Assignment_1 - chetan_kandula

2025-03-12

Loading data set and libraries

```
# Load required dataset library
library(datasets)

# Set CRAN repository explicitly before installing packages
options(repos = c(CRAN = "https://cloud.r-project.org/"))

# Install required packages
install.packages("mice", dependencies = TRUE)
```

```
## Installing package into 'C:/Users/Asus/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'mice' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'mice'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\Asus\AppData\Local\R\win-library\4.4\00LOCK\mice\libs\x64\mice.dll to
## C:\Users\Asus\AppData\Local\R\win-library\4.4\mice\libs\x64\mice.dll:
## Permission denied
```

```
## Warning: restored 'mice'
```

```
##
## The downloaded binary packages are in
##  C:\Users\Asus\AppData\Local\Temp\RtmpaMCKYc\downloaded_packages
```

```
install.packages("tinytex", dependencies = TRUE)
```

```
## Installing package into 'C:/Users/Asus/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'tinytex' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Asus\AppData\Local\Temp\RtmpaMCKYc\downloaded_packages
```

```r
# Load the 'mice' library
library(mice)
```

```
## Warning: package 'mice' was built under R version 4.4.3
```

```
##
## Attaching package: 'mice'
```

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

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
# Read the dataset
dataset_cars <- read.csv("C:\\Users\\Asus\\Downloads\\cars_data_10K.csv")

# Check if TinyTeX is installed; install only if missing
if (!tinytex::is_tinytex()) {
  tinytex::install_tinytex(force = TRUE)
}

# Install ggplot2 if not installed
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2", dependencies = TRUE)
}

# Load ggplot2 library
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

Exploring and knowing the dataset:

In this part I have explored and studied whole dataset and columns.

```r
summary(dataset_cars)
```

```
##      Make               Model                Year      Engine.Fuel.Type
##  Length:10000       Length:10000       Min.   :1990   Length:10000
##  Class :character   Class :character   1st Qu.:2007   Class :character
##  Mode  :character   Mode  :character   Median :2015   Mode  :character
##                                        Mean   :2010
##                                        3rd Qu.:2016
##                                        Max.   :2017
##
##    Engine.HP    Engine.Cylinders Transmission.Type  Driven_Wheels
##  Min.   :  55   Min.   : 0.000   Length:10000       Length:10000
##  1st Qu.: 170   1st Qu.: 4.000   Class :character   Class :character
```

```
##  Median : 227   Median : 6.000   Mode  :character   Mode  :character
##  Mean   : 249   Mean   : 5.632
##  3rd Qu.: 300   3rd Qu.: 6.000
##  Max.   :1001   Max.   :16.000
##  NA's   :62     NA's   :25
##  Number.of.Doors Market.Category    Vehicle.Size       Vehicle.Style
##  Min.   :2.000   Length:10000       Length:10000       Length:10000
##  1st Qu.:2.000   Class :character   Class :character   Class :character
##  Median :4.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :3.434
##  3rd Qu.:4.000
##  Max.   :4.000
##  NA's   :3
##   highway.MPG        city.mpg       Popularity        MSRP
##  Min.   : 12.00   Min.   :  7.0   Min.   :   2   Min.   :   2000
##  1st Qu.: 22.00   1st Qu.: 15.0   1st Qu.: 549   1st Qu.:  20960
##  Median : 25.00   Median : 18.0   Median :1385   Median :  29935
##  Mean   : 26.59   Mean   : 19.7   Mean   :1558   Mean   :  40341
##  3rd Qu.: 30.00   3rd Qu.: 22.0   3rd Qu.:2009   3rd Qu.:  42146
##  Max.   :354.00   Max.   :137.0   Max.   :5657   Max.   :1705769
##
```

```
head(dataset_cars)
```

```
##         Make                   Model Year         Engine.Fuel.Type Engine.HP
## 1  Chevrolet Black Diamond Avalanche 2013 flex-fuel (unleaded/E85)       320
## 2      Lexus                 RX 330 2005         regular unleaded       230
## 3     Suzuki                Sidekick 1996         regular unleaded       120
## 4 Land Rover             Range Rover 2016                   diesel       254
## 5   Cadillac               ATS Coupe 2015 flex-fuel (unleaded/E85)       321
## 6       Ford                  Fusion 2015         regular unleaded       175
##   Engine.Cylinders Transmission.Type     Driven_Wheels Number.of.Doors
## 1                8         AUTOMATIC  four wheel drive               4
## 2                6         AUTOMATIC   all wheel drive               4
## 3                4            MANUAL  four wheel drive               4
## 4                6         AUTOMATIC  four wheel drive               4
## 5                6         AUTOMATIC  rear wheel drive               2
## 6                4         AUTOMATIC front wheel drive               4
##                    Market.Category Vehicle.Size   Vehicle.Style highway.MPG
## 1               Crossover,Flex Fuel        Large Crew Cab Pickup          21
## 2                  Crossover,Luxury      Midsize         4dr SUV          22
## 3                               N/A      Compact         4dr SUV          23
## 4                    Diesel,Luxury        Large         4dr SUV          29
## 5 Flex Fuel,Luxury,High-Performance      Compact           Coupe          28
## 6                               N/A      Midsize           Sedan          34
##   city.mpg Popularity  MSRP
## 1       15       1385 47885
## 2       16        454 37425
## 3       20        481  2000
## 4       22        258 93450
## 5       18       1624 48165
## 6       22       5657 22500
```

```r
colnames(dataset_cars)
```

```
##  [1] "Make"               "Model"              "Year"
##  [4] "Engine.Fuel.Type"   "Engine.HP"          "Engine.Cylinders"
##  [7] "Transmission.Type" "Driven_Wheels"      "Number.of.Doors"
## [10] "Market.Category"    "Vehicle.Size"       "Vehicle.Style"
## [13] "highway.MPG"        "city.mpg"           "Popularity"
## [16] "MSRP"
```

```r
dim(dataset_cars)
```

```
## [1] 10000    16
```

Cleaning Dataset:

1.To check the data 2.Identifying Null[NA] values 3.Identifying special characters and replacing/removing from the column to convert the data in appropriate data 4.Handling Outliers.

Cleaning each and every column step-by-step:

column-1 : Make

```r
summary(dataset_cars$Make)
```

```
##    Length     Class      Mode
##     10000 character character
```

```r
#finding how many times a word got repeated and finding frequency of the words
frequency(dataset_cars$Make)
```

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

```r
#converting all the column names to lowercase:
dataset_cars$Make <- tolower(dataset_cars$Make)
#checking null values[NA]:
sum(is.na(dataset_cars$Make))
```

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

```r
#removing unwanted spaces from this column:
dataset_cars$Make <- gsub(" ","",dataset_cars$Make)

#now removing all the special characters from the columns:
#In this columns we are having special character {'-'} -> removing '-' and replacing:
dataset_cars$Make <-gsub("-","",dataset_cars$Make)

#once again checking for other special characters in data:
#grepl("[^a-zA-Z ]",dataset_cars$Make)

table(dataset_cars$Make)
```

```
##
##        acura    alfaromeo  astonmartin         audi      bentley          bmw
##          210            4           72          269           63          286
##      bugatti        buick     cadillac    chevrolet     chrysler        dodge
##            1          167          323          937          154          533
##      ferrari         fiat         ford      genesis          gmc        honda
##           58           51          741            3          439          382
##       hummer      hyundai     infiniti          kia  lamborghini    landrover
##           16          245          264          196           45          115
##        lexus      lincoln        lotus     maserati      maybach        mazda
##          176          146           27           47           14          370
##      mclaren mercedesbenz   mitsubishi       nissan   oldsmobile     plymouth
##            4          289          179          471          127           69
##      pontiac      porsche   rollsroyce         saab        scion       spyker
##          152          113           28           87           50            3
##       subaru       suzuki        tesla       toyota   volkswagen        volvo
##          210          305           14          645          665          235
```

SUMMARY OF COLUMN-1:MAKE

1.The above column(Make) is entirely cleaned and preprocessed. 2.I have also removed all the special charactes and unwanted spaces in the column. 3.chevrolet made more no.of cars = 937

Column-2:Model

```
#Summary of the column:
summary(dataset_cars$Model)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#checking null characters:
sum(is.na(dataset_cars$Model))
```

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

```
#frequency:
frequency(dataset_cars$Model)
```

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

```
#Converting to lowercase:
dataset_cars$Model <- tolower(dataset_cars$Model)

#so here we can see that there are special characters like ['-' , '/' , '.', "space"]:
#removing Special characters and converting the data into appropriate format:
dataset_cars$Model <- gsub(" ","",dataset_cars$Model)
dataset_cars$Model <- gsub("-","",dataset_cars$Model)
dataset_cars$Model <- gsub("/","",dataset_cars$Model)
dataset_cars$Model <- gsub("\\.","",dataset_cars$Model)
```

Summary of column-2: Model

1.The data is cleaned and preprocessed properly 2.In this column I found Special characters[-,/,.]. 3.So i removed all the special characters and unwanted spaces and made the data ready.

Column-3 : YEAR

```
summary(dataset_cars$Year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1990    2007    2015    2010    2016    2017
```

```
#validating data:
is.numeric(dataset_cars$Year)
```

```
## [1] TRUE
```

```
#checking null values:
sum(is.na(dataset_cars$year))
```

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

```
#checking for any special characters in YEAR column:
specialchars3 <- gsub("[0-9 ]","",dataset_cars$Year)
cat(specialchars3)
```

```
#outliers
IQR_0 <- 2016 - 2007
LB_0 <- 2007 - 1.5 * IQR_0
UB_0 <- 2016 + 1.5 * IQR_0
#finding outliers and represemting them using boxplot
outliers_0 <- dataset_cars$YEAR[dataset_cars$YEAR < LB_0 |
                            dataset_cars$YEAR > UB_0]
outliers_0
```

```
## NULL
```

Summary of column-3: Year 1.This is a numerical column 2.This represents in which year a particular car is being manufactured 3.There are no outliers

Column -4 : Engine.Fuel.Type

```
#Summary of the column:
summary(dataset_cars$Engine.Fuel.Type)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
table(dataset_cars$Engine.Fuel.Type)
```

```
##
##
##                                         3
##                                    diesel
##                                       127
##                                  electric
##                                        56
## flex-fuel (premium unleaded recommended/E85)
##                                        22
##    flex-fuel (premium unleaded required/E85)
##                                        48
##                      flex-fuel (unleaded/E85)
##                                       741
##             flex-fuel (unleaded/natural gas)
##                                         6
##                                 natural gas
##                                         1
##               premium unleaded (recommended)
##                                      1251
##                  premium unleaded (required)
##                                      1673
##                             regular unleaded
##                                      6072
```

```
#Converting to lowercase:
dataset_cars$Engine.Fuel.Type <- tolower(dataset_cars$Engine.Fuel.Type)
#Identifying special characters in Engine.Fuel.Type column:
options(max.print = 10000000)
#printing all the special characters
special_characters <- function(column){
   a <- gsub("[a-zA-Z0-9 ]", "", column)
  b <- unlist(strsplit(paste(a, collapse=""), ""))
  b <- unique(b[b != ""])
  cat(b, sep = "")
}

special_characters(dataset_cars$Engine.Fuel.Type)
```

```
## -(/)
```

```
#removing special characters:
dataset_cars$Engine.Fuel.Type <- gsub(" ","",dataset_cars$Engine.Fuel.Type)
dataset_cars$Engine.Fuel.Type <- gsub("-","",dataset_cars$Engine.Fuel.Type)
dataset_cars$Engine.Fuel.Type <- gsub("/","",dataset_cars$Engine.Fuel.Type)
dataset_cars$Engine.Fuel.Type <- gsub("\\(","",dataset_cars$Engine.Fuel.Type)
dataset_cars$Engine.Fuel.Type <- gsub("\\.","",dataset_cars$Engine.Fuel.Type)
dataset_cars$Engine.Fuel.Type <- gsub("\\)","",dataset_cars$Engine.Fuel.Type)

table(dataset_cars$Engine.Fuel.Type)
```

```
##
##                                                    diesel
##                                 3                     127
```

```
##                                electric flexfuelpremiumunleadedrecommendede85
##                                      56                                    22
##    flexfuelpremiumunleadedrequirede85                   flexfuelunleadede85
##                                      48                                   741
##            flexfuelunleadednaturalgas                             naturalgas
##                                       6                                     1
##            premiumunleadedrecommended                premiumunleadedrequired
##                                    1251                                  1673
##                         regularunleaded
##                                    6072
```

Summary od Column-4: 1.Cleaned and preprocessed the entire column 2.Found special characters in this column and they are [-,/,(,),.] 3.So I removed all the special characters and unwanted spaces 4.The data(column) is preprocessed and cleaned completely.

column-5: Engine.HP

```
summary(dataset_cars$Engine.HP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      55     170     227     249     300    1001      62
```

```
#validating data:
is.numeric(dataset_cars$Engine.HP)
```

```
## [1] TRUE
```

```
#checking null values:
sum(is.na(dataset_cars$Engine.HP))
```

```
## [1] 62
```

```
#Here we found Null values -> Using Mice library to fill null values
#creating a temporary column and filling that column with 1 because mice function needs 2 columns to im
dataset_cars$Engine.HP.2 <- 1
#imputing values using mice function
imputed.dataset.cars <- mice(dataset_cars[,c("Engine.HP","Engine.HP.2")],
                             m=18,method='pmm',seed=50)
```

```
##
##  iter imp variable
##   1   1  Engine.HP
##   1   2  Engine.HP
##   1   3  Engine.HP
##   1   4  Engine.HP
##   1   5  Engine.HP
##   1   6  Engine.HP
##   1   7  Engine.HP
##   1   8  Engine.HP
##   1   9  Engine.HP
##   1   10  Engine.HP
```

```
##   1   11  Engine.HP
##   1   12  Engine.HP
##   1   13  Engine.HP
##   1   14  Engine.HP
##   1   15  Engine.HP
##   1   16  Engine.HP
##   1   17  Engine.HP
##   1   18  Engine.HP
##   2   1  Engine.HP
##   2   2  Engine.HP
##   2   3  Engine.HP
##   2   4  Engine.HP
##   2   5  Engine.HP
##   2   6  Engine.HP
##   2   7  Engine.HP
##   2   8  Engine.HP
##   2   9  Engine.HP
##   2   10  Engine.HP
##   2   11  Engine.HP
##   2   12  Engine.HP
##   2   13  Engine.HP
##   2   14  Engine.HP
##   2   15  Engine.HP
##   2   16  Engine.HP
##   2   17  Engine.HP
##   2   18  Engine.HP
##   3   1  Engine.HP
##   3   2  Engine.HP
##   3   3  Engine.HP
##   3   4  Engine.HP
##   3   5  Engine.HP
##   3   6  Engine.HP
##   3   7  Engine.HP
##   3   8  Engine.HP
##   3   9  Engine.HP
##   3   10  Engine.HP
##   3   11  Engine.HP
##   3   12  Engine.HP
##   3   13  Engine.HP
##   3   14  Engine.HP
##   3   15  Engine.HP
##   3   16  Engine.HP
##   3   17  Engine.HP
##   3   18  Engine.HP
##   4   1  Engine.HP
##   4   2  Engine.HP
##   4   3  Engine.HP
##   4   4  Engine.HP
##   4   5  Engine.HP
##   4   6  Engine.HP
##   4   7  Engine.HP
##   4   8  Engine.HP
##   4   9  Engine.HP
##   4   10  Engine.HP
```

```
##   4   11  Engine.HP
##   4   12  Engine.HP
##   4   13  Engine.HP
##   4   14  Engine.HP
##   4   15  Engine.HP
##   4   16  Engine.HP
##   4   17  Engine.HP
##   4   18  Engine.HP
##   5   1  Engine.HP
##   5   2  Engine.HP
##   5   3  Engine.HP
##   5   4  Engine.HP
##   5   5  Engine.HP
##   5   6  Engine.HP
##   5   7  Engine.HP
##   5   8  Engine.HP
##   5   9  Engine.HP
##   5   10  Engine.HP
##   5   11  Engine.HP
##   5   12  Engine.HP
##   5   13  Engine.HP
##   5   14  Engine.HP
##   5   15  Engine.HP
##   5   16  Engine.HP
##   5   17  Engine.HP
##   5   18  Engine.HP

## Warning: Number of logged events: 1
```

```
#extracting one fully imputed data set
completed_data_1 <- complete(imputed.dataset.cars, 1)
#Values form completed data set are assigned back to Engine.Engine COLUMN like updating the original co
dataset_cars$Engine.HP <- completed_data_1$Engine.HP
#removing temporary column
dataset_cars$Engine.HP.2 <- NULL

colnames(dataset_cars)
```

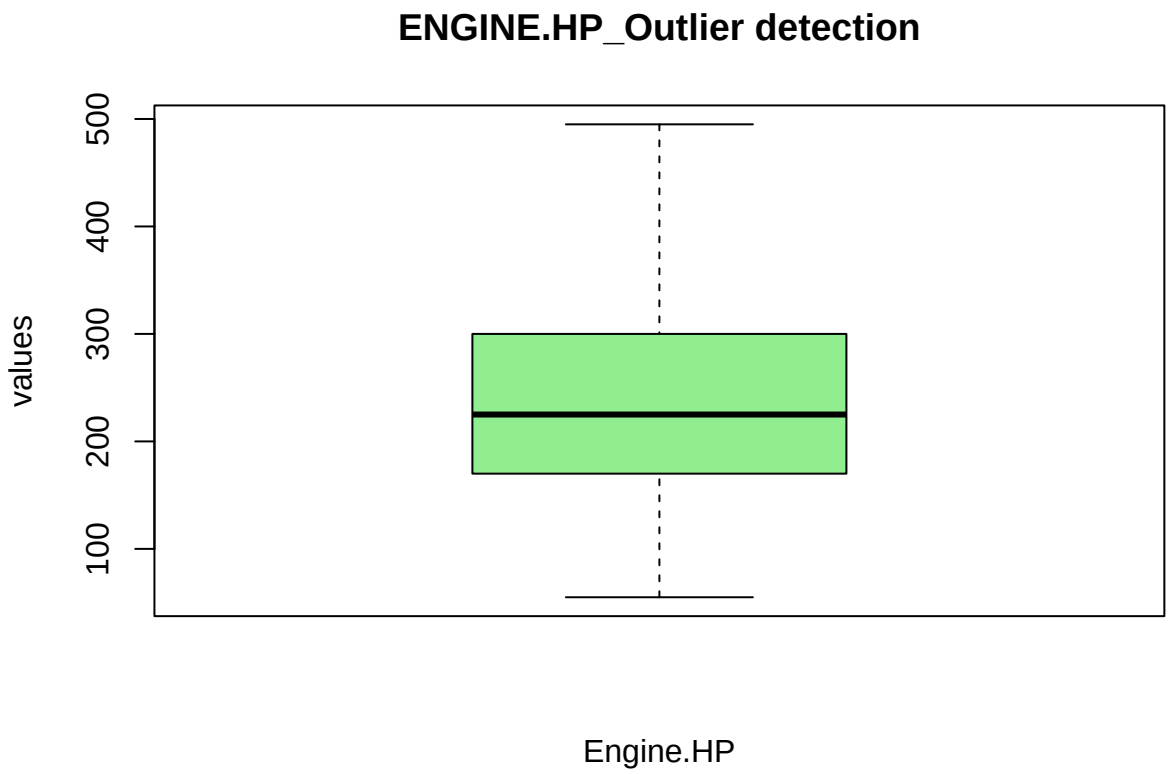
```
##  [1] "Make"              "Model"             "Year"
##  [4] "Engine.Fuel.Type"  "Engine.HP"         "Engine.Cylinders"
##  [7] "Transmission.Type" "Driven_Wheels"     "Number.of.Doors"
## [10] "Market.Category"   "Vehicle.Size"      "Vehicle.Style"
## [13] "highway.MPG"       "city.mpg"          "Popularity"
## [16] "MSRP"
```

```
#outliers

#treating outliers:
#calculating Interquantile range
IQR_1 <- 300 - 170
LB_1 <- 170 - 1.5 * IQR_1
UB_1 <- 300 + 1.5 * IQR_1
#finding outliers and represemting them using boxplot
```

```
outliers_1 <- dataset_cars$Engine.HP[dataset_cars$Engine.HP < LB_1 |
                                     dataset_cars$Engine.HP > UB_1]
outliers_1
```

```
##   [1]  650  605  631  543  560  510  700  525  645  536  515  521  510  645  645
##  [16]  560  583  582  567  536  543  583  560  536  720  505  577  521  640  577
##  [31]  621  560  621  505  580  556  560  515  523  556  650  510  503  645  563
##  [46]  570  621  720  567  510  510  550  510  540  510  510  570  510  540  540
##  [61]  550  573  600  650  510  510  500  570  563  510  560  510  552  650  577
##  [76]  620  532  621  662  510  510  563  650  605  563  550  500  540  523  662
##  [91]  560  510  523  510  550  556  640  600  605  582  560  650  550  700  580
## [106]  560  560  526  510  632  621  621  510  500  550  707  520  568  503  550
## [121]  526  510  641  565  563  510  525  540  515  597  700  552  510  645  518
## [136]  520  570  530  510  651  510  510  550  624  550  650  560  600  500  570
## [151]  556  731  550  505  572  510  552  567  550  600  626  617  621  503  510
## [166]  500  616  640  621  510  577  631  650  510  645  707  583  552  605  510
## [181]  563 1001  700  510  610  545  750  577  510  577  550  570  520  521  577
## [196]  620  621  600  660  567  560  550  567  610  525  560  560  650  560  526
## [211]  577  560  523  510  570  616  560  707  520  563  707  562  560  545  545
## [226]  650  583  563  510  720  568  518  626  580  510  525  631  570  650  550
## [241]  577  641  567  552  632  510  563  503  645  545  565  631  577  500  670
## [256]  550  700  600  500  631  621  611  552  510  640  550  500  611  500  563
## [271]  543  526  515  510  510  500  550  520  550  620  525  503  645  552  640
## [286]  567  562  750  520  510  568  570  616  521  577  570  550  520  550  562
## [301]  550  532  731  720  520  611  567  600  500  525  562  550  510  550  621
## [316]  611  510  662  523  557  540  624  604  553  540  645  540  650  577  552
## [331]  567  518  536  510  510  540  567  600  560  510  577  577  631  568  580
## [346]  565  510  604  510  520  563  626  510  620  552  707  510  543  650  510
## [361]  626  540  650  651  510  515  540  525  563  616  510  707  520  562  523
## [376]  520  621  532  640  563  650  500  560  600  568  624  521  621  661  650
## [391]  565  523  577  510  610  510  550  570  500  662  563  631  560  570  550
## [406]  645  543  520  621  530  645  535  562  597  620  523  505  580  632  570
## [421]  525  532  562  565  500  617  626  510  583  621
```

```
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound vaue
dataset_cars$Engine.HP[dataset_cars$Engine.HP < LB_1] <- LB_1
#HERE all values greater than upper bound value will be converted into upper bound vaue
dataset_cars$Engine.HP[dataset_cars$Engine.HP > UB_1] <- UB_1
#Hence outliers are treated and to represent that thing I used boxplot
boxplot(dataset_cars$Engine.HP,main="ENGINE.HP_Outlier detection",ylab="values",
        xlab="Engine.HP",col="lightgreen")
```



summary of column-5:

1.This is a complete numeric column 2.And there are outliers present in this column 3.I have treated outliers in this column and represented using boxplot 4.I used IQR method to treat outliers 5.There are 62 null values in this column 6.I have treated the null values with mice library and mean imputation technique 7.After this all these steps I ensured that there are no special characters and the column is properly cleaned.

Column-6 : Engine.Cylinders

```
summary(dataset_cars$Engine.Cylinders)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   4.000   6.000   5.632   6.000  16.000      25
```

```
#validating data:
is.numeric(dataset_cars$Engine.Cylinder)
```

```
## [1] TRUE
```

```
#checking null values:
sum(is.na(dataset_cars$Engine.Cylinder))
```

```
## [1] 25
```

```r
#we have null values:
#treating null values:
dataset_cars$Engine.Cylinders2 <- 1
imputed_dataset2 <- mice(dataset_cars[, c("Engine.Cylinders",
                      "Engine.Cylinders2")], m = 18, method = 'pmm', seed = 50)
```

```
##
##  iter imp variable
##   1   1  Engine.Cylinders
##   1   2  Engine.Cylinders
##   1   3  Engine.Cylinders
##   1   4  Engine.Cylinders
##   1   5  Engine.Cylinders
##   1   6  Engine.Cylinders
##   1   7  Engine.Cylinders
##   1   8  Engine.Cylinders
##   1   9  Engine.Cylinders
##   1   10  Engine.Cylinders
##   1   11  Engine.Cylinders
##   1   12  Engine.Cylinders
##   1   13  Engine.Cylinders
##   1   14  Engine.Cylinders
##   1   15  Engine.Cylinders
##   1   16  Engine.Cylinders
##   1   17  Engine.Cylinders
##   1   18  Engine.Cylinders
##   2   1  Engine.Cylinders
##   2   2  Engine.Cylinders
##   2   3  Engine.Cylinders
##   2   4  Engine.Cylinders
##   2   5  Engine.Cylinders
##   2   6  Engine.Cylinders
##   2   7  Engine.Cylinders
##   2   8  Engine.Cylinders
##   2   9  Engine.Cylinders
##   2   10  Engine.Cylinders
##   2   11  Engine.Cylinders
##   2   12  Engine.Cylinders
##   2   13  Engine.Cylinders
##   2   14  Engine.Cylinders
##   2   15  Engine.Cylinders
##   2   16  Engine.Cylinders
##   2   17  Engine.Cylinders
##   2   18  Engine.Cylinders
##   3   1  Engine.Cylinders
##   3   2  Engine.Cylinders
##   3   3  Engine.Cylinders
##   3   4  Engine.Cylinders
##   3   5  Engine.Cylinders
##   3   6  Engine.Cylinders
##   3   7  Engine.Cylinders
##   3   8  Engine.Cylinders
##   3   9  Engine.Cylinders
```

```
##   3   10  Engine.Cylinders
##   3   11  Engine.Cylinders
##   3   12  Engine.Cylinders
##   3   13  Engine.Cylinders
##   3   14  Engine.Cylinders
##   3   15  Engine.Cylinders
##   3   16  Engine.Cylinders
##   3   17  Engine.Cylinders
##   3   18  Engine.Cylinders
##   4   1  Engine.Cylinders
##   4   2  Engine.Cylinders
##   4   3  Engine.Cylinders
##   4   4  Engine.Cylinders
##   4   5  Engine.Cylinders
##   4   6  Engine.Cylinders
##   4   7  Engine.Cylinders
##   4   8  Engine.Cylinders
##   4   9  Engine.Cylinders
##   4   10  Engine.Cylinders
##   4   11  Engine.Cylinders
##   4   12  Engine.Cylinders
##   4   13  Engine.Cylinders
##   4   14  Engine.Cylinders
##   4   15  Engine.Cylinders
##   4   16  Engine.Cylinders
##   4   17  Engine.Cylinders
##   4   18  Engine.Cylinders
##   5   1  Engine.Cylinders
##   5   2  Engine.Cylinders
##   5   3  Engine.Cylinders
##   5   4  Engine.Cylinders
##   5   5  Engine.Cylinders
##   5   6  Engine.Cylinders
##   5   7  Engine.Cylinders
##   5   8  Engine.Cylinders
##   5   9  Engine.Cylinders
##   5   10  Engine.Cylinders
##   5   11  Engine.Cylinders
##   5   12  Engine.Cylinders
##   5   13  Engine.Cylinders
##   5   14  Engine.Cylinders
##   5   15  Engine.Cylinders
##   5   16  Engine.Cylinders
##   5   17  Engine.Cylinders
##   5   18  Engine.Cylinders

## Warning: Number of logged events: 1
```

```r
# Extracting one imputed data set
completed_data2 <- complete(imputed_dataset2, 1)

# Updating original column with imputed values
dataset_cars$Engine.Cylinders <- completed_data2$Engine.Cylinders
```

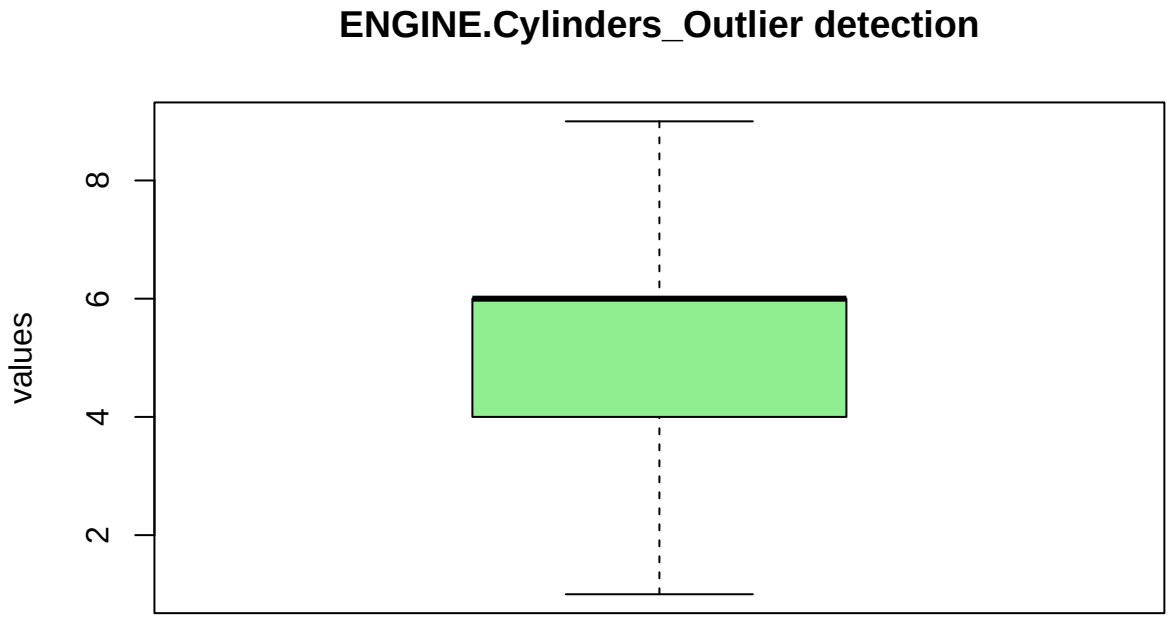
```r
# Removing the temporary column
dataset_cars$Engine.Cylinders2 <- NULL

colnames(dataset_cars)
```

```
##  [1] "Make"               "Model"              "Year"
##  [4] "Engine.Fuel.Type"   "Engine.HP"          "Engine.Cylinders"
##  [7] "Transmission.Type"  "Driven_Wheels"      "Number.of.Doors"
## [10] "Market.Category"    "Vehicle.Size"       "Vehicle.Style"
## [13] "highway.MPG"        "city.mpg"           "Popularity"
## [16] "MSRP"
```

```r
#outliers:
#treating outliers:
#calculating Interquantile range
IQR_2 <- 6 - 4
LB_2 <- 4 - 1.5 * IQR_2
UB_2 <- 6 + 1.5 * IQR_2
#finding outliers and represemting them using boxplot
outliers_2 <- dataset_cars$Engine.Cylinders[dataset_cars$Engine.Cylinders < LB_2
                                            | dataset_cars$Engine.Cylinders > UB_2]
outliers_2
```

```
##   [1] 10 12 12  0  0 10 12 10 10 12 10 10  0 12 12  0  0 12 12 12 10 12 10 12 10
##  [26] 12 12  0 12  0 10 12 12 12 12 12 12 12 12 12 10 12 10  0 12 12  0 12 12 10
##  [51] 12  0  0 12 12 12 12 12 12 12  0 10 10 10 12 10 10 12 12 10 12 10 12 10 12
##  [76] 10 12 12 12 12 12 12 12 10 12 10  0 12 12 12 10 12  0 12 12 12  0 10 10 12
## [101]  0 12 12 12 10  0 12  0 10 12 12  0  0 12 12  0 12 12  0 12 10  0 12 12 12
## [126] 16  0 12 12  0 10  0 12 12  0 10 12 12 12 12 12 10 10 12 10 10 10 12 10 12
## [151] 12 12 12  0 12 12 12 12 10 10 12 12  0  0 12 12  0 12  0  0 12 10 12 12  0
## [176] 10 12 12 12 12  0 12 12 10 12 10 12 12 12 12 12 12 12  0 12 10 10 12 12 12
## [201] 12 12 12 12 10 12 12  0 10 12  0 12  0 12 12 12 12 12 12 12 10 12 10 12 12
## [226]  0 12 12 12 12 10 12  0 12 12  0 12  0 12 12 12 12 12 12 12  0 12 12 12 12
## [251] 12 12 12 12 12 12 12  0 12  0 12 12 10 12 12  0 12  0 12 12 10  0 12 12 12
## [276] 12 12 12 12 10 12 10 12 12 12 12 10 10 12 12 12 10 12 12 12 12 12 12 12 10
## [301] 12 10 12 12 12 12 12 12
```

```r
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound vaue
dataset_cars$Engine.Cylinders[dataset_cars$Engine.Cylinders < LB_2] <- LB_2
#HERE all values greater than upper bound value will be converted into upper bound vaue
dataset_cars$Engine.Cylinders[dataset_cars$Engine.Cylinders > UB_2] <- UB_2
#representation of boxplot:
boxplot(dataset_cars$Engine.Cylinders,main="ENGINE.Cylinders_Outlier detection",
        ylab="values",xlab="Engine.Cylinders",col="lightgreen")
```



summary of column-5: 1.Numeric column 2.Total null values of 25 3.Treated null values with mice library and imputation techniques 4.Outliers are present in this column 5.Removed outliers and represented with boxplot

column-7: Transmission.Type

```
summary(dataset_cars$Transmission.Type)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
table(dataset_cars$Transmission.Type)
```

```
##
## AUTOMATED_MANUAL        AUTOMATIC     DIRECT_DRIVE           MANUAL
##              524             6924               58             2477
##          UNKNOWN
##               17
```

```
#Converting to lowercase:
dataset_cars$Transmission.Type <- tolower(dataset_cars$Transmission.Type)
#looking for nullvalues:
sum(is.na(dataset_cars$Transmission.Type))
```

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

```
#checking for special characters:
#Using manually created function
special_characters(dataset_cars$Transmission.Type)
```

```
## _
```

```
#removing special characters from the column
dataset_cars$Transmission.Type <- gsub(" ","",dataset_cars$Transmission.Type)
dataset_cars$Transmission.Type <- gsub("_","",dataset_cars$Transmission.Type)
```

summary of column-7: 1.chategorical column 2.Only one special character found and removed that special character 3.No null values

column-8: Driven_Wheels

```
summary(dataset_cars$Driven_Wheels)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
table(dataset_cars$Driven_Wheels)
```

```
##
##   all wheel drive  four wheel drive front wheel drive  rear wheel drive
##              1975              1200              3996              2829
```

```
#Converting to lowercase:
dataset_cars$Driven_Wheels <- tolower(dataset_cars$Driven_Wheels)
#looking for nullvalues:
sum(is.na(dataset_cars$Driven_Wheels))
```

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

```
#checking for special characters:
#Using manually created function
special_characters(dataset_cars$Driven_Wheels)
#removing space between words:
dataset_cars$Driven_Wheels <- gsub(" ","",dataset_cars$Driven_Wheels)
```

summary of column-8: 1.categorical column 2.No null values 3.No special characters just unwanted space between words

column-9:Number.of.Doors

```
summary(dataset_cars$Number.of.Doors)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   2.000   2.000   4.000   3.434   4.000   4.000       3
```

```r
#validating data:
is.numeric(dataset_cars$Number.of.Doors)
```

```
## [1] TRUE
```

```r
#checking null values:
sum(is.na(dataset_cars$Number.of.Doors))
```

```
## [1] 3
```

```r
#searching for special characters:
special_characters(dataset_cars$Number.of.Doors)
```

```
## NA
```

```r
#filling null values with mice library:
dataset_cars$Number.of.Doors2 <- 1
imputed_dataset3 <-mice(dataset_cars[, c("Number.of.Doors", "Number.of.Doors2")]
                        , m = 18, method = 'pmm', seed = 50)
```

```
##
##  iter imp variable
##   1   1  Number.of.Doors
##   1   2  Number.of.Doors
##   1   3  Number.of.Doors
##   1   4  Number.of.Doors
##   1   5  Number.of.Doors
##   1   6  Number.of.Doors
##   1   7  Number.of.Doors
##   1   8  Number.of.Doors
##   1   9  Number.of.Doors
##   1   10  Number.of.Doors
##   1   11  Number.of.Doors
##   1   12  Number.of.Doors
##   1   13  Number.of.Doors
##   1   14  Number.of.Doors
##   1   15  Number.of.Doors
##   1   16  Number.of.Doors
##   1   17  Number.of.Doors
##   1   18  Number.of.Doors
##   2   1  Number.of.Doors
##   2   2  Number.of.Doors
##   2   3  Number.of.Doors
##   2   4  Number.of.Doors
##   2   5  Number.of.Doors
##   2   6  Number.of.Doors
##   2   7  Number.of.Doors
##   2   8  Number.of.Doors
##   2   9  Number.of.Doors
##   2   10  Number.of.Doors
##   2   11  Number.of.Doors
```

```
##   2   12  Number.of.Doors
##   2   13  Number.of.Doors
##   2   14  Number.of.Doors
##   2   15  Number.of.Doors
##   2   16  Number.of.Doors
##   2   17  Number.of.Doors
##   2   18  Number.of.Doors
##   3   1  Number.of.Doors
##   3   2  Number.of.Doors
##   3   3  Number.of.Doors
##   3   4  Number.of.Doors
##   3   5  Number.of.Doors
##   3   6  Number.of.Doors
##   3   7  Number.of.Doors
##   3   8  Number.of.Doors
##   3   9  Number.of.Doors
##   3   10  Number.of.Doors
##   3   11  Number.of.Doors
##   3   12  Number.of.Doors
##   3   13  Number.of.Doors
##   3   14  Number.of.Doors
##   3   15  Number.of.Doors
##   3   16  Number.of.Doors
##   3   17  Number.of.Doors
##   3   18  Number.of.Doors
##   4   1  Number.of.Doors
##   4   2  Number.of.Doors
##   4   3  Number.of.Doors
##   4   4  Number.of.Doors
##   4   5  Number.of.Doors
##   4   6  Number.of.Doors
##   4   7  Number.of.Doors
##   4   8  Number.of.Doors
##   4   9  Number.of.Doors
##   4   10  Number.of.Doors
##   4   11  Number.of.Doors
##   4   12  Number.of.Doors
##   4   13  Number.of.Doors
##   4   14  Number.of.Doors
##   4   15  Number.of.Doors
##   4   16  Number.of.Doors
##   4   17  Number.of.Doors
##   4   18  Number.of.Doors
##   5   1  Number.of.Doors
##   5   2  Number.of.Doors
##   5   3  Number.of.Doors
##   5   4  Number.of.Doors
##   5   5  Number.of.Doors
##   5   6  Number.of.Doors
##   5   7  Number.of.Doors
##   5   8  Number.of.Doors
##   5   9  Number.of.Doors
##   5   10  Number.of.Doors
##   5   11  Number.of.Doors
```

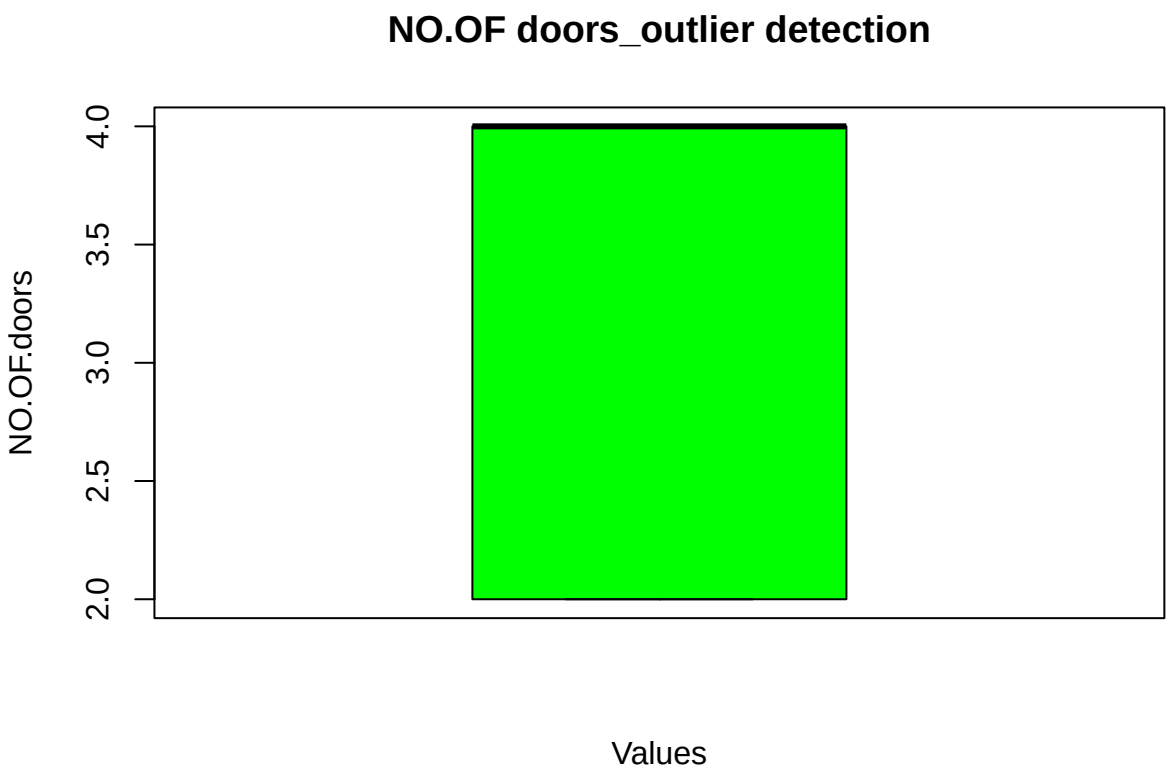
```
##   5   12  Number.of.Doors
##   5   13  Number.of.Doors
##   5   14  Number.of.Doors
##   5   15  Number.of.Doors
##   5   16  Number.of.Doors
##   5   17  Number.of.Doors
##   5   18  Number.of.Doors
```

```
## Warning: Number of logged events: 1
```

```
# Extracting one imputed data set
completed_data3 <- complete(imputed_dataset3, 1)

# Updating original column with imputed values
dataset_cars$Number.of.Doors <- completed_data3$Number.of.Doors

# Removing the temporary column
dataset_cars$Number.of.Doors2 <- NULL

sum(is.na(dataset_cars$Number.of.Doors))
```

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

```
#outliers:
#In this particular column there's no outliers
boxplot(dataset_cars$Number.of.Doors,main="NO.OF doors_outlier detection",
        ylab="NO.OF.doors",xlab="Values",col='green')
```



Summary of column-9: 1.Numerical column 2.3 null values are there 3.Treated null values with Mice library 4.No outliers

column-10 : Market.Category

```
summary(dataset_cars$Market.Category)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
#Converting to lowercase:
dataset_cars$Market.Category <- tolower(dataset_cars$Market.Category)
#looking for null values:
sum(is.na(dataset_cars$Market.Category))
```

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

```
#lookng for special characters:
special_characters(dataset_cars$Market.Category)
```

```
## ,/-
```

```r
#removing all the special characters:
dataset_cars$Market.Category <- gsub(" ","",dataset_cars$Market.Category)
dataset_cars$Market.Category <- gsub(",","",dataset_cars$Market.Category)
dataset_cars$Market.Category <- gsub("/","",dataset_cars$Market.Category)
dataset_cars$Market.Category <- gsub("\\-","",dataset_cars$Market.Category)

table(dataset_cars$Market.Category)
```

```
##
##                                      crossover
##                                            943
##                                  crossoverdiesel
##                                              5
##            crossoverexoticluxuryhighperformance
##                                              1
##                crossoverexoticluxuryperformance
##                                              1
##      crossoverfactorytunerluxuryhighperformance
##                                             21
##          crossoverfactorytunerluxuryperformance
##                                              2
##                crossoverfactorytunerperformance
##                                              4
##                                crossoverflexfuel
##                                             55
##                          crossoverflexfuelluxury
##                                              9
##              crossoverflexfuelluxuryperformance
##                                              6
##                    crossoverflexfuelperformance
##                                              3
##                                crossoverhatchback
##                                             59
##       crossoverhatchbackfactorytunerperformance
##                                              6
##                          crossoverhatchbackluxury
##                                              6
##                   crossoverhatchbackperformance
##                                              6
##                                 crossoverhybrid
##                                             38
##                                crossoverluxury
##                                            337
##                            crossoverluxurydiesel
##                                             29
##                  crossoverluxuryhighperformance
##                                              8
##                            crossoverluxuryhybrid
##                                             22
##                       crossoverluxuryperformance
##                                             96
##                 crossoverluxuryperformancehybrid
##                                              2
```

```
##                             crossoverperformance
##                                               58
##                                           diesel
##                                               72
##                                      dieselluxury
##                                               45
##                 exoticfactorytunerhighperformance
##                                               16
##           exoticfactorytunerluxuryhighperformance
##                                               43
##               exoticfactorytunerluxuryperformance
##                                                2
## exoticflexfuelfactorytunerluxuryhighperformance
##                                               12
##               exoticflexfuelluxuryhighperformance
##                                                9
##                               exotichighperformance
##                                              216
##                                       exoticluxury
##                                               11
##                         exoticluxuryhighperformance
##                                               66
##                 exoticluxuryhighperformancehybrid
##                                                1
##                             exoticluxuryperformance
##                                               30
##                                  exoticperformance
##                                                9
##                         factorytunerhighperformance
##                                               89
##                                  factorytunerluxury
##                                                2
##                 factorytunerluxuryhighperformance
##                                              179
##                     factorytunerluxuryperformance
##                                               24
##                             factorytunerperformance
##                                               76
##                                         flexfuel
##                                              734
##                                      flexfueldiesel
##                                               14
##       flexfuelfactorytunerluxuryhighperformance
##                                                1
##                                      flexfuelhybrid
##                                                2
##                                      flexfuelluxury
##                                               31
##                     flexfuelluxuryhighperformance
##                                               28
##                         flexfuelluxuryperformance
##                                               22
##                               flexfuelperformance
##                                               70
```

```
##                       flexfuelperformancehybrid
##                                               2
##                                       hatchback
##                                             529
##                                 hatchbackdiesel
##                                              10
##          hatchbackfactorytunerhighperformance
##                                              11
##       hatchbackfactorytunerluxuryperformance
##                                               9
##             hatchbackfactorytunerperformance
##                                              16
##                                 hatchbackflexfuel
##                                               6
##                                 hatchbackhybrid
##                                              62
##                                 hatchbackluxury
##                                              39
##                           hatchbackluxuryhybrid
##                                               1
##                       hatchbackluxuryperformance
##                                              32
##                             hatchbackperformance
##                                             201
##                                 highperformance
##                                             160
##                                          hybrid
##                                             102
##                                          luxury
##                                             713
##                           luxuryhighperformance
##                                             279
##                       luxuryhighperformancehybrid
##                                              11
##                                    luxuryhybrid
##                                              42
##                                luxuryperformance
##                                             555
##                           luxuryperformancehybrid
##                                               7
##                                              na
##                                            3196
##                                     performance
##                                             495
##                                performancehybrid
##                                               1
```

summary of column-10: 1.chategorical column 2.No null values 3.found some special characters(,/.) 4.So removed all the special characters and unwanted spaces

column-11: Vehicle.Size

```
summary(dataset_cars$Vehicle.Size)
```

```
##    Length     Class      Mode
```

```
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
#Converting to lowercase:
dataset_cars$Vehicle.Size <- tolower(dataset_cars$Vehicle.Size)
#looking for null values:
sum(is.na(dataset_cars$Vehicle.Size))
```

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

```
#lookng for special characters:
special_characters(dataset_cars$Vehicle.Size)
table(dataset_cars$Vehicle.Size)
```

```
##
## compact   large midsize
##    3992    2342    3666
```

summary of column-11: 1.chategorical colum 2.no special characters except spaces 3.removed all the spaces and cleaned 4.No null values

column-12:Vehicle.Style

```
summary(dataset_cars$Vehicle.Style)
```

```
##    Length     Class      Mode
##     10000 character character
```

```
#finding how many times a word got repeated and finding frequency of the words
#Converting to lowercase:
dataset_cars$Vehicle.Style <- tolower(dataset_cars$Vehicle.Style)
#looking for null values:
sum(is.na(dataset_cars$Vehicle.Style))
```

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

```
#lookng for special characters:
special_characters(dataset_cars$Vehicle.Style)
#removing space between words:
dataset_cars$Vehicle.Style <- gsub(" ","",dataset_cars$Vehicle.Style)
table(dataset_cars$Vehicle.Style)
```

```
##
##      2drhatchback            2drsuv      4drhatchback            4drsuv
##               411               118               582              2112
##      cargominivan          cargovan       convertible    convertiblesuv
##                61                82               660                24
##             coupe     crewcabpickup extendedcabpickup  passengerminivan
##              1016               560               541               355
##      passengervan  regularcabpickup             sedan             wagon
##               112               343              2536               487
```

summary of column-12: 1.Categorical column 2.no null values 3.No special characters only spaces are there.

column-13 : highway.MPG

```
summary(dataset_cars$highway.MPG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   22.00   25.00   26.59   30.00  354.00
```

```
#validating data:
is.numeric(dataset_cars$highway.MPG)
```

```
## [1] TRUE
```

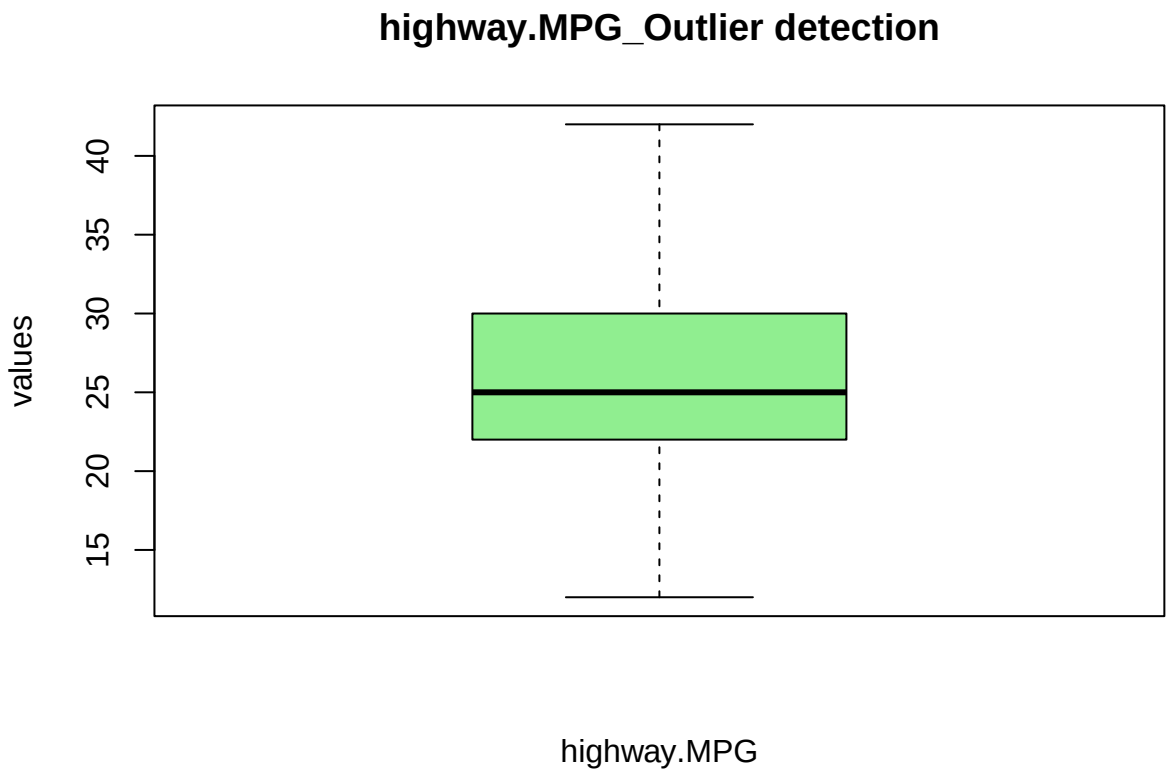
```
#checking null values:
sum(is.na(dataset_cars$highway.MPG))
```

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

```
#searching for special characters:
special_characters(dataset_cars$highway.MPG)
#outliers:
#treating outliers:
#calculating Interquantile range
IQR_3 <- 30 - 22
LB_3 <- 22 - 1.5 * IQR_3
UB_3 <- 30 + 1.5 * IQR_3
#finding outliers and represemting them using boxplot
outliers_3 <- dataset_cars$highway.MPG[dataset_cars$highway.MPG < LB_3 |
                                       dataset_cars$highway.MPG > UB_3]
outliers_3
```

```
##   [1]  46  99  82  50  48 105 101  47  92 111  50  43  44  45  90 109  43  43
##  [19]  50 105  45  90  47  53 101 105  48  44  44 109  74  48  47  44  43  43
##  [37]  45  44  44  50  47  46  74  43  44 106  47 101  47  94  43  45  50  44
##  [55]  44 103  99  46  43 108  45 101  46 105 101  48  44  82 102 109 101  46
##  [73]  44  46 101 354  45  50  46  45  44  46  44  43  43  90  48  50  46  43
##  [91]  46 101 109  44  44  48  92  97 111  46  53  92  46  44 105 106  45  46
## [109]  48  97  48 105  48  46  50  46  90  43  44  92  98  43 101  45  44  48
## [127]  43 102  99 100  46  43  50  48  92  43 109  74  99  46 109  44  45  99
## [145] 101  43  48  48 103  48  50  43  46 105  46  44  44  44  45  44  46  44
```

```
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound vaue
dataset_cars$highway.MPG[dataset_cars$highway.MPG < LB_3] <- LB_3
#HERE all values greater than upper bound value will be converted into upper bound vaue
dataset_cars$highway.MPG[dataset_cars$highway.MPG > UB_3] <- UB_3
#Boxplot to prove that there are no outliers:
boxplot(dataset_cars$highway.MPG,main="highway.MPG_Outlier detection",
        ylab="values",xlab="highway.MPG",col="lightgreen")
```



summary of column-13: 1.Numerical column 2.There are no null values 3.There are outliers 4.Treated outliers using IQR method. ———————————————————————————————————————-

column-14: city.mpg

```
summary(dataset_cars$city.mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     7.0    15.0    18.0    19.7    22.0   137.0
```

```
#validating data:
is.numeric(dataset_cars$city.mpg)
```

```
## [1] TRUE
```

```
#checking null values:
sum(is.na(dataset_cars$city.mpg))
```

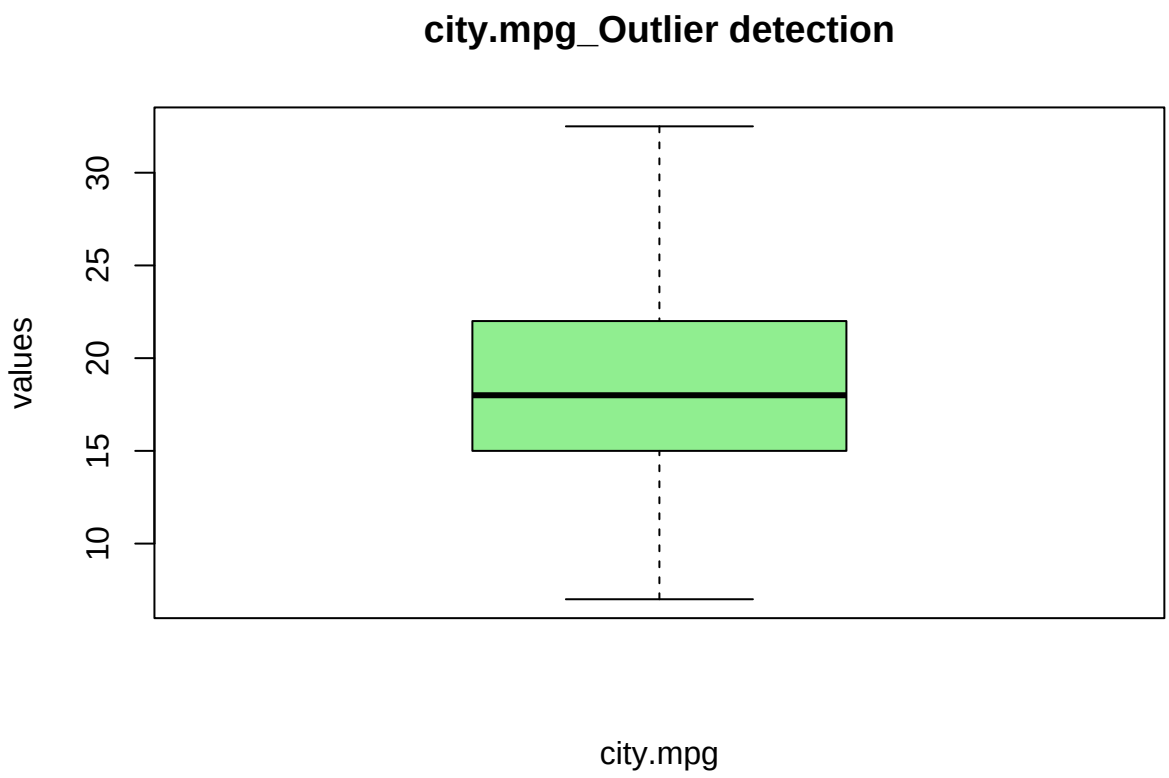
```
## [1] 0
```

```
#searching for special characters:
special_characters(dataset_cars$city.mpg)
#outliers:
#treating outliers:
```

```r
#calculating Interquantile range
IQR_4 <- 22 - 15
LB_4 <- 15 - 1.5 * IQR_4
UB_4 <- 22 + 1.5 * IQR_4
#finding outliers and represemting them using boxplot
outliers_4 <- dataset_cars$city.mpg[dataset_cars$city.mpg < LB_4 |
                                    dataset_cars$city.mpg > UB_4]
outliers_4
```

```
##   [1] 110  85  54  51 126 126  35  44 120 137  42  54  35  39  41  40  41  50
##  [19]  34  44  88 128  38  54  33  42 102  35  42  88  44  40  36  49  33  55
##  [37] 126 132  42  37  41 128  40  78  33  42  33  44  34  49  41  35  36  37
##  [55]  40  33  37  41  40  54  49  53  78  49  40 129  34  44 126  44  36  86
##  [73]  34  50  33  44  35  54  41  43  41 121  40 110  53  34 122  34  39 126
##  [91]  53 132 124  43  51  41  85 101  33 128  41 126  39  40  41  47  34  98
## [109]  34  35  54  47  41  53  44  37  39  37  43  41  88  36  42  54  43  40
## [127]  41 126 128  41  36  41  42 120  44  94  35 137  53  42  36  35  55 120
## [145]  40  53  36 126  95  36  36  50  39  40  43  44  41  42  94  42 126  51
## [163]  53  54  53  40  88  44  41 120  89  38  44  36  41  33  35  44  41  40
## [181]  40  44 126  44  34  34  44  33  36  42  35 101 110  97  34  41  35  54
## [199]  42 120  34  33  34 128  78  35  36 126  36  53  40 128  41  50 126  33
## [217]  36 126  33  34  51  42 121  51  34  34  54  53  36 126  41  35  43  53
## [235]  41  41  34  42  41  36  43  43  37  53  41
```

```r
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound vaue
dataset_cars$city.mpg[dataset_cars$city.mpg < LB_4] <- LB_4
#HERE all values greater than upper bound value will be converted into upper bound vaue
dataset_cars$city.mpg[dataset_cars$city.mpg > UB_4] <- UB_4
#Boxplot to prove that there are no outliers:
boxplot(dataset_cars$city.mpg,main="city.mpg_Outlier detection",ylab="values",
        xlab="city.mpg",col="lightgreen")
```



summary of column-14: 1.Numerical column 2.There are no null values 3.There are outliers 4.Treated outliers using IQR method.

---

column-15 : Popularity

```
summary(dataset_cars$Popularity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       2     549    1385    1558    2009    5657
```

```
#validating data:
is.numeric(dataset_cars$Popularity)
```

```
## [1] TRUE
```

```
#checking null values:
sum(is.na(dataset_cars$Popularity))
```

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

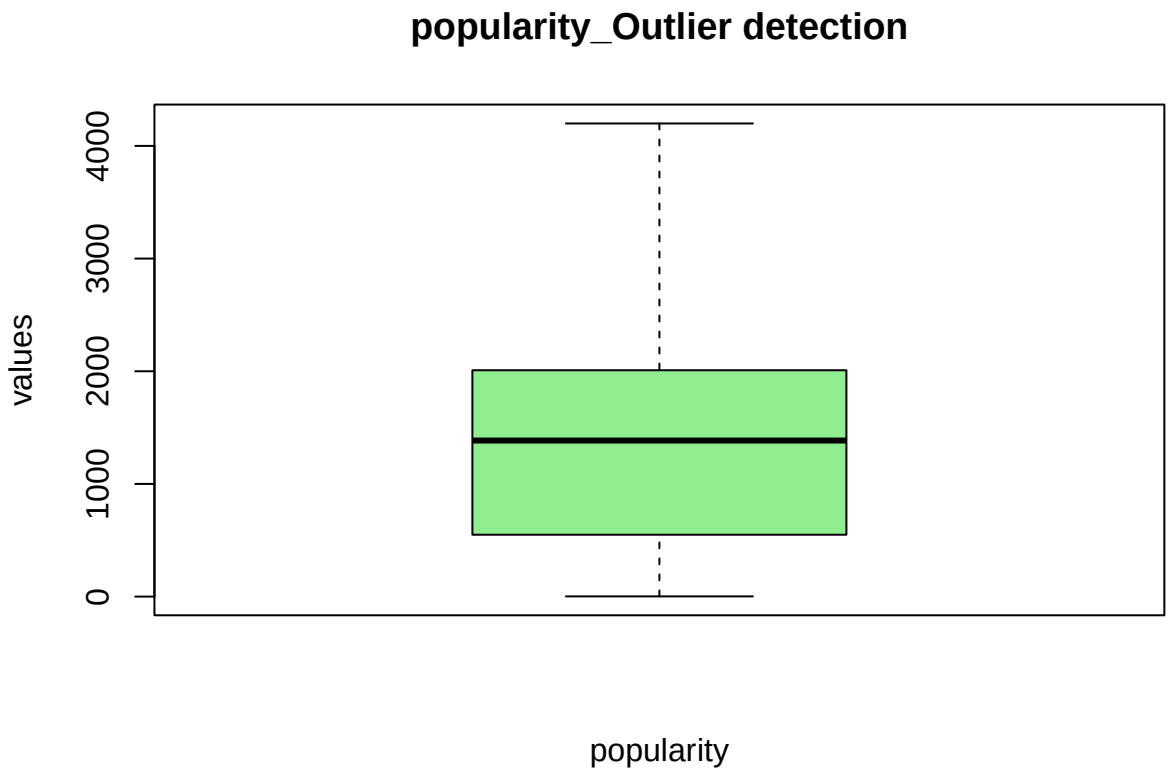
```
#searching for special characters:
special_characters(dataset_cars$Popularity)
#outliers:
```

```r
#treating outliers:
#calculating Interquantile range
IQR_5 <- 2009 - 549
LB_5 <- 549 - 1.5 * IQR_5
UB_5 <- 2009 + 1.5 * IQR_5
#finding outliers and represemting them using boxplot
outliers_5 <- dataset_cars$Popularity[dataset_cars$Popularity < LB_5 |
                                      dataset_cars$Popularity > UB_5]
outliers_5
```

```
##   [1] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [16] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [31] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [46] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [61] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [76] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
##  [91] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [106] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [121] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [136] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [151] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [166] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [181] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [196] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [211] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [226] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [241] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [256] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [271] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [286] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [301] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [316] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [331] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [346] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [361] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [376] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [391] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [406] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [421] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [436] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [451] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [466] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [481] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [496] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [511] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [526] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [541] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [556] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [571] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [586] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [601] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [616] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [631] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
```

```
## [646] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [661] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [676] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [691] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [706] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [721] 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657 5657
## [736] 5657 5657 5657 5657 5657 5657
```

```r
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound vaue
dataset_cars$Popularity[dataset_cars$Popularity < LB_5] <- LB_5
#HERE all values greater than upper bound value will be converted into upper bound vaue
dataset_cars$Popularity[dataset_cars$Popularity > UB_5] <- UB_5
#Boxplot to prove that there are no outliers:
boxplot(dataset_cars$Popularity,main="popularity_Outlier detection",ylab="values",
        xlab="popularity",col="lightgreen")
```



summary of column-15: 1.Numerical column 2.There are no null values 3.There are outliers 4.Treated outliers using IQR method.

column-16: MSRP

```r
summary(dataset_cars$MSRP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2000   20960   29935   40341   42146 1705769
```

```r
#validating data:
is.numeric(dataset_cars$MSRP)
```

```
## [1] TRUE
```

```r
#checking null values:
sum(is.na(dataset_cars$MSRP))
```

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

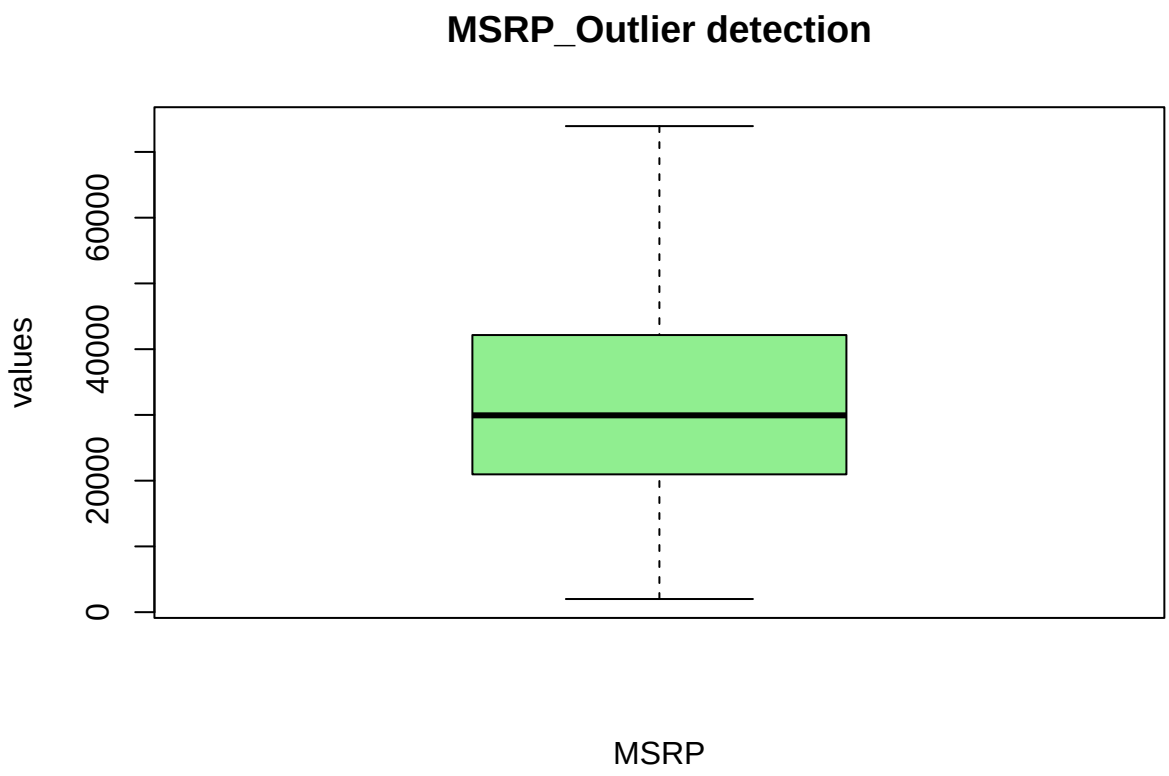
```r
#searching for special characters:
special_characters(dataset_cars$MSRP)
#outliers:
#treating outliers:
#calculating Interquantile range
IQR_6 <- 42146 - 20960
LB_6 <- 20960 - 1.5 * IQR_6
UB_6 <- 42146 + 1.5 * IQR_6
#finding outliers and represemting them using boxplot
outliers_6 <- dataset_cars$MSRP[dataset_cars$MSRP < LB_6
                                | dataset_cars$MSRP > UB_6]
outliers_6
```

```
##   [1]   93450   78335   83015  440000  392400  366000  202000   74600   93295
##  [10]  443800  166100   95895  151500  224670  231800  123845   95895   82900
##  [19]  118795   96200  182700   82500   83195  122600   82305  228080  108600
##  [28]  236100  218500  154600   81500   74145  145200  423500  208000   74260
##  [37]  480175   78570   91200   94100  153000  497650   84300   75570  303700
##  [46]  108840  199700  132800  136400  107385   84300  144700   87577  213200
##  [55]   74000  225400  267000  303700   74195   84160  200400  122400   80045
##  [64]  202000  214670   98500   77300   78100  102930  166900   84470   80000
##  [73]  203295  131200   84995   89995  438325  335000   83295  159600   98172
##  [82]  247900  153400   81795  118400  108050  548800   75200  198600  283900
##  [91]  109000  200800  294900   77200  319900   78995  270400  128300  144995
## [100]   91900  100100  251600  198700   74950  171500  135200  182500  156000
## [109]   79500   79900   75570   92000  137000   82700   88900 1500000  142100
## [118]  105300  363000   87252  262990  137500   77900  259900  126100  241200
## [127]   89000  133205  296000   92400  165627  111900  113400  184105  138400
## [136]  132800  203995   82965  151350  470350  313088  217900   75465  131400
## [145]  195895  140615  142995  295850   88600   79400  151800   94600  129400
## [154]   86950  114900  189600   81300   88100  328990  120440   89000  170545
## [163]  206300  162900  145500   82305   98300  225400  142100  224990  140500
## [172]  275461  143400  187900   97395  124000   74600  117530  110800   81295
## [181]  236100  440000  214600  138195  225400   89622   83000  149995  399500
## [190]   89000  199800  184309   96100   86215  309900  198900   99900  115700
## [199]   77045   87400  180535  336400   74995   82570   76050  224650  214500
## [208]  113400   96200  106995   84500  492425   89825  218400   74135   78100
## [217]   99900  199600  198190   84070  161100  302695  201213   75000  179645
## [226]  137995  265500  110800  182395  329325  185800  164700  213250   96500
## [235]   89275  132800   97200  114200  217890  291744  359990   99200  441600
## [244]  181200  119900  106700   93075  107995   97250   76395  114900  199900
## [253]   82645  335600   74700   79100  295000   95195   79995   84545  104500
```

```
## [262]  103195  111350  304350   94950  118160   79450   93600  104600   77100
## [271]  209600  201000   85935  200500   84325  315888   83870  209500  298900
## [280]  337000   94900  180535  125600  113400  143300  147495   98800  191900
## [289]   75200   92275   80300   75700  207700  132825  109900   97400  227600
## [298]  218310  495000   85542   79800  135795  234050  102930  136750  242990
## [307]  222300   85995   84300  230900  191995   92195  101700  109300   76970
## [316]  291900   86270  131500  186495  107995   75465  123500  206000  184200
## [325]  176287  115900  118845  286750 1705769  187925   75000   81013  397500
## [334]  198295  189900  115710  535500  104300   88300  114900  266000   96300
## [343]  101700   76875  101690  117995  237600  163000  219400  107100 1382750
## [352]  205840   76650  128400  229990  267000  149990   88900   97795  279900
## [361]  221990  643330   94200  102100  202000  191900  212800  237250  160300
## [370]  165986  108900  170829  110700   86877   86965  182700  110800   91875
## [379]   78000  106550  225400  169050  108195  237600   90900  154090   79100
## [388]  239400  117500   75010   82800  224585  136900  115900  114900  263200
## [397]  221990  479775   84500   74100  257412  194600  105630  111510  101770
## [406]   79000  402940   79200  104122  154495   93400   90825  201500  196100
## [415]  279500  139995   81500   79980  548800  301695   93200   94795   96100
## [424]  263990   79570  250100   76645   83600  188100  103400   92400   84000
## [433]   84950   89950   98200   90300   76400   86450  175100   75195   89350
## [442]   93225  433550   87495   91415  132825   82633   92350   89500   88850
## [451]  455500  200500   83450  119450  103200   74100  130000  248500  280225
## [460]   98700  119700  345400  195840  118605  256650  209990   81855   88435
## [469]  229447  474600  116200  107995   80600  101770  109990  354000  143250
## [478]  195200  450000   76100  190600  397500  149990  223970  246990   75500
## [487]  506500  219900   85000  153195  280400  310543  375000  180535   84885
## [496]  149995  191400  310543  115400   93200  141495  109900   83495  209990
## [505]  270990  224585  250000  379050   89380  217890  286739  268660   85100
## [514]  187900   84145   74295  110475  113600   84300   83495  105630   87465
## [523]  119450   77900  150900   86600  412000  117530  138800  153900  129900
## [532]   94995  206000   83995  203500   78395  207895  239340  180408   80655
## [541]   78570  150694  228625   84800  490700  141300  195895  287650  248000
## [550]   82800  417825  200500  121550  196500  108900   91500   79900  180300
## [559]  209500  151100   96600   99950   93850  234260  105000  184900   93600
## [568]  219990  147332   95650  313088  114200   98995  315888   89622  497650
## [577]   84300  260000  120900  320580   87800   98800  111200  206600   82795
## [586]  177500  151200   76600  263553   75300  273104  173500   78775  267000
## [595]   88880  320580   91900  275861  170750  147300  173079  128300  294025
## [604]   80650   76000   89400  299900  160829  456500  198973  131300   74850
## [613]   78750   81900  118795  117300  128000   82270  149700   99600   92900
## [622]  161070  223600   98300   75000  223295   98400  207895  137100   87500
## [631]   91030   78600  105300  160300  329990   78695  116700  110300  131300
## [640]   74500  102200  209600  492000  106500  131200  136750  176400   76200
## [649]  163150   76975  364000  283695  130400  130400  148795   89600   98872
## [658]  198195   95500  139995  101300   76100  116000  405500   74200  284576
## [667]   85200  151100  183000  145740   81300   82500  238700  234945  284976
## [676] 1380000   83465   88495  184200  156300   78820  120060   80155   82900
## [685]  117530   74260   98650   94400   94600  158700  141800  430450  198190
## [694]  124900   92395  182009  100100   81470  239400  159200   89950  120440
## [703]  143860  295000  101995  228339   83300   83995  198250  126500  173800
## [712]   77350   80000   85650  290000  217000   92495   74425   99500  163000
## [721]  233509  156300  145500  114700   87145  211000  313088  112700   76685
## [730]   82500  120395  114100  139900  109500   91650  195100   75995   94400
## [739]   82500   82895  118200   79900  186925  122500  305650  284900   85895
```

```
## [748]  210700  222000  197850   92600   97900  165627  410000   91700  104215
## [757]   81000   91950   88395  183695  141500  160900   79950  238500  296387
## [766]  191900  182009  180300  137900  319400  382400  102930  120000  449525
## [775]  103300  157300  191900   87895  208295  417000  161100   98900   85050
## [784]  226900  340990   77295  200054   95895  145740   79700  141200  257412
## [793]  228625  119900  288000  463000   94000   74350  263990   83000   89200
## [802]  191300   85542   74100   91000   87070   87300  100800  369200  248000
## [811]  226850  162900   91030  281170   85795  233509  296295  175900  106900
## [820]  150465  495000   98900  134295   96200   77600  187124   83825  263400
## [829]  334990  295987  103900  199500  217550
```

```r
#removing outliers:
#HERE all values lesser than lower bound value will be converted into lower bound value
dataset_cars$MSRP[dataset_cars$MSRP < LB_6] <- LB_6
#HERE all values greater than upper bound value will be converted into upper bound value
dataset_cars$MSRP[dataset_cars$MSRP > UB_6] <- UB_6
#Boxplot to prove that there are no outliers:
boxplot(dataset_cars$MSRP,main="MSRP_Outlier detection",ylab="values",
        xlab="MSRP",col="lightgreen")
```



summary of column-16: 1.Numerical column 2.There are no null values 3.There are outliers 4.Treated outliers using IQR method.

so the whole process of cleaning is completed.All the columns of the dataset have been cleaned and preprocessed correctly

TASK-2:

Basic Statistical analysis is done in each and everystep of task-1

Business understanding:

Column-1: MAKE ->Categorical column -> Highest NO.OF cars manufactured by chevrolet
-> Lowest NO.OF cars manufactured by Bugatti

column-2:Model ->categorical column ->Highest NO.OF car model being sold is Beetle Convertible ->There are many models which have sold less than 5.

column-3:Year ->Numerical column ->Latest car is being manufactured in 2017 -> Oldest car is manufactured in 1990 ->No null values

column-4:Engine.Fuel.Type ->categorical column ->Most NO.OF cars having their engine type as regular unleaded ->very few cars are having their engine type as natural gas

column-5:Engine.HP ->Numerical column ->Mean: 249 ->Median: 227 ->Cars having 1001 HP as their engine capacity, which is maximum of all cars ->Minimum car HP is 55

column-6:Engine.Cylinders ->Numerical column ->Mean: 5.86 ->Median: 6 ->Maximum no.of Cylinders for a car is 16 ->And there are cars with no cylinders

column-7:Transmission.Type ->Categorical column ->Maximum NO.OF cars are Automatic cars which are 6924.

column-8:Driven_wheels: ->categorical column ->Mainly most NO.OF cars are front wheel drive type cars which are 3996. ->There are Less NO.OF four wheel drive which are 1200.

column-9: NO.Of Doors ->Numerical column ->Mean:3.43 ->Median:4 ->Minimum NO.OF doors per car is 2. -> Maximum NO.OF doors per car is 4.

column-10: Market.Category ->Numerical column ->Mean:3.43 ->Median:4 ->Mostly Crossover is the market type of 998 cars.

column-11:Vehicle.size ->Categorical column ->3992 cars are being compact.

column-12:Vehicle.Style ->Categorical column ->Most cars are having sedan style structure

column-13:Highway.MPG ->Numerial column ->mean: 26.59 ->median: 25.00

column-14:city.mpg ->Numerical column ->mean:19.7 ->Median:18

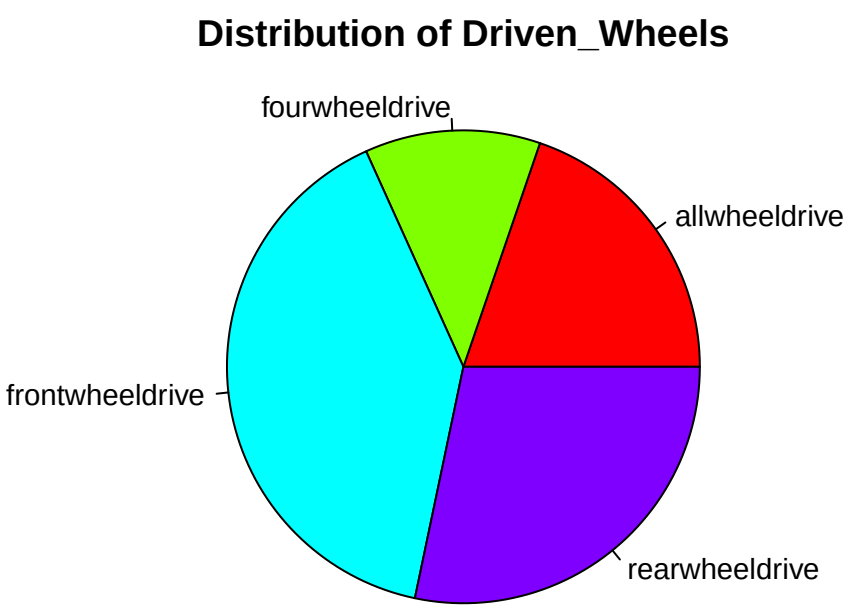
column-15:Popularity ->Numerical column ->Mean:1558 ->Median:1385

column-16:MSRP ->Numerical column ->mean : 40341 ->median: 29935

completion of business understanding

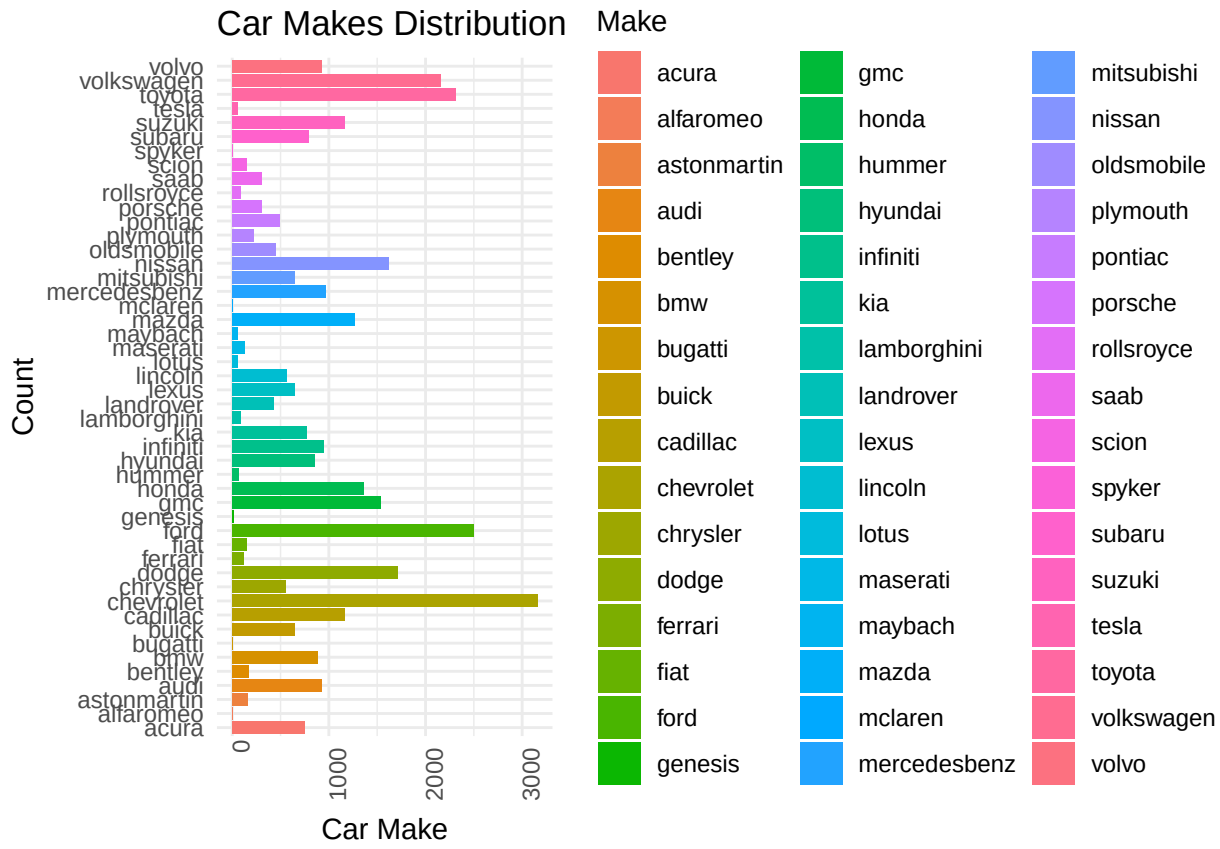
PIECHART

```
#Piechart of Driven_wheels:
pie_dw <- table(dataset_cars$Driven_Wheels)
pie(pie_dw, main = "Distribution of Driven_Wheels",
    col = rainbow(length(pie_dw)), radius = 1, cex = 0.9)
```



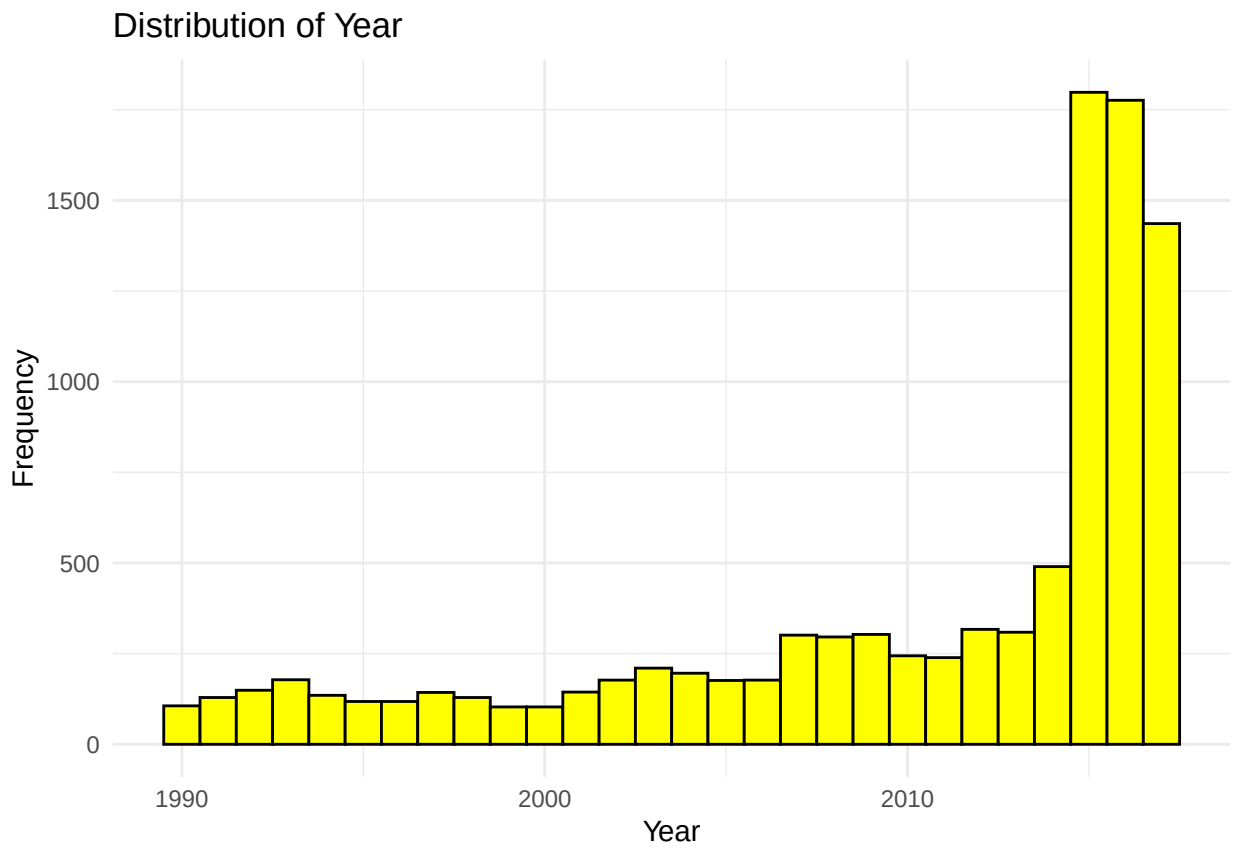
bar-chart

```
#Building bar-graph between Make column
#Using ggplot library for plotting
ggplot(dataset_cars, aes(x = Number.of.Doors , y = Make  , fill = Make)) +
  geom_bar(stat = "identity") +
  labs(title = "Car Makes Distribution", x = "Car Make", y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1.2))
```



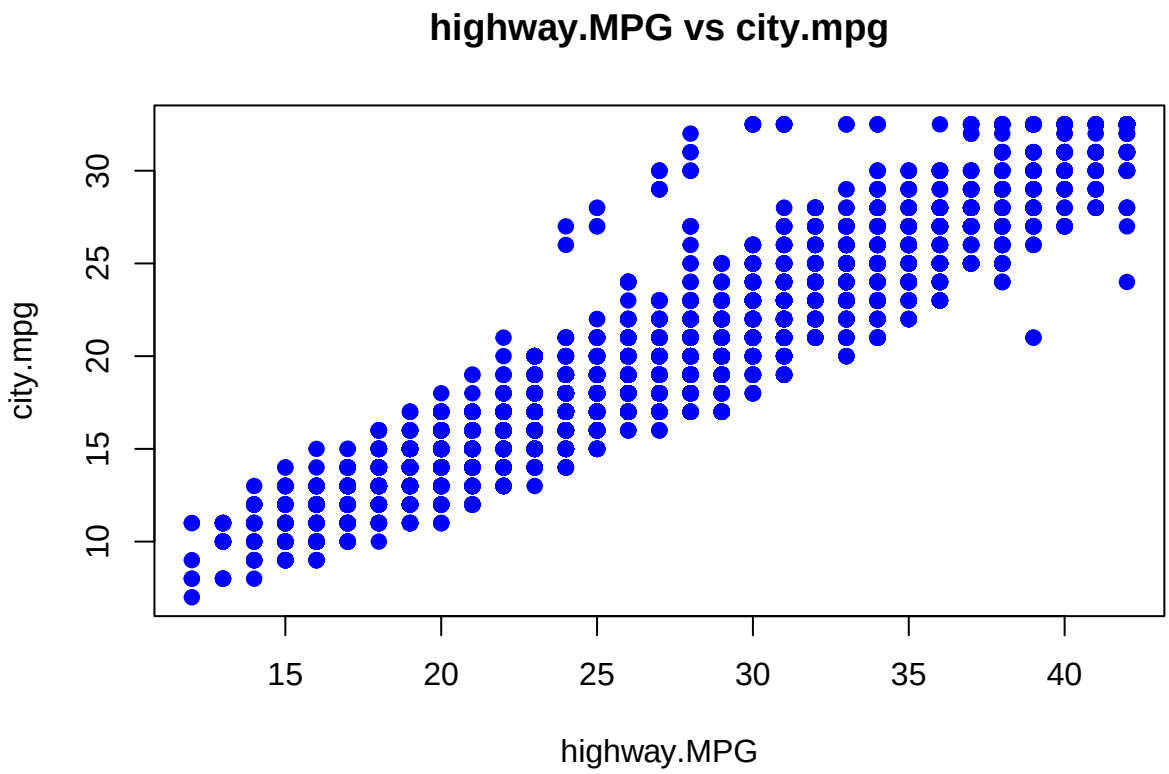
histogram

```
#histogram of Year column:
ggplot(dataset_cars, aes(x = Year)) +
  geom_histogram(binwidth = 1, fill = "yellow", color = "black") +
  labs(title = "Distribution of Year", x = "Year", y = "Frequency") +
  theme_minimal()
```



Scatter plot

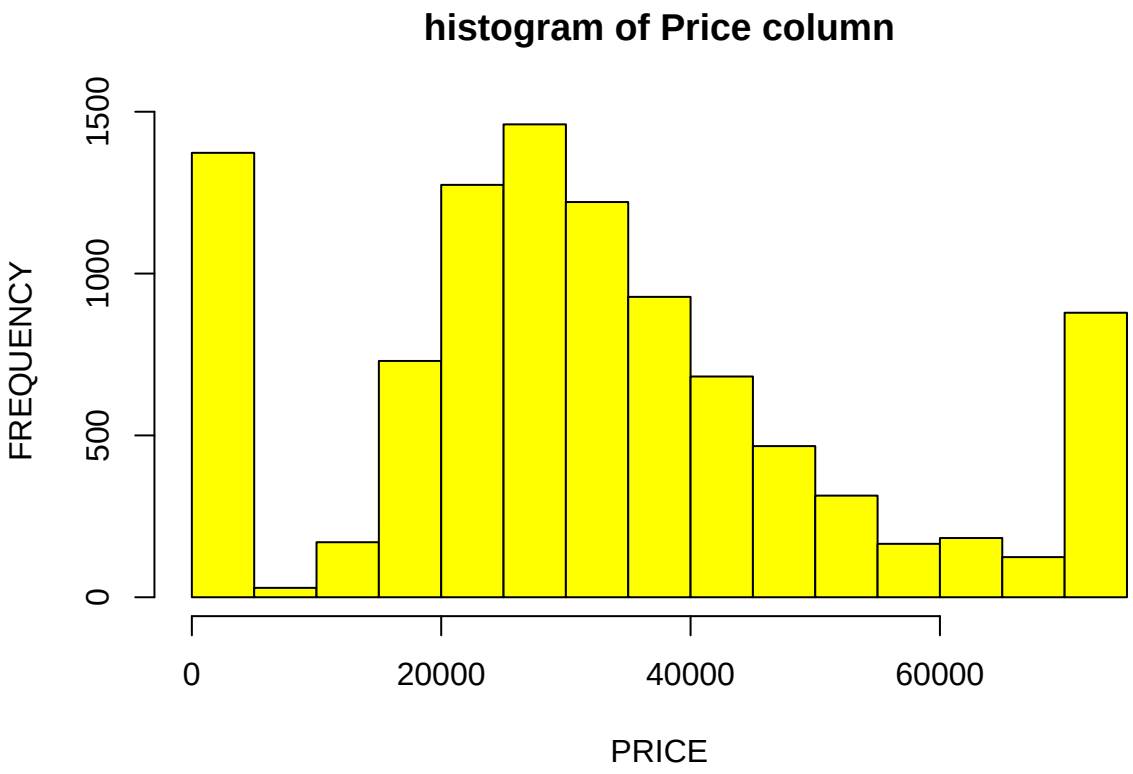
```
# Scatterplot of highway.MPG vs city.mpg
plot(dataset_cars$highway.MPG,dataset_cars$city.mpg,
     main="highway.MPG vs city.mpg",xlab="highway.MPG",ylab="city.mpg",
     pch=19,col="blue")
```



TASK - 3

Analysis on price variable:

```
# 4A :
#histogram of price variable:
hist(dataset_cars$MSRP,main="histogram of Price column",xlab="PRICE",
              ylab="FREQUENCY",col="yellow",border="black")
```



Explanation and summary of Histogram

- From the above histogram of price(MSRP) column:
- More than 1250 cars price ranges from 0 - 5000
- Very few cars price ranges from 5000 - 10000
- More than 100 cars price ranges from 10000 - 15000
- More than 750 cars price ranges from 15000 - 20000
- More than 1200 cars price ranges from 20000 - 25000
- More than 1250 cars price ranges from 25000 - 30000
- More than 1100 cars price ranges from 30000 - 35000
- More than 750 cars price ranges from 35000 - 40000
- More than 600 cars price ranges from 40000 - 45000
- More than 400 cars price ranges from 45000 - 50000
- More than 200 cars price ranges from 50000 - 55000
- More than 100 cars price ranges from 55000 - 60000
- More than 120 cars price ranges from 60000 - 65000
- More than 100 cars price ranges from 65000 - 70000
- More than 800 cars price are from 70000 and above.

Calculation of mean median variance

```
#calculating mean:
mean(dataset_cars$MSRP)
```

```
## [1] 32347.39
```

```r
#median of MSRP
median(dataset_cars$MSRP)
```

```
## [1] 29935
```

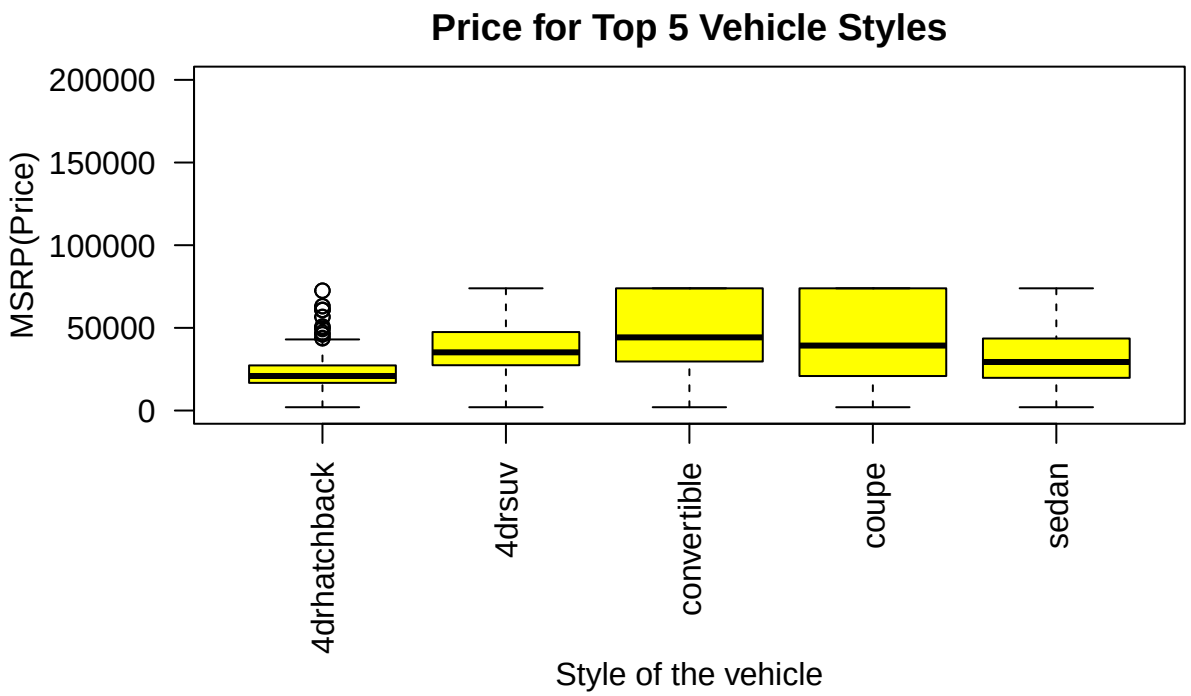
```r
#varience of price:
var(dataset_cars$MSRP)
```

```
## [1] 390277748
```

```r
# 4b:

#Group cars by price ranges
price_groups <- cut(dataset_cars$MSRP, breaks = c(0, 20000, 50000, 100000, Inf),
                    labels = c("Low(<20)", "Medium (20K-50K)", "High(50K-100K)",
                               "Luxury(>100K)"), include.lowest = TRUE)
group_summary <- table(price_groups)
print(group_summary)
```

```
## price_groups
##         Low(<20) Medium (20K-50K)   High(50K-100K)    Luxury(>100K)
##             2302             6033             1665                0
```

```r
# 4c:

# Boxplot for different car types
vehicle_style_01<- names(sort(table(dataset_cars$Vehicle.Style),
                              decreasing = TRUE))[1:5]
vehicle_style_01<-dataset_cars[dataset_cars$Vehicle.Style %in% vehicle_style_01,]


par(mar = c(11,5.5,2,1) +0.1)
par(mgp = c(3, 1, 0))

boxplot(MSRP ~ Vehicle.Style, data = vehicle_style_01,
        main = "Price for Top 5 Vehicle Styles", xlab = " ",
        ylab = " ", col = "yellow", las = 2, ylim = c(0, 200000))
mtext("Style of the vehicle", side = 1, line = 6, cex = 1)
mtext("MSRP(Price)", side = 2, line = 4, cex = 1, las = 3)
```




```
#4[d] -> Correlation with MSRP (numeric variables only)
numeric_variables <- sapply(dataset_cars, is.numeric)
correlation_matrix <- cor(dataset_cars[, numeric_variables], use = "complete.obs")
msrp_correlation <- correlation_matrix["MSRP", ]
msrp_correlation <- msrp_correlation[!names(msrp_correlation) %in% "MSRP"]
sorted_correlation <- sort(abs(msrp_correlation), decreasing = TRUE)
top_3_correlation <- sorted_correlation[1:3]

print(msrp_correlation)
```

```
##             Year        Engine.HP Engine.Cylinders  Number.of.Doors
##       0.60707874       0.82432483       0.47162049       0.02690935
##      highway.MPG         city.mpg       Popularity
##      -0.17819913      -0.25233575       0.03409810
```

```
print("Top 3 correlated variables with MSRP -> Price:")
```

```
## [1] "Top 3 correlated variables with MSRP -> Price:"
```

```
print(top_3_correlation)
```

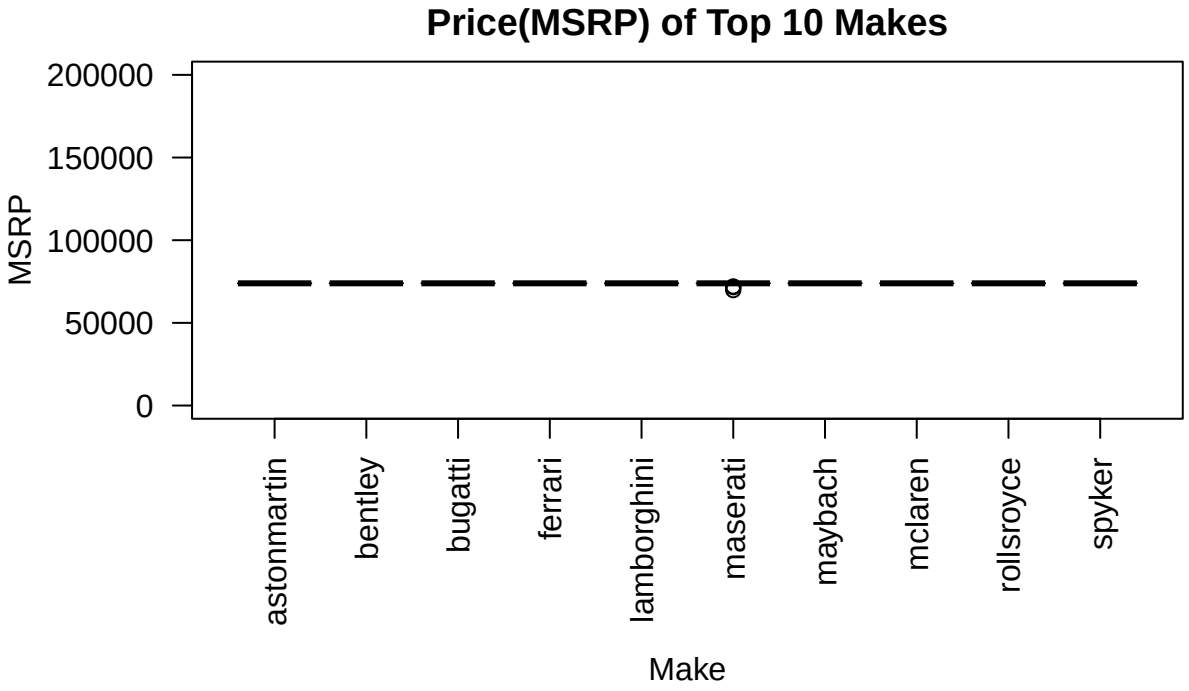
```
##        Engine.HP             Year Engine.Cylinders
##        0.8243248        0.6070787        0.4716205
```

Brand affecting the popularity and the price

```
#5:
#Summarize MSRP(Price) by Make:
brand_summary_01 <- aggregate(MSRP ~ Make, data = dataset_cars, FUN = function(x)
  c(mean = mean(x), median = median(x)))
brand_summary_01 <- brand_summary_01[order(-brand_summary_01$MSRP[, "mean"]), ]
print(head(brand_summary_01, 10))
```

```
##           Make MSRP.mean MSRP.median
## 3  astonmartin  73925.00    73925.00
## 5      bentley  73925.00    73925.00
## 7      bugatti  73925.00    73925.00
## 13     ferrari  73925.00    73925.00
## 23 lamborghini  73925.00    73925.00
## 29     maybach  73925.00    73925.00
## 31     mclaren  73925.00    73925.00
## 39  rollsroyce  73925.00    73925.00
## 42      spyker  73925.00    73925.00
## 28    maserati  73746.81    73925.00
```

```
# Representing MSRP column using boxplot:
# Boxplot of MSRP(Price) for top 10 Makes:

top10.makes <- brand_summary_01$Make[1:10]
top10.makes <- dataset_cars[dataset_cars$Make %in% top10.makes, ]
par(mar = c(11,5.5,2,1) + 0.1)
par(mgp = c(3, 1, 0))

boxplot(MSRP ~ Make, data = top10.makes, main ="Price(MSRP) of Top 10 Makes",
        xlab = "", ylab = "", col= "lightgreen", las = 2,
        cex.names = 0.9, ylim = c(0,200000))

mtext("Make", side = 1, line = 6, cex = 1)
mtext("MSRP", side = 2, line = 4, cex = 1, las = 3)
```


Price(MSRP) of Top 10 Makes
MSRP
200000
150000
100000
50000
0
astonmartin
bentley
bugatti
ferrari
lamborghini
maserati
maybach
mclaren
rollsroyce
spyker
Make

THANK YOU