Assignment 4: Amran Ahmed

Amran Ahmed

04/03/2022

#Importing datasets

```
if (!require('ggplot2'))
{
  install.packages('ggplot2');
  library(ggplot2);
}
```

```
## Loading required package: ggplot2

## Warning in register(): Can't find generic 'scale_type' in package ggplot2 to
## register S3 method.
```

```
#importing my data sets
data2015 <- read.csv("2015_ontario_public_library_statistics_open_data_dec_2017rev.csv")

data2016 <- read.csv("2016 ontario_public_library_statistics_open_data_2016.csv")

data2017 <- read.csv("2017 ontario_public_library_statistics_open_data_july_2019_rev1.csv")
```

#Merged all files into one object

```
#organizing my column names
common_columns <- Reduce(intersect, list(colnames(data2015),colnames(data2016),colnames(data2017)))

#combining my files data
datacombined <- rbind(subset(data2015, select = common_columns), subset(data2016, select = common_colum
```

#Creating a new column that represents operating revenue per active card holder (total operating Revenue / number of active card holders).

```
#Removing  the commas
datacombined$B2.9..Total.Operating.Revenues <-  gsub(",", "", datacombined$B2.9..Total.Operating.Revenu

datacombined$A1.14..No..of.Active.Library.Cardholders <- gsub(",", "", datacombined$A1.14..No..of.Activ

#Checking the class of the columns
class(datacombined$A1.14..No..of.Active.Library.Cardholders)
```

```
## [1] "character"
```

```r
class(datacombined$B2.9..Total.Operating.Revenues)
```

```
## [1] "character"
```

```r
#Making NA into zero's
datacombined[is.na(datacombined)] <- 0

# Making the variables  numeric form
datacombined$A1.14..No..of.Active.Library.Cardholders <- as.numeric(datacombined$A1.14..No..of.Active.Li

datacombined$B2.9..Total.Operating.Revenues <-  as.numeric(datacombined$B2.9..Total.Operating.Revenues)

#Removing the zeros from the specified columns
datacombined <- subset(datacombined, datacombined$A1.14..No..of.Active.Library.Cardholders > 0 & datacon

#Dividing Operating Revenue by Active Card Holder
datacombined$B2.9..Total.Operating.Revenues / datacombined$A1.14..No..of.Active.Library.Cardholders
```

```
##   [1]  106.711771   80.325359  126.425102  107.902344  205.356113  390.569620
##   [7]  116.555893  206.356877   96.624116  201.863696   60.553506   91.902072
##  [13]  151.567500  120.850985   69.581386  302.703125  130.684037   48.255371
##  [19]  205.587570   88.434641   93.102474  170.855263   67.011848  149.464939
##  [25]  145.128591   80.890073  264.297450   45.161580  113.392346  105.590629
##  [31]  100.743738  150.912301  442.074469   74.795663   25.116438  209.057592
##  [37]  109.165008  141.558969   77.022573   87.677265  165.610394  206.853414
##  [43]  190.575955  138.507433   66.164443   56.319471   58.533333  208.913655
##  [49]   73.432878   38.337781   84.838095  139.061472  248.936090  149.922004
##  [55]  372.751220  107.945170  160.546154   29.830667  106.120000  146.583493
##  [61]   72.164487  242.569177   54.133159  117.712737  218.292514  166.266421
##  [67]  178.254192   94.132222   71.230469  139.651642   43.765333  131.111215
##  [73]  186.716667  118.690000  114.532086  125.894146   43.364384  232.547310
##  [79]   68.359786  214.441613  141.654289  190.952808   60.398788  113.891471
##  [85]   53.790301   66.024507   71.891044  163.539935  165.087435  159.114120
##  [91]  168.025806   30.294493   55.976809  121.879570  116.004558  325.496552
##  [97]  112.869367  151.764017   19.989474   62.441333  121.178670  113.222993
## [103]  152.007663   93.012220  197.268702   81.249531  118.888818  155.329964
## [109]  107.408293  147.301955  193.036535  200.387682  280.572838  116.365539
## [115]   77.300657   73.615176  129.243243  128.644120  224.900000  161.950769
## [121]  299.026247   79.290918  161.306644  144.363296  162.176887  151.593859
## [127]  101.012994  636.750000  141.286070  102.973529  135.933264  105.002165
## [133]  142.982210  111.894260  169.877575  132.281601   62.386788  127.629481
## [139]  107.454187   73.904298  189.556004   74.110340  229.047244  442.788732
## [145]  122.375424  139.162454  147.106562  146.491761   20.816279  295.536913
## [151]   95.578261 4257.600000   98.504057  269.068650  199.868293   91.864660
## [157]  160.507799   48.998602 1179.750000   62.255385  191.002304   93.796243
## [163]  115.550000  177.463335   70.635271  477.369048  180.375575   86.581167
## [169]   78.870183  124.845395   40.912360  262.714286   60.081798  125.776124
## [175]   93.354260  244.145329  295.983063   73.059490  109.930769   22.886947
## [181]  135.978166  120.881826   48.928561  155.039422 1322.322581  158.856919
## [187]   53.400114   63.539545   47.332021  203.698938  100.560897  125.539207
## [193]  153.055556  349.527778   99.350877  208.381356  215.163069  173.937219
## [199]  157.455276   76.551477  155.513249  129.043111  215.108025   86.157506
```

```
## [205]  111.798650  137.029165   83.757526  271.973373   96.293919  230.657192
## [211]   68.673167  249.577116   25.453875  133.143695   83.829708   55.412323
## [217]  138.738182  109.374663  154.602072  193.412121   77.367898  126.260054
## [223]  294.850000  187.962062  121.331210  169.295276  101.979915  184.816131
## [229]   87.557119  109.332440  171.580488  169.365714   95.352941  621.223301
## [235]  177.781726  181.870098   85.320038  117.683612 1081.437500  106.307306
## [241]  113.452632  225.987217  104.783251   56.549446 1010.100000  912.164420
## [247]  192.880700   93.512876   87.209448   99.183201  105.277567   86.707576
## [253]  108.796974   93.597409  239.621145  109.925806  101.503557  245.880658
## [259]   27.154462  151.568850  177.370803   49.259098  248.051338  287.072648
## [265]  100.034858  183.819643   32.808791  116.006873  232.264000  141.130268
## [271]  174.866946  196.249933   37.959049  119.340000  135.561240  212.089208
## [277]  146.284965  208.610315  187.005554   86.421166   58.151111   54.942104
## [283]   43.410322   72.697457  197.913194  113.671145  614.114286  127.108168
## [289]   53.076273   51.223958   89.750439  164.829244  168.901627  148.225271
## [295]  217.830913  151.823116  265.767750  147.704429  131.689416   83.639151
## [301]  138.853073   80.411833  196.549407  377.240000   83.630827   92.450459
## [307]   78.809917  199.661299   95.277778  162.045303   83.206790   81.787500
## [313]  135.124173   98.051471  216.810714  586.732143  115.932687  221.263345
## [319]  102.843046  214.842759   53.806763   89.299492   28.004000  182.279786
## [325]   50.039066  259.888889  135.764750   48.018204  205.514177  200.756598
## [331]  105.021980  123.686833  133.992000   73.392500  163.112805  116.187643
## [337]   72.284694  253.860947  112.222963  102.602520   99.083024   97.852610
## [343]  156.695189  444.642386   66.494212   25.197674  233.045990  129.779652
## [349]  134.689309  232.883929   84.479201  156.692297  207.333286  228.142764
## [355]  143.990816   65.431511   57.322581   83.619048  174.814061   73.038916
## [361]   36.277068   96.523906  140.580856  291.067114  185.946024  336.339623
## [367]  113.294710  166.687023  275.126946  318.360000   91.645935   80.718406
## [373]  296.895613   88.645374  118.084290  214.765321  166.889215  186.930057
## [379]   99.478099   69.686347  134.825337   44.859416  140.289000  129.373333
## [385]  115.343750  107.737101  161.141764   32.035616  466.326613   69.992063
## [391]  226.216560  146.275000  240.986677   64.435583  207.751066   42.083333
## [397]   65.534375   79.842620  182.844572  139.790085  146.664530  169.351304
## [403]  157.682699   59.957070  118.595745   62.581674  144.013514  113.585394
## [409]  158.340032    9.736538  100.373563  157.291416  116.810038  124.558719
## [415]   95.076810  174.857143  133.649235  124.276651  164.702032  114.195579
## [421]  103.920101  200.678554  196.891057  286.379410  148.510870   79.027455
## [427]   85.332100  138.363636  138.984428  522.661538  134.119186  295.813411
## [433]  101.749405  165.031206  203.385906  150.618721  143.849691   74.384312
## [439]  693.285714  115.541353   90.964441  116.740211  105.257225  142.395769
## [445]   93.044118  189.479848  127.301763   96.807504  122.716972  109.359667
## [451]   75.463169  193.047076   47.086935  230.057534  409.656357  134.686656
## [457]  177.882774  170.043134  149.843342   19.147727  184.000000  134.905952
## [463] 4157.600000  101.586399  137.619647  170.625227   91.048375  123.074873
## [469]   43.422650  864.325000   74.058115  192.220513   69.334776  152.556164
## [475]  178.159029   76.729255  441.102273  216.144653   96.663254   81.306673
## [481]  128.165674   38.924612  112.295082   62.069897  116.887646   81.251429
## [487]  224.124561  260.394962   67.699717   94.439216  251.762027  138.191731
## [493]  223.481640   65.423697  133.316589 1280.483871  164.152834   53.059600
## [499]   78.359744   45.550986  266.137626   96.047710   89.129401   66.666667
## [505]   84.030670  137.709677  192.936745  151.615144  146.735934   87.171642
## [511]  156.899200  137.925990  223.049634   75.607409  160.369184  173.914738
## [517]  174.171081  203.607407  105.892094  230.283806   63.203339  230.473744
## [523]   58.152985  145.438370   88.358576   53.030612  123.018622  109.118412
```

```
## [529]  158.923952  111.838542   78.125573  113.805699  270.912500  255.192972
## [535]  121.519681  162.088583  137.705728  195.461378   89.408884  114.046343
## [541]  159.588133  593.800000  246.333333  292.804878  185.198203  160.261521
## [547]   89.806686  120.307890  677.187500   95.847805   93.442105  221.231405
## [553]   81.214634   51.825437  365.100000 1052.010568  177.901402   91.813808
## [559]   93.590402  117.865823   72.752242   82.139062   44.808549   87.092185
## [565]  271.376147  109.959693   95.608604  212.653696   26.044614  188.750097
## [571]  213.636265   47.746483  231.975458  290.123388   91.553419  216.863338
## [577]   17.253304  286.306773  162.830769  150.695260  183.080172  172.289461
## [583]   32.841280  100.031847  137.789383  241.442399  154.069370  205.199539
## [589]   92.669331   72.349930   41.411483   57.905740  147.301176   72.879230
## [595]  231.926108  126.158545  553.566667  114.348463   58.555800   48.644886
## [601]   97.700018  177.556331  164.131100  180.623306  226.969834  129.835449
## [607]  260.837578  148.967373  147.203262  126.491379  146.008367   68.963396
## [613]  168.412752  287.240000   96.804461   71.125000  141.610032  210.544492
## [619]   74.755140  185.595974   88.664234  117.094851  137.382782  106.081897
## [625]  186.201014  583.576271  117.276252  189.196262   98.186605  300.909091
## [631]   72.199715   67.218750  136.510000  205.123245   57.668299  337.460000
## [637]  139.610759  110.832685  218.222236  173.294430   97.085516  131.096085
## [643]  202.929412  129.428571  168.715385   39.456300   71.229743  237.794872
## [649]  142.750590  118.807251   94.024484  113.643527  175.613411  327.074902
## [655]   69.075391   44.448485  233.395324  159.924072  125.740174  161.147771
## [661]   81.884254  128.724431  253.833910  155.329345  154.228523   71.166588
## [667]   64.573333  176.015571  105.982667   38.346498  128.811523  143.008837
## [673]  225.102881  170.014822  321.813636  140.858289  170.250000  292.956000
## [679]  518.360000   94.907474   85.590932  277.360745   46.475170  135.648046
## [685]  205.437873  170.519692  181.907965  111.657658   73.721190  137.387108
## [691]   46.432000  105.141647  130.960526  229.419355  233.955357  149.391443
## [697]   50.380822  492.458065  178.781928  237.101029  224.061633  253.416888
## [703]   77.129073  138.549180   41.062559   96.261000   73.706846  136.610962
## [709]  205.958214  149.478702  289.360596  134.408469   15.019654  105.355060
## [715]   68.587836  150.066667   53.264821  163.551067   12.163900   90.066895
## [721]  145.425400  136.043713  151.904851   77.446612  219.691698  132.553883
## [727]  148.826694  169.881215  115.317392  115.733650  177.771451  198.706762
## [733]  299.477464  155.066759   73.318410   80.340000  131.185185  148.334468
## [739]  568.815385  110.683924  361.048295  108.484406  167.611132  239.144231
## [745]  164.642202  143.730141  101.923039  656.200000  113.078947   95.976986
## [751]  111.794029  141.408719  181.098044   97.994879  187.317375  135.205521
## [757]   62.882829  125.607035  105.585182   58.332764  206.018664  129.484808
## [763]  216.541985  310.838150  120.503350  175.899056  199.228814  170.815126
## [769]   20.425000  221.856250  137.958549  363.366667  119.410596  146.377203
## [775]  179.718157  101.392118  122.427732   89.920536  987.900000   67.361732
## [781]  177.329923   73.641243  118.143460  170.039611   70.405863  691.450000
## [787]  212.252415   71.799893   88.810971  131.438623   39.938731  143.362832
## [793]   73.194576  110.531700   57.713889  258.484000  228.285714   79.150142
## [799]  169.627451  164.751532  148.472875  183.011408  205.635223  142.124551
## [805] 1045.138889  171.491278   52.292974   87.179679   41.982225  308.953055
## [811]  112.399616  170.846065   92.326667   70.681077  147.903226  268.932865
## [817]  145.972341  144.311111  104.326707  174.264676  141.177373  260.028008
## [823]  117.360363  145.224606  330.792866  159.497802  228.426667  134.503925
## [829]  222.083333   93.147802  321.666473   44.161616  144.823141  103.170672
## [835]   56.006699  148.565879  113.493169  173.913754   98.413793   79.340213
## [841]  151.140940  290.887500  296.216783  185.728163  182.244898  110.251532
## [847]  179.219979   89.788765  109.472652  144.981848  454.973333  407.453846
```

```
## [853]  464.993750  173.005482  294.556461   97.839046   92.950076  804.097561
## [859]  110.724331   91.041026  276.912910   84.336585   65.770387  385.120000
## [865] 1041.451910  215.318996   57.221992   77.358232  118.772529   79.319520
## [871]  119.400000   51.282986   78.241710  355.076923  112.742852   95.836355
## [877]  215.525000   30.582349  186.745799  264.530883  184.664495  234.570502
## [883]  300.387232  124.319209  193.668218   41.169604  265.450909  140.544000
## [889]  170.715209  226.761376  188.490511   31.184795  117.875000  107.906382
## [895]  236.494644  165.234729  213.374567   91.651453   70.506980   44.395122
## [901]   82.305238  192.078910   70.546171  294.758824  134.061499  559.666667
## [907]  149.595066   57.929555   44.165644  101.029023  186.662175  150.462099
## [913]  191.888845  235.069203  177.057416  283.810078  147.106479  149.008870
## [919]  148.262987  125.347220   61.601161  111.863551  893.780000  100.827612
## [925]   61.522876  102.530401  219.009448   84.821053  189.225933
```

```
#duplicating my dataframe
newdatacombined <- datacombined

#Creating new variable column for Operating Revenue per Active Card Holder
newdatacombined$newcolumn <- newdatacombined$B2.9..Total.Operating.Revenues / newdatacombined$A1.14..No
```

INTRODUCTION

#The objective of this project is to understand how Ontario Public Libraries can be more successful? In this analysis we will be looking at how to improve Ontario public libraries and the metric for this will mostly be revenue as a cue for success. The data set that will be used is between 2015-2017. In this analysis we will gather the data for three different insights which can be used to improve the success of Ontario public libraries. Success Criteria for this analysis includes a library being more successful when it has: Higher revenue, and more cardholders. The measured data set or columns that will be looked at include: revenue, and the number of active library cardholders. The categorical data set or columns that will be looked at include: cities or the town. The assumptions in the analysis include: Libraries with more visitors have higher revenues. Having a higher revenue makes a library more successful. To analyze this data, the analysis will gain insight from data visualized looking at tables, graphs, mean distribution, median distribution, and the quantile points of the data.

#INSIGHT 1: Creating a table of two columns that has the data for city names and newcolumn (total revenue per active cardholder). The insight we will gain from this data will indicate which cities have the highest operating revenue per cardholder.

```
 #Making newcolumn and City/Town into one object
twodatacombined<- dplyr::select (newdatacombined,newcolumn,A1.10.City.Town)

 #Putting the new object in ascending order by values in (newcolumn)
twodatacombined <- twodatacombined[order(twodatacombined$newcolumn),]

 #Looking at the last values in the object, this shows the highest values in (newcolumn) which shows th
tail(twodatacombined)
```

```
##     newcolumn A1.10.City.Town
## 305  1081.438  Seine River FN
## 206  1179.750          Gogama
## 623  1280.484  Garden Village
## 243  1322.323  Garden Village
```

```
## 579  4157.600           Britt
## 199  4257.600           Britt
```

#INSIGHT 2: looking at the average and overall distribution, with plot of revenue vs cardholders. Seeing higher total revenue with higher total cardholder. Libraries with more cardholders have higher total operating revenue numbers and so the insight gained here is that we need to invest more in understanding how these libraries are able to attract more cardholders.

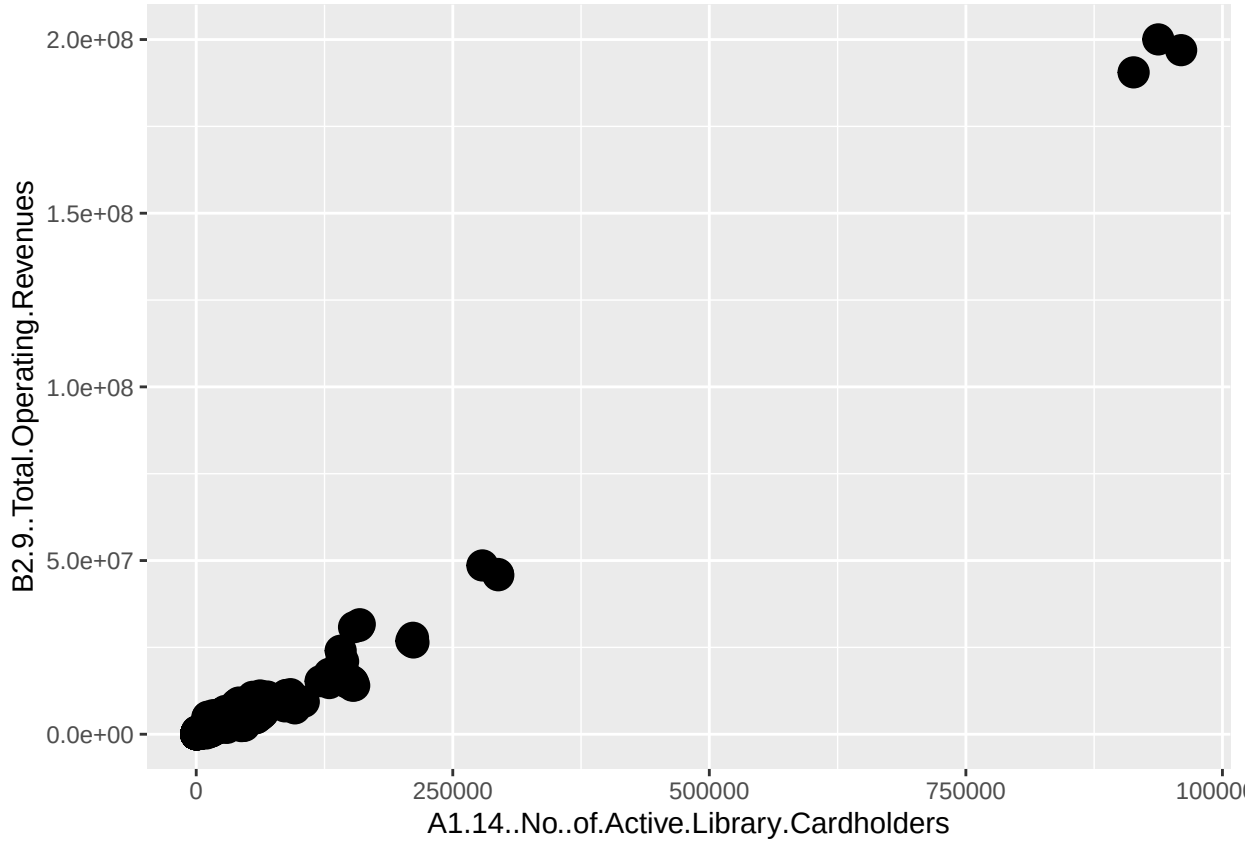
```
#average for newcolumn, total revenue per cardholder
mean(newdatacombined$newcolumn, na.rm = TRUE)
```

```
## [1] 172.1813
```

```
median(newdatacombined$newcolumn, na.rm = TRUE)
```

```
## [1] 134.906
```

```
 #plot showing the distribution of operating revenue vs number of active cardholders
ggplot(data = newdatacombined, mapping = aes(x = A1.14..No..of.Active.Library.Cardholders, y = B2.9..To
```



#INSIGHT 3: Looking at the quantile points of the newcolumn to understand the distribution of the newcolumn values. This data will show us the quantile points which demonstrate the five values for the minimum, the 1st quantile, the median, then mean 3rd quantile, and the maximum for every level of operating revenue per active cardholder.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)

 #this shows the quantile points
quantile(twodatacombined$newcolumn, probs=c(0, 0.25, 0.5, 0.75, 1))
```

```
##          0%         25%         50%         75%        100%
##    9.736538   91.041026  134.905952  187.317375 4257.600000
```

#Recommended Actions With the data gained from this report we've seen specific insights on total revenue per active cardholder. The government is advised to do further investigation on the top performing (highest revenue per cardholder) libraries logistial functioning information and their data to better understand their success and how to replicate that for other less performing libraries to improve their success..

The insights gained from looking at the median and average operating revenue per number cardholder is the knowledge of how well a typical library is expected to perform on average operating revenue per number and the median tells the exact middle value for which any other value is either below or above that number. This insight into the data allows the government to further visualize how the data is distributed and how high or low a library's operating revenue per number value is in comparison to other libraries.

The government should also perform more focused analysis on the libraries in the lowest and highest ends of the quantile points. This will provide the government with further information on exactly how to classify the lower end performing libraries, further research can look into the availability of resources in these areas including electronic resources which could be contributing to their decreased success.

#Fixing PDF knitting issue with this package install.packages("tinytex") tinytex::install_tinytex()

#Appendix of Lab 3 Source Code

```
if (!require('ggplot2'))
{
  install.packages('ggplot2');
  library(ggplot2);
}

#importing my data sets
data2015 <- read.csv("2015_ontario_public_library_statistics_open_data_dec_2017rev.csv")

data2016 <- read.csv("2016 ontario_public_library_statistics_open_data_2016.csv")

data2017 <- read.csv("2017 ontario_public_library_statistics_open_data_july_2019_rev1.csv")
```

```r
#organizing my column names
common_columns <- Reduce(intersect, list(colnames(data2015),colnames(data2016),colnames(data2017)))

#combining my files data
datacombined <- rbind(subset(data2015, select = common_columns), subset(data2016, select = common_colum




#Removing  the commas
datacombined$B2.9..Total.Operating.Revenues <-  gsub(",", "", datacombined$B2.9..Total.Operating.Revenu

datacombined$A1.14..No..of.Active.Library.Cardholders <- gsub(",", "", datacombined$A1.14..No..of.Activ

#Checking the class of the columns
class(datacombined$A1.14..No..of.Active.Library.Cardholders)

class(datacombined$B2.9..Total.Operating.Revenues)

#Making NA into zero's
datacombined[is.na(datacombined)] <- 0

# Making the variables  numeric form
datacombined$A1.14..No..of.Active.Library.Cardholders <- as.numeric(datacombined$A1.14..No..of.Active.Li

datacombined$B2.9..Total.Operating.Revenues <-  as.numeric(datacombined$B2.9..Total.Operating.Revenues)

class(datacombined$A1.14..No..of.Active.Library.Cardholders)

class(datacombined$B2.9..Total.Operating.Revenues)

#Removing the zeros from the specified columns
datacombined <- subset(datacombined, datacombined$A1.14..No..of.Active.Library.Cardholders > 0 & datacon

#Dividing Operating Revenue by Active Card Holder
datacombined$B2.9..Total.Operating.Revenues / datacombined$A1.14..No..of.Active.Library.Cardholders

#duplicating my dataframe
newdatacombined <- datacombined

#Creating new variable column for Operating Revenue per Active Card Holder
newdatacombined$newcolumn <- newdatacombined$B2.9..Total.Operating.Revenues / newdatacombined$A1.14..No
```