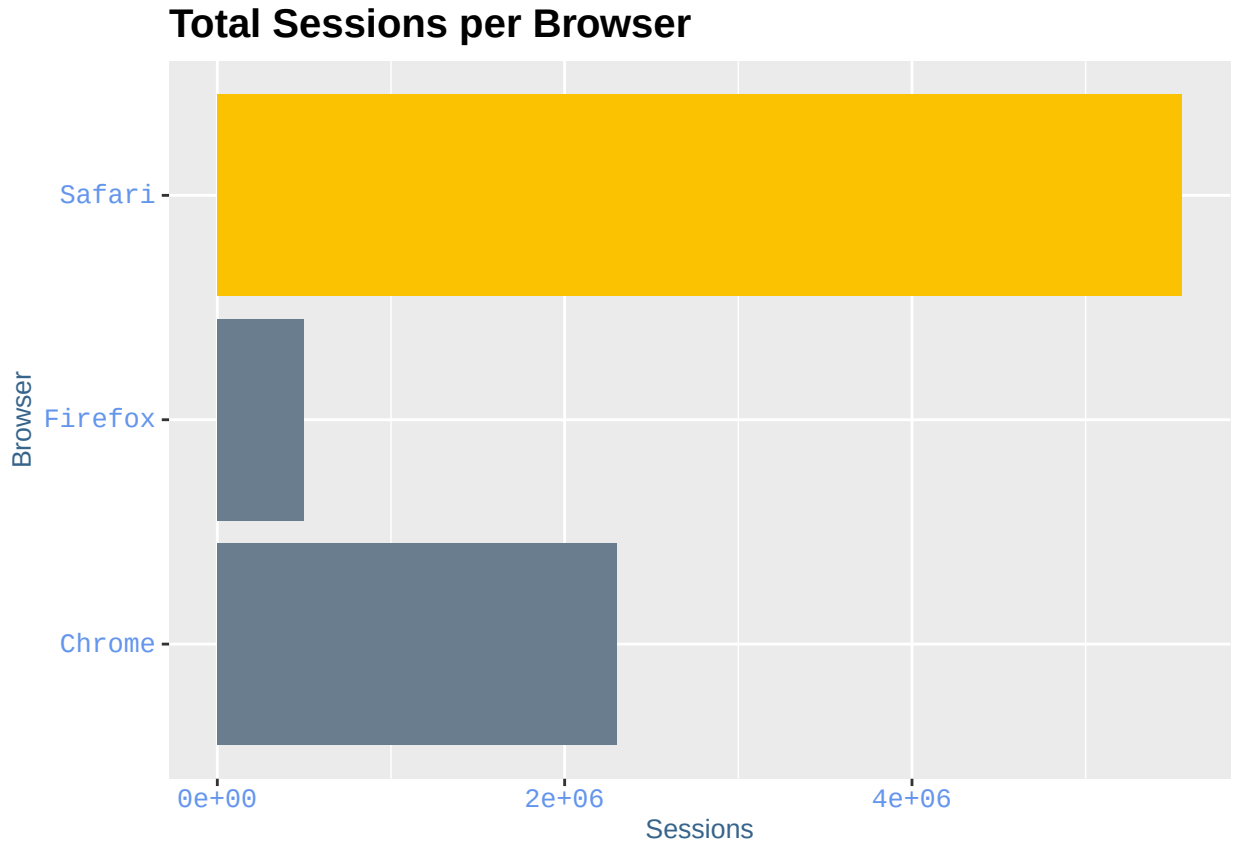

```
totals_fig2 <- ggplot(df_browser1, aes(x=dim_browser, y=sessions, fill=df_browser1$dim_browser)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+coord_flip()+coord_flip()
   myTheme+scale_fill_manual(values = c("#6A7D8E", "#6A7D8E","#FAC200")) +
                xlab("Browser")+
                ylab("Sessions")+
                ggtitle("Total Sessions per Browser")+ guides(fill = FALSE)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
totals_fig2
```

```
## Warning: Use of `df_browser1$dim_browser` is discouraged. Use `dim_browser`
## instead.
```

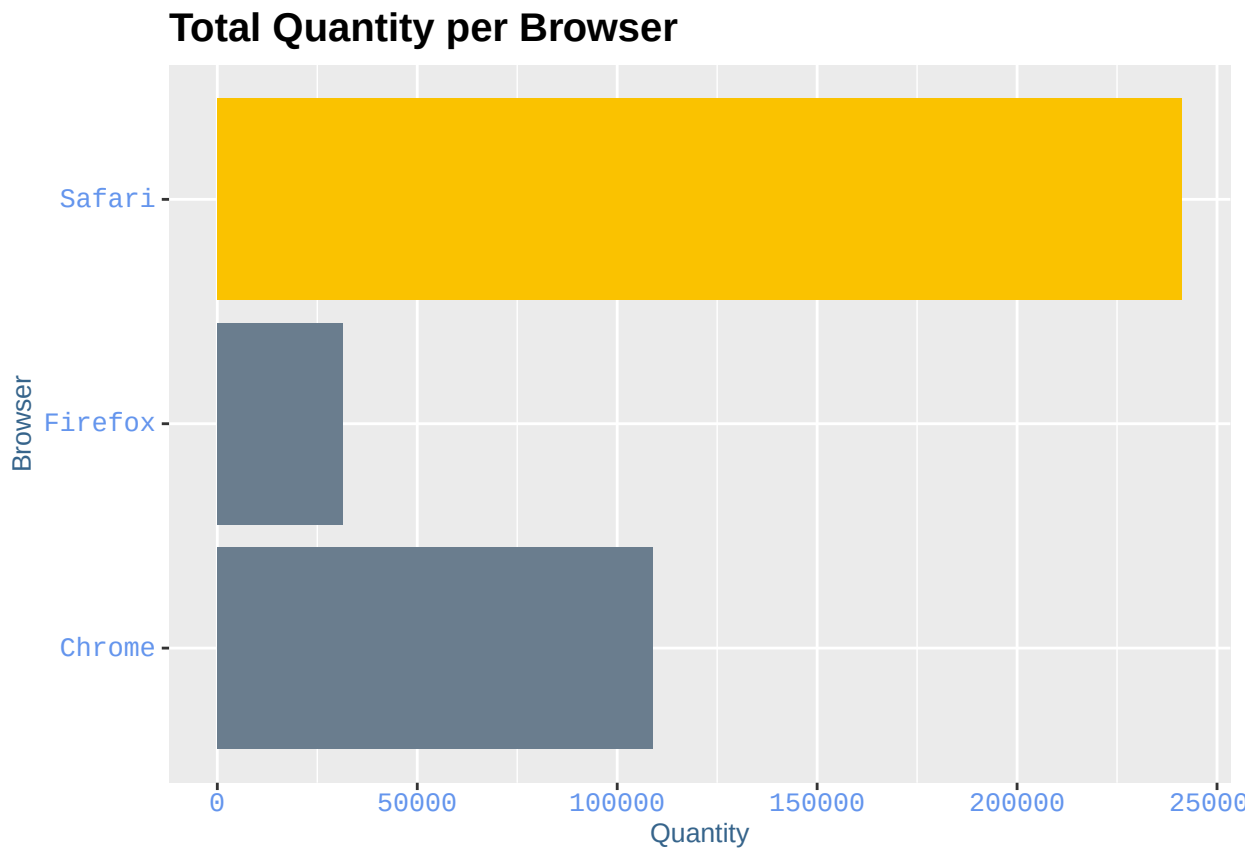



```
totals_fig3 <- ggplot(df_browser1, aes(x=dim_browser, y=QTY, fill=df_browser1$dim_browser)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+coord_flip()+
               myTheme+scale_fill_manual(values = c("#6A7D8E", "#6A7D8E","#FAC200")) +
                xlab("Browser")+
                ylab("Quantity")+
                ggtitle("Total Quantity per Browser")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
totals_fig3
```

```
## Warning: Use of `df_browser1$dim_browser` is discouraged. Use `dim_browser`
## instead.
```



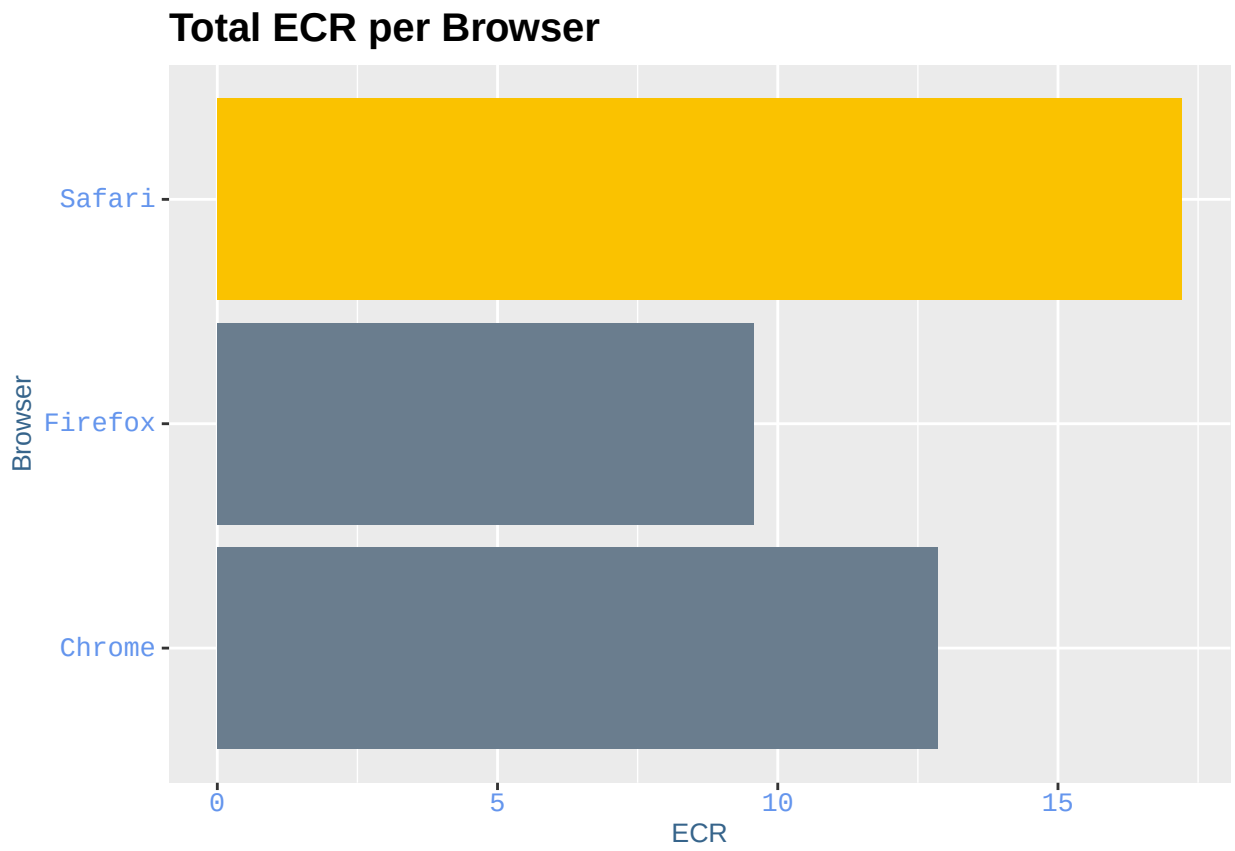

```
totals_fig4 <- ggplot(df_browser1, aes(x=dim_browser, y=ECR, fill = df_browser1$dim_browser)
               )+geom_col(show.legend = FALSE, alpha=1)+scale_colour_ordinal()+coord_flip()+
                 myTheme+scale_fill_manual(values = c("#6A7D8E", "#6A7D8E","#FAC200")) +
                xlab("Browser")+
                ylab("ECR")+
                ggtitle("Total ECR per Browser")+ guides(fill = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
totals_fig4
```

```
## Warning: Use of `df_browser1$dim_browser` is discouraged. Use `dim_browser`
## instead.
```

```
## Warning: Removed 25 rows containing missing values (position_stack).
```



Conclusion

In conclusion the analysis of the transactions yields that the majority of them are made on the desktop in Safari. Leading to the conclusion that people who use Macintosh desktop products are making transactions. These transactions are being made mostly during late fall and early summer: April - May. Moreover, Safari has the most sessions, Quantity, and ECR.