COMP 8157 - ADVANCED DATABASE TOPICS

Lab 1

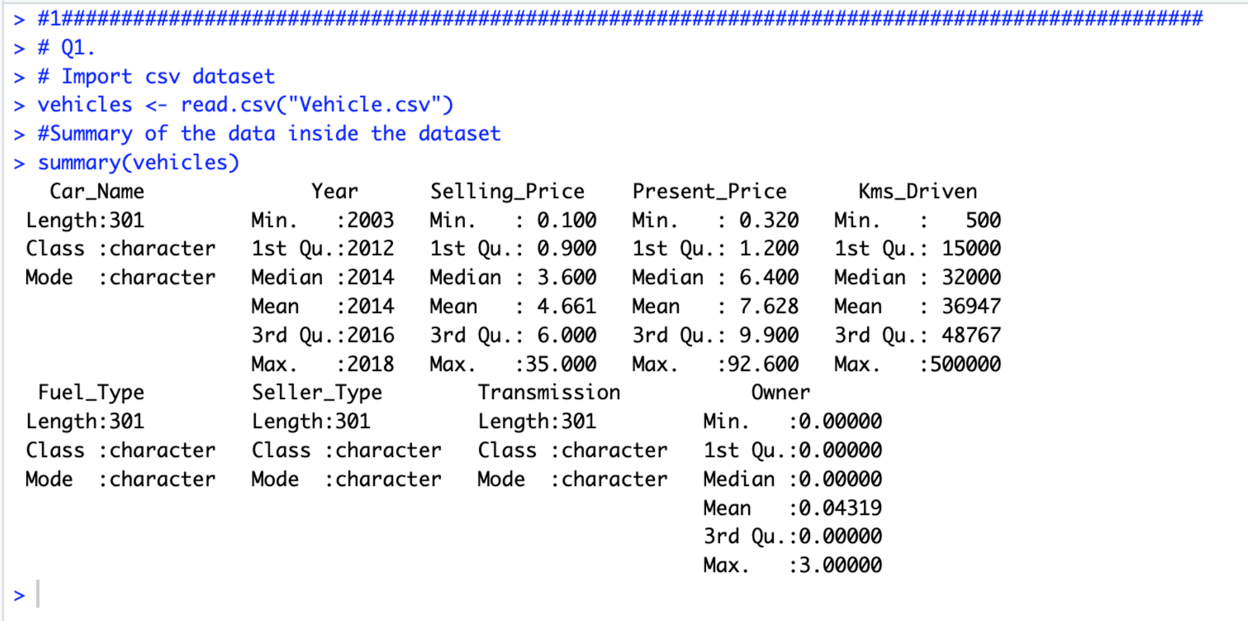
Aathik Thayyil Radhakrishnan

Student No: 110094762

25 May 2023

Q1. Import the Vehicle dataset, summarize it and explain the output.

A1: The dataset “Vehicle.csv” is imported using the read_csv() function. We can use the summary() function to get a summary or stats of each column of the dataset. We can see the output of Q1 in Fig 1. The summary function outputs details like the length, class, mode, minimum value, median, 1st and 3rd Quartile, maximum value of each column as shown in the below figure.



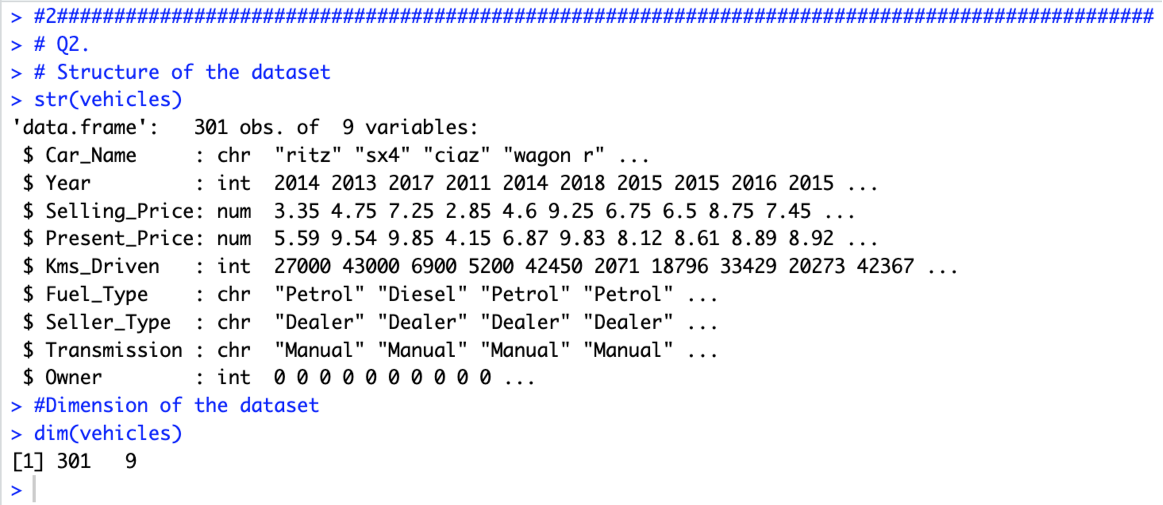
```
> #1################################################################################################
> # Q1.
> # Import csv dataset
> vehicles <- read.csv("Vehicle.csv")
> #Summary of the data inside the dataset
> summary(vehicles)
   Car_Name               Year       Selling_Price     Present_Price      Kms_Driven    
 Length:301         Min.   :2003   Min.   : 0.100   Min.   : 0.320   Min.   :   500  
 Class :character   1st Qu.:2012   1st Qu.: 0.900   1st Qu.: 1.200   1st Qu.: 15000  
 Mode  :character   Median :2014   Median : 3.600   Median : 6.400   Median : 32000  
                    Mean   :2014   Mean   : 4.661   Mean   : 7.628   Mean   : 36947  
                    3rd Qu.:2016   3rd Qu.: 6.000   3rd Qu.: 9.900   3rd Qu.: 48767  
                    Max.   :2018   Max.   :35.000   Max.   :92.600   Max.   :500000  
  Fuel_Type         Seller_Type        Transmission            Owner        
 Length:301         Length:301         Length:301         Min.   :0.00000  
 Class :character   Class :character   Class :character   1st Qu.:0.00000  
 Mode  :character   Mode  :character   Mode  :character   Median :0.00000  
                                                          Mean   :0.04319  
                                                          3rd Qu.:0.00000  
                                                          Max.   :3.00000  

> 
```

Fig 1: Output of Q1

Q2. Show the structure and dimension of the dataset and explain it.

A2: We can get the structure and dimension of the dataset using the str() function and dim() functions respectively. The str() function outputs details like the total number of observations and the number of variables, along with some details of all the columns in the dataset. The dim() function outputs the total number of observations and the number of variables in the dataset. Fig 2 shows the output for Q2.

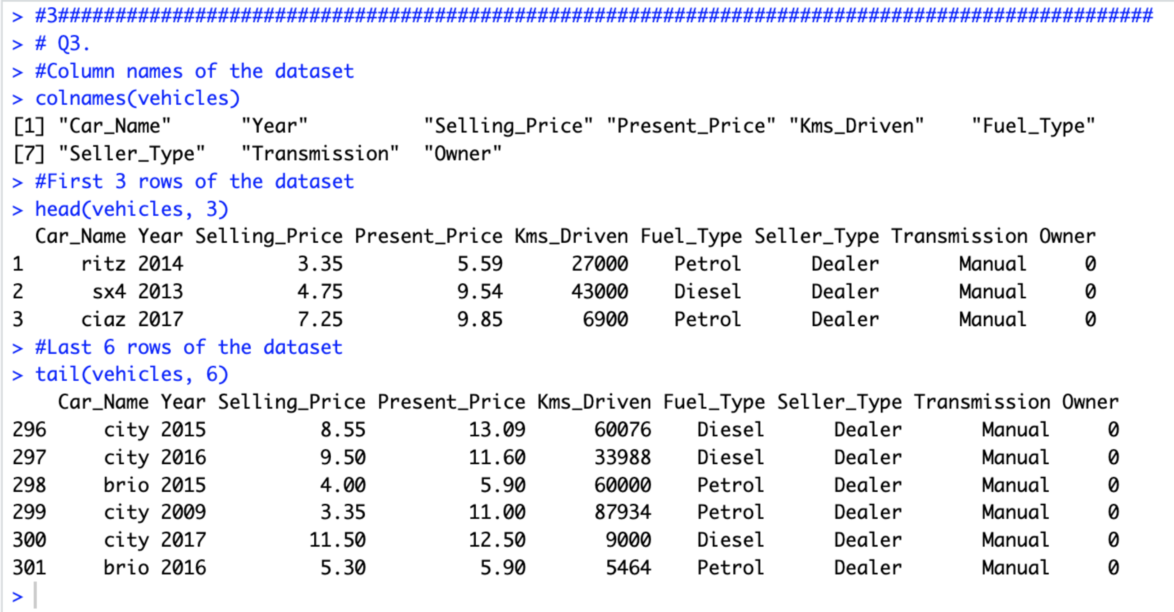


```
> #2#############################################################################################
> # Q2.
> # Structure of the dataset
> str(vehicles)
'data.frame':	301 obs. of  9 variables:
 $ Car_Name     : chr  "ritz" "sx4" "ciaz" "wagon r" ...
 $ Year         : int  2014 2013 2017 2011 2014 2018 2015 2015 2016 2015 ...
 $ Selling_Price: num  3.35 4.75 7.25 2.85 4.6 9.25 6.75 6.5 8.75 7.45 ...
 $ Present_Price: num  5.59 9.54 9.85 4.15 6.87 9.83 8.12 8.61 8.89 8.92 ...
 $ Kms_Driven   : int  27000 43000 6900 5200 42450 2071 18796 33429 20273 42367 ...
 $ Fuel_Type    : chr  "Petrol" "Diesel" "Petrol" "Petrol" ...
 $ Seller_Type  : chr  "Dealer" "Dealer" "Dealer" "Dealer" ...
 $ Transmission : chr  "Manual" "Manual" "Manual" "Manual" ...
 $ Owner        : int  0 0 0 0 0 0 0 0 0 0 ...
> #Dimension of the dataset
> dim(vehicles)
[1] 301   9
> 
```

Fig 2: Output of Q2

Q3. Show the column names of the Vehicle dataset and the first 3 rows and the last 6 rows of it.

A3: We can use the colnames() function to display all the column names in the dataset and the head() and tail() functions are used to display any number of rows from the dataset. The head() function displays rows at the beginning or top of the dataset and tail() function displays the rows from the end or bottom of the dataset. Fig 3 shows the outputs of Q3.

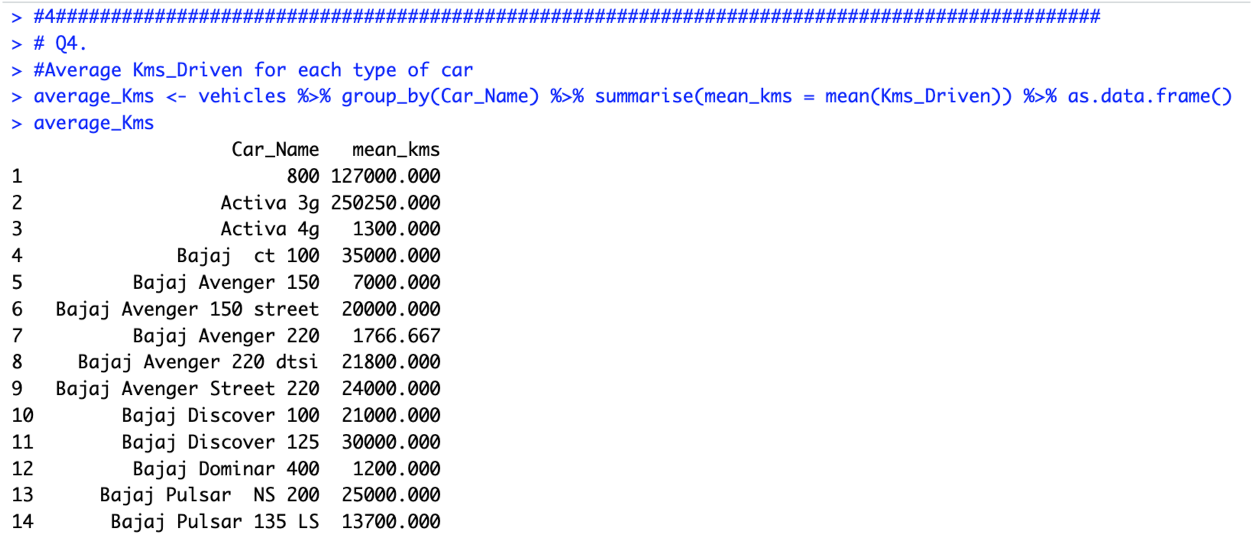


```
> #3#############################################################################################
> # Q3.
> #Column names of the dataset
> colnames(vehicles)
[1] "Car_Name"      "Year"          "Selling_Price" "Present_Price" "Kms_Driven"    "Fuel_Type"    
[7] "Seller_Type"   "Transmission"  "Owner"        
> #First 3 rows of the dataset
> head(vehicles, 3)
  Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner
1     ritz 2014          3.35          5.59      27000    Petrol      Dealer       Manual     0
2      sx4 2013          4.75          9.54      43000    Diesel      Dealer       Manual     0
3     ciaz 2017          7.25          9.85       6900    Petrol      Dealer       Manual     0
> #Last 6 rows of the dataset
> tail(vehicles, 6)
    Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner
296     city 2015          8.55         13.09      60076    Diesel      Dealer       Manual     0
297     city 2016          9.50         11.60      33988    Diesel      Dealer       Manual     0
298     brio 2015          4.00          5.90      60000    Petrol      Dealer       Manual     0
299     city 2009          3.35         11.00      87934    Petrol      Dealer       Manual     0
300     city 2017         11.50         12.50       9000    Diesel      Dealer       Manual     0
301     brio 2016          5.30          5.90       5464    Petrol      Dealer       Manual     0
> 
```

Fig 3: Output of Q3

Q4. Show the average Kms_Driven for each type of car (Car_Name) in the dataset.

A4: We can use the “dplyr” to solve this question. The package was installed and the library was imported into the R script as given the code. Once the package is installed and imported, we can use the summarize(), mean() and group_by() functions to find the average kms driven by each car. The output of Q4 is shown in Fig 4 *(note: screenshot only contains partial output screen).*

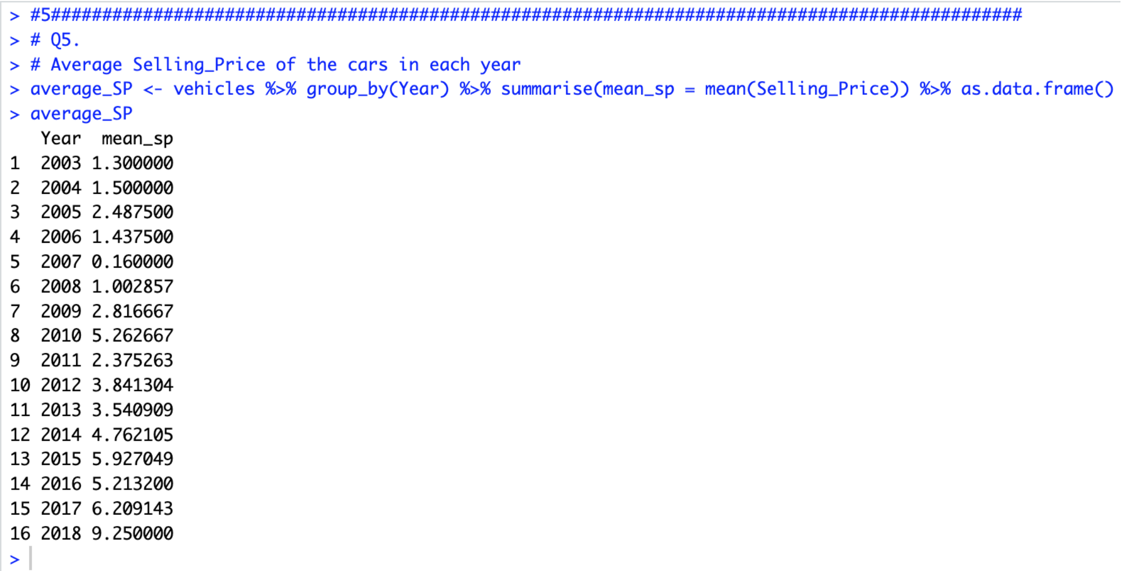


```
> #4################################################################################################
> # Q4.
> #Average Kms_Driven for each type of car
> average_Kms <- vehicles %>% group_by(Car_Name) %>% summarise(mean_kms = mean(Kms_Driven)) %>% as.data.frame()
> average_Kms
                   Car_Name   mean_kms
1                       800 127000.000
2                 Activa 3g 250250.000
3                 Activa 4g   1300.000
4             Bajaj  ct 100  35000.000
5         Bajaj Avenger 150   7000.000
6  Bajaj Avenger 150 street  20000.000
7         Bajaj Avenger 220   1766.667
8    Bajaj Avenger 220 dtsi  21800.000
9  Bajaj Avenger Street 220  24000.000
10       Bajaj Discover 100  21000.000
11       Bajaj Discover 125  30000.000
12        Bajaj Dominar 400   1200.000
13      Bajaj Pulsar  NS 200  25000.000
14       Bajaj Pulsar 135 LS  13700.000
```

Fig 4: Output of Q4

Q5. What is the average Selling_Price of the cars in each year?

A5: We are using the same package “dplyr” in this problem. We use the same summarize(), group_by() and mean() functions to find the average selling price of the cars in each year. Fig 5 shows the output of the average selling price of the cars in each year.

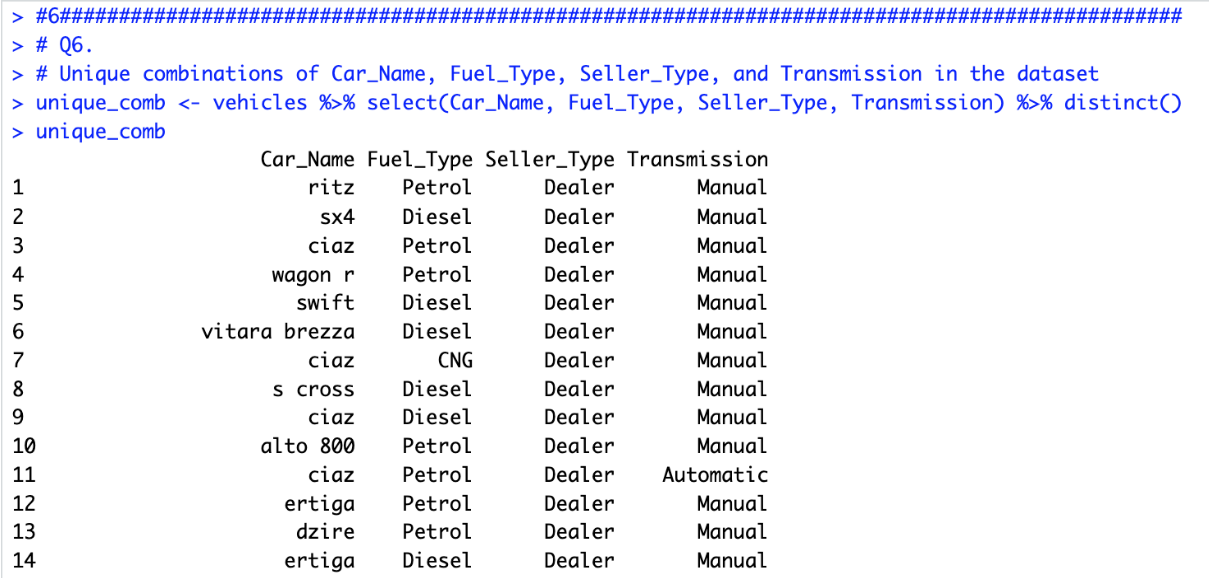


```
> #5###############################################################################################
> # Q5.
> # Average Selling_Price of the cars in each year
> average_SP <- vehicles %>% group_by(Year) %>% summarise(mean_sp = mean(Selling_Price)) %>% as.data.frame()
> average_SP
   Year  mean_sp
1  2003 1.300000
2  2004 1.500000
3  2005 2.487500
4  2006 1.437500
5  2007 0.160000
6  2008 1.002857
7  2009 2.816667
8  2010 5.262667
9  2011 2.375263
10 2012 3.841304
11 2013 3.540909
12 2014 4.762105
13 2015 5.927049
14 2016 5.213200
15 2017 6.209143
16 2018 9.250000
> 
```

Fig 12: Output of Q5

Q6. Show the unique combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the Vehicle dataset.

A6: Using the "dplyr" library, we can use the functions select() to select the columns with their respective column names and with the help of distinct() we can get all the unique combinations. Fig 6 shows the output for Q6.

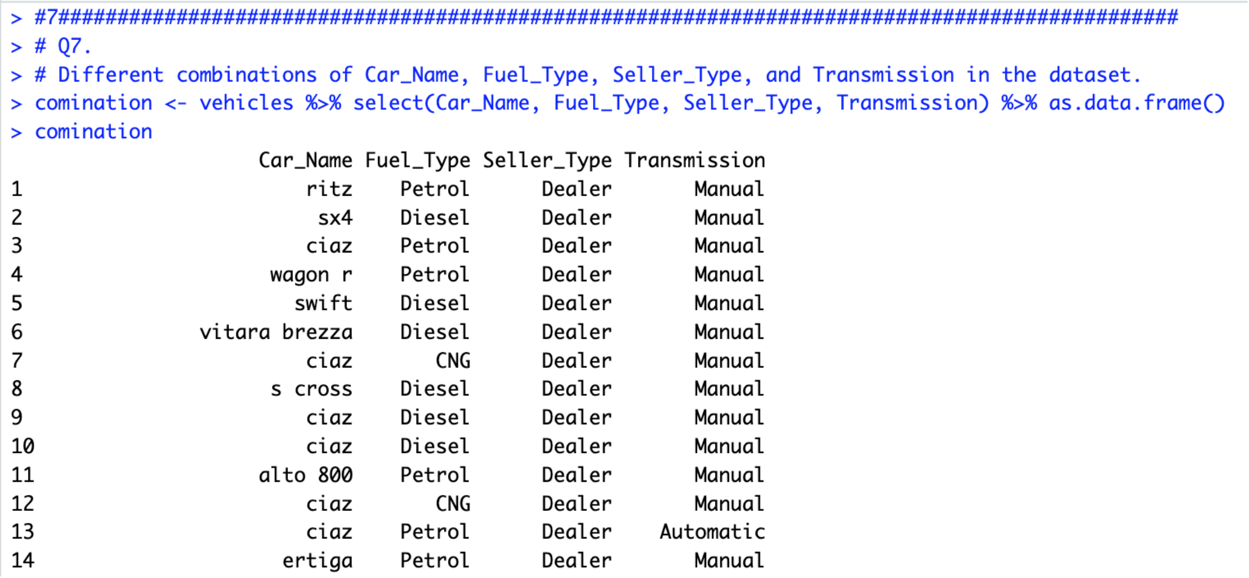


```
> #6###############################################################################################
> # Q6.
> # Unique combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the dataset
> unique_comb <- vehicles %>% select(Car_Name, Fuel_Type, Seller_Type, Transmission) %>% distinct()
> unique_comb
                     Car_Name Fuel_Type Seller_Type Transmission
1                        ritz    Petrol      Dealer       Manual
2                         sx4    Diesel      Dealer       Manual
3                        ciaz    Petrol      Dealer       Manual
4                     wagon r    Petrol      Dealer       Manual
5                       swift    Diesel      Dealer       Manual
6               vitara brezza    Diesel      Dealer       Manual
7                        ciaz       CNG      Dealer       Manual
8                     s cross    Diesel      Dealer       Manual
9                        ciaz    Diesel      Dealer       Manual
10                   alto 800    Petrol      Dealer       Manual
11                       ciaz    Petrol      Dealer    Automatic
12                     ertiga    Petrol      Dealer       Manual
13                      dzire    Petrol      Dealer       Manual
14                     ertiga    Diesel      Dealer       Manual
```

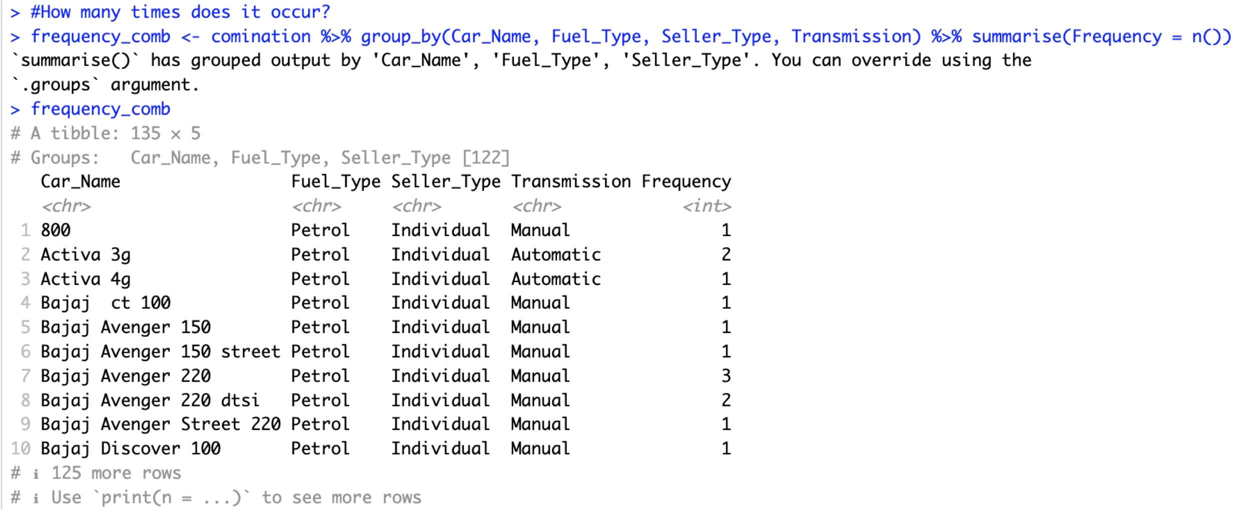
Fig 6. The output of Q6

Q7. What are the different combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the Vehicle dataset, and how many times does it occur? (Display all such in both ascending and descending orders).

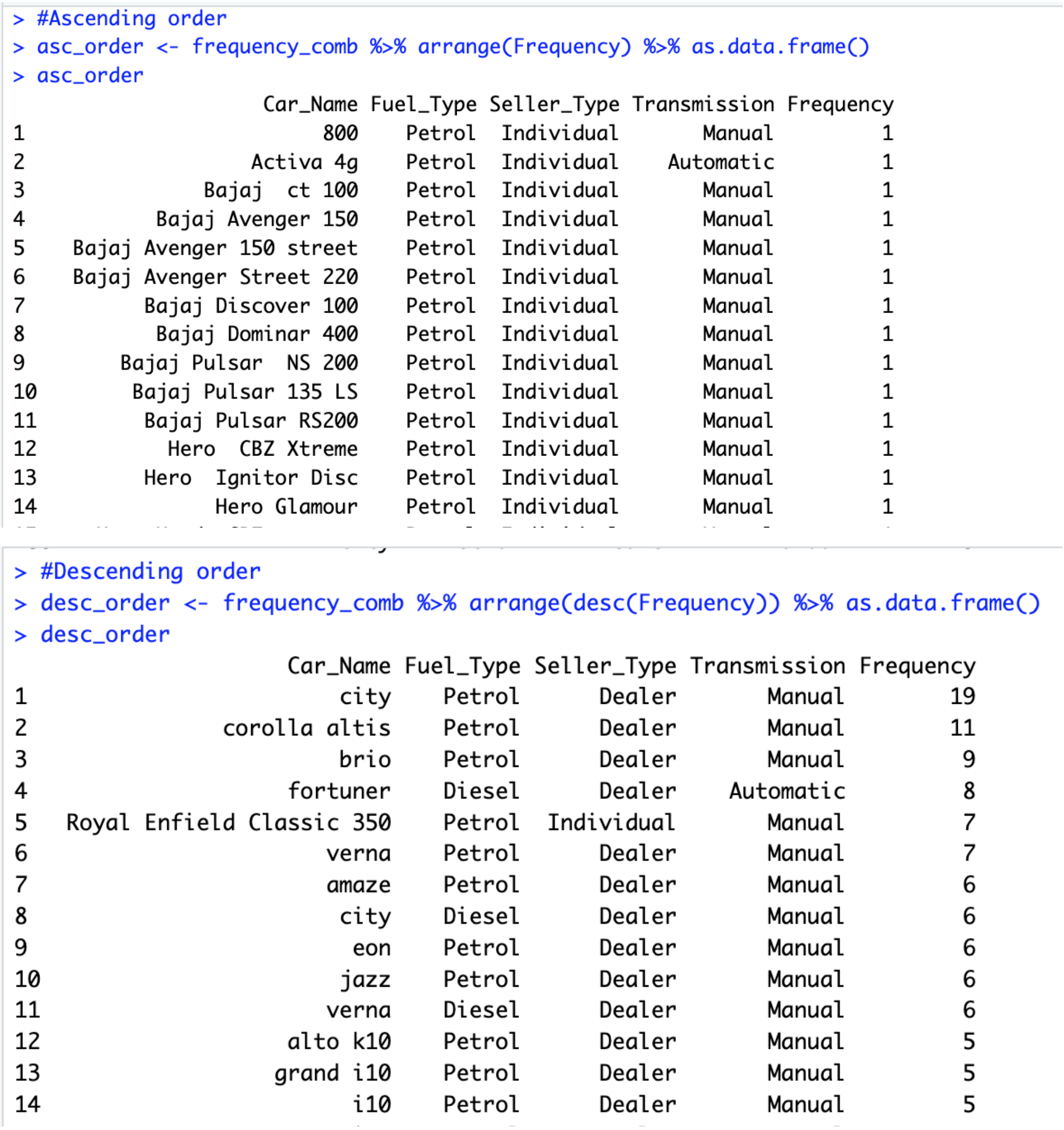
A7: Using the “dplyr” library, we can use the select() function to select the columns and display all the different combinations. We can use the group_by() and summarize() functions along with the n() function to calculate the frequency of the different combinations. We can use the arrange() function to display the output in ascending order and a desc() inside the arrange() function to display the outputs in decreasing order of frequency as shown in Fig 7. The outputs of Q7 are shown in Fig 7.



```
> #7###############################################################################################
> # Q7.
> # Different combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the dataset.
> comination <- vehicles %>% select(Car_Name, Fuel_Type, Seller_Type, Transmission) %>% as.data.frame()
> comination
                     Car_Name Fuel_Type Seller_Type Transmission
1                        ritz    Petrol      Dealer       Manual
2                         sx4    Diesel      Dealer       Manual
3                        ciaz    Petrol      Dealer       Manual
4                     wagon r    Petrol      Dealer       Manual
5                       swift    Diesel      Dealer       Manual
6               vitara brezza    Diesel      Dealer       Manual
7                        ciaz       CNG      Dealer       Manual
8                     s cross    Diesel      Dealer       Manual
9                        ciaz    Diesel      Dealer       Manual
10                       ciaz    Diesel      Dealer       Manual
11                   alto 800    Petrol      Dealer       Manual
12                       ciaz       CNG      Dealer       Manual
13                       ciaz    Petrol      Dealer    Automatic
14                     ertiga    Petrol      Dealer       Manual
```



```
> #How many times does it occur?
> frequency_comb <- comination %>% group_by(Car_Name, Fuel_Type, Seller_Type, Transmission) %>% summarise(Frequency = n())
`summarise()` has grouped output by 'Car_Name', 'Fuel_Type', 'Seller_Type'. You can override using the
`.groups` argument.
> frequency_comb
# A tibble: 135 × 5
# Groups:   Car_Name, Fuel_Type, Seller_Type [122]
   Car_Name                 Fuel_Type Seller_Type Transmission Frequency
   <chr>                    <chr>     <chr>       <chr>            <int>
 1 800                      Petrol    Individual  Manual               1
 2 Activa 3g                Petrol    Individual  Automatic            2
 3 Activa 4g                Petrol    Individual  Automatic            1
 4 Bajaj  ct 100            Petrol    Individual  Manual               1
 5 Bajaj Avenger 150        Petrol    Individual  Manual               1
 6 Bajaj Avenger 150 street Petrol    Individual  Manual               1
 7 Bajaj Avenger 220        Petrol    Individual  Manual               3
 8 Bajaj Avenger 220 dtsi   Petrol    Individual  Manual               2
 9 Bajaj Avenger Street 220 Petrol    Individual  Manual               1
10 Bajaj Discover 100       Petrol    Individual  Manual               1
# ℹ 125 more rows
# ℹ Use `print(n = ...)` to see more rows
```



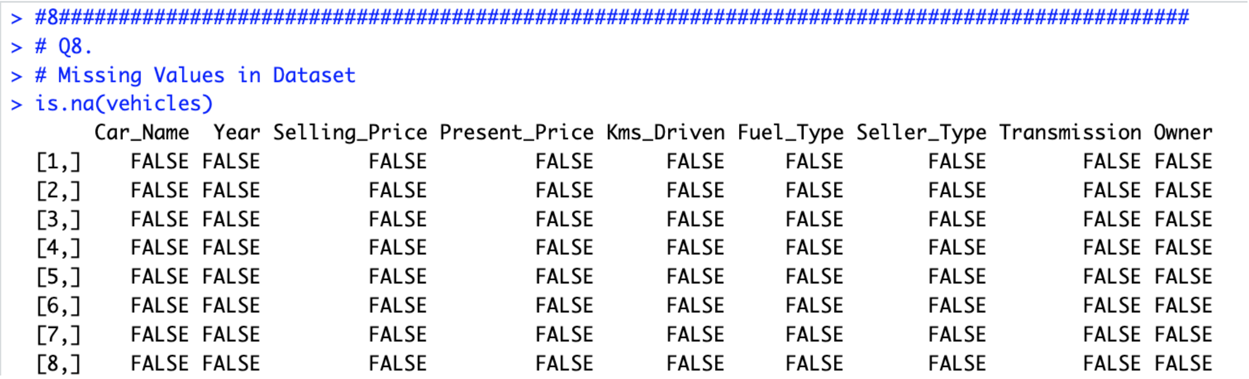
```
> #Ascending order
> asc_order <- frequency_comb %>% arrange(Frequency) %>% as.data.frame()
> asc_order
                   Car_Name Fuel_Type Seller_Type Transmission Frequency
1                       800    Petrol  Individual       Manual         1
2                 Activa 4g    Petrol  Individual    Automatic         1
3               Bajaj  ct 100  Petrol  Individual       Manual         1
4           Bajaj Avenger 150  Petrol  Individual       Manual         1
5    Bajaj Avenger 150 street  Petrol  Individual       Manual         1
6    Bajaj Avenger Street 220  Petrol  Individual       Manual         1
7          Bajaj Discover 100  Petrol  Individual       Manual         1
8           Bajaj Dominar 400  Petrol  Individual       Manual         1
9        Bajaj Pulsar  NS 200  Petrol  Individual       Manual         1
10         Bajaj Pulsar 135 LS Petrol  Individual       Manual         1
11         Bajaj Pulsar RS200  Petrol  Individual       Manual         1
12           Hero  CBZ Xtreme  Petrol  Individual       Manual         1
13         Hero  Ignitor Disc  Petrol  Individual       Manual         1
14               Hero Glamour  Petrol  Individual       Manual         1
```

```
> #Descending order
> desc_order <- frequency_comb %>% arrange(desc(Frequency)) %>% as.data.frame()
> desc_order
                   Car_Name Fuel_Type Seller_Type Transmission Frequency
1                      city    Petrol      Dealer       Manual        19
2             corolla altis    Petrol      Dealer       Manual        11
3                      brio    Petrol      Dealer       Manual         9
4                  fortuner    Diesel      Dealer    Automatic         8
5   Royal Enfield Classic 350  Petrol  Individual       Manual         7
6                     verna    Petrol      Dealer       Manual         7
7                     amaze    Petrol      Dealer       Manual         6
8                      city    Diesel      Dealer       Manual         6
9                       eon    Petrol      Dealer       Manual         6
10                     jazz    Petrol      Dealer       Manual         6
11                    verna    Diesel      Dealer       Manual         6
12                 alto k10    Petrol      Dealer       Manual         5
13                grand i10    Petrol      Dealer       Manual         5
14                      i10    Petrol      Dealer       Manual         5
```

Fig 7. The output of Q7

Q8. Find if there are any missing values in the Vehicle dataset.

A8: We can use the is.na() function to find if there are any blank or null or missing values in the dataset. However, the vehicle dataset is completely filled and does not contain any missing values. The output of Q8 is as shown in Fig 8.

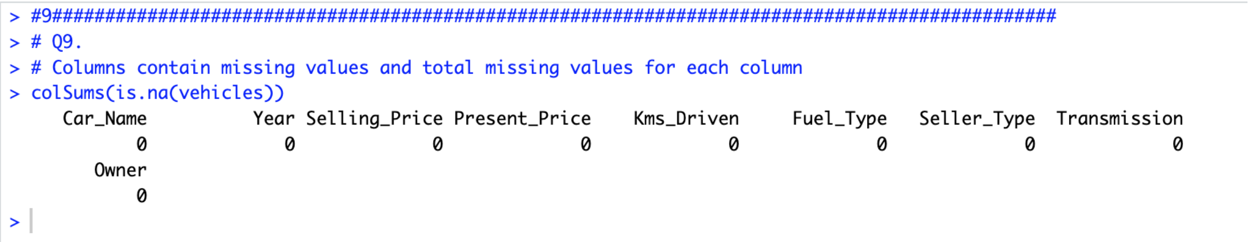


```
> #8################################################################################################
> # Q8.
> # Missing Values in Dataset
> is.na(vehicles)
      Car_Name  Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner
 [1,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [2,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [3,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [4,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [5,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [6,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [7,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
 [8,]    FALSE FALSE         FALSE         FALSE      FALSE     FALSE       FALSE        FALSE FALSE
```

Fig 8. The output of Q8.

Q9. Find which columns contain missing values in the vehicle dataset. What are the total missing values for each column?

A9: We can use the is.na() function inside the colSums() function to get the number of missing or blank values in each column of the dataset. However, in the vehicle dataset we don't have any missing values, hence the result is 0 in all the columns.

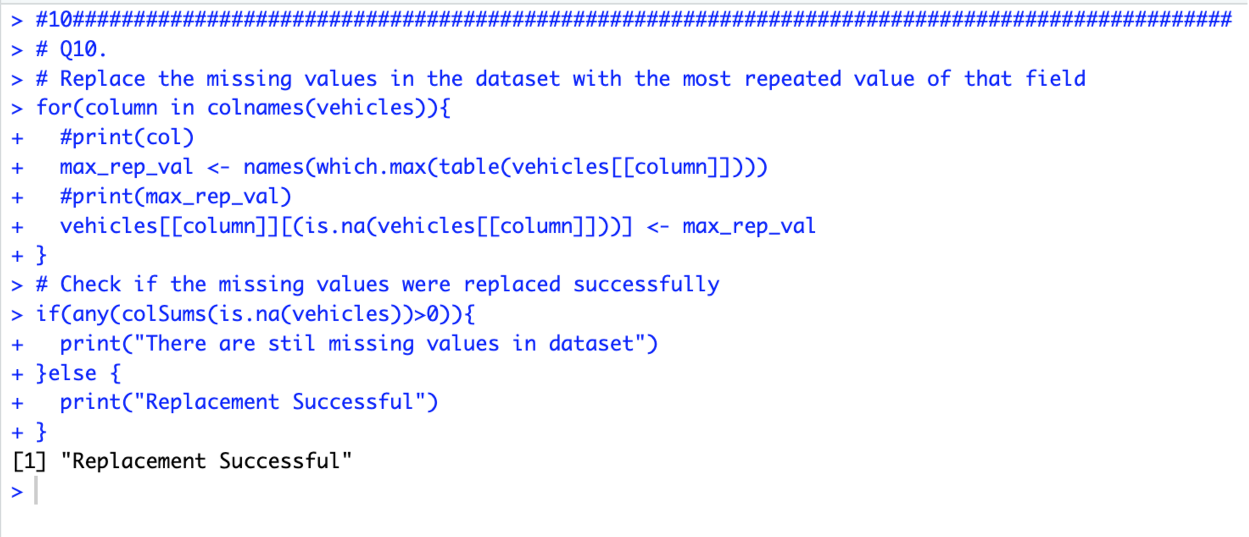


```
> #9################################################################################################
> # Q9.
> # Columns contain missing values and total missing values for each column
> colSums(is.na(vehicles))
     Car_Name          Year Selling_Price Present_Price    Kms_Driven     Fuel_Type   Seller_Type  Transmission 
            0             0             0             0             0             0             0             0 
        Owner 
            0 
> 
```

Fig 9. The output of Q9.

Q10. Replace the missing values in the dataset with the most repeated value of that field. Check if the missing values were replaced successfully.

A10: The can use a for loop to iterate through all the columns in the dataset and replace the missing values if found. The most repeated value in a column can be calculated using the names() and which.max() functions provide the value that has been repeated the most in a particular column. In this case, since our dataset does not have any missing values, no values are replaced. We can use a simple if else condition statement with the functions used in the previous question (Q9) to check if there are any more missing values in the dataset as shown in Fig 10. The outputs obtained are as shown in Fig 10.

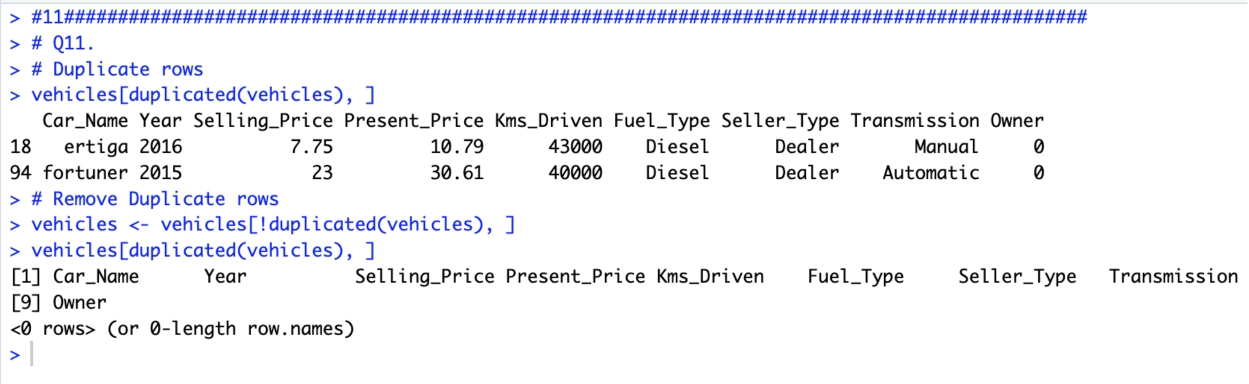


```
> #10##################################################################################################
> # Q10.
> # Replace the missing values in the dataset with the most repeated value of that field
> for(column in colnames(vehicles)){
+   #print(col)
+   max_rep_val <- names(which.max(table(vehicles[[column]])))
+   #print(max_rep_val)
+   vehicles[[column]][(is.na(vehicles[[column]]))] <- max_rep_val
+ }
> # Check if the missing values were replaced successfully
> if(any(colSums(is.na(vehicles))>0)){
+   print("There are stil missing values in dataset")
+ }else {
+   print("Replacement Successful")
+ }
[1] "Replacement Successful"
> 
```

Fig 10. The output of Q10.

Q11. Find if the dataset has duplicate rows. Remove them, if exist.

A11: We can check the presence of duplicate rows using the duplicated() function in R. It returns all the rows that are the duplicates of each other. We can use the !duplicated() function to remove these duplicate values and keep only one element as unique in the dataset. Fig 11 shows the output of Q11.



```
> #11##############################################################################################
> # Q11.
> # Duplicate rows
> vehicles[duplicated(vehicles), ]
   Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner
18   ertiga 2016          7.75         10.79      43000    Diesel      Dealer       Manual     0
94 fortuner 2015            23         30.61      40000    Diesel      Dealer    Automatic     0
> # Remove Duplicate rows
> vehicles <- vehicles[!duplicated(vehicles), ]
> vehicles[duplicated(vehicles), ]
[1] Car_Name      Year          Selling_Price Present_Price Kms_Driven    Fuel_Type     Seller_Type   Transmission 
[9] Owner        
<0 rows> (or 0-length row.names)
> 
```

Fig 11. The output of Q11.

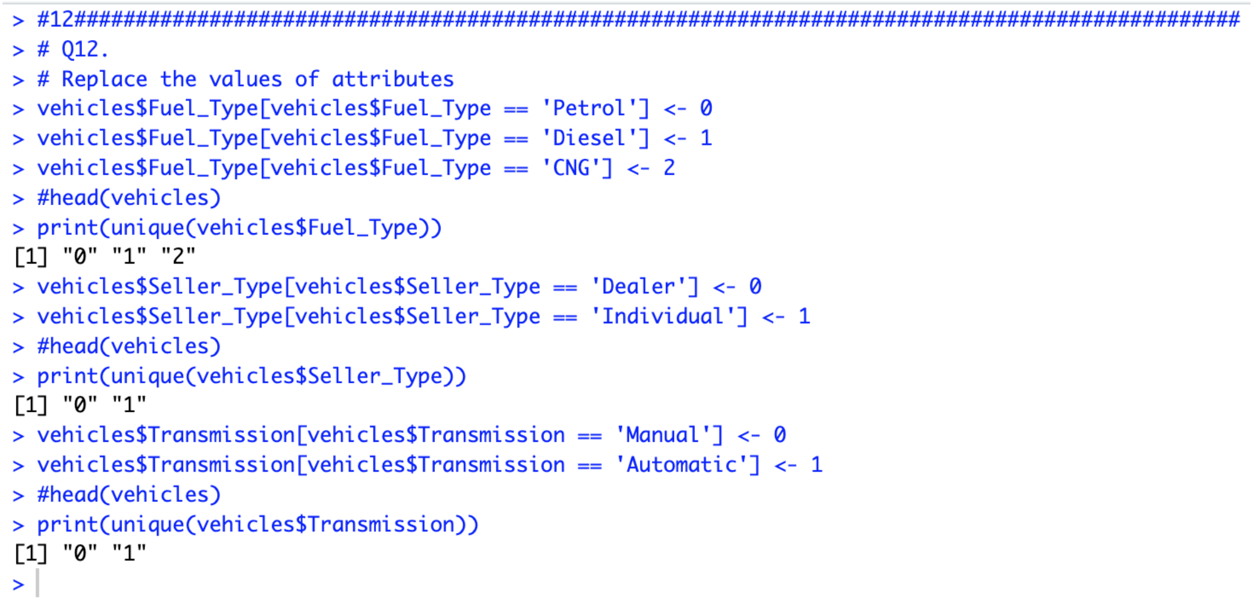
Q12. Replace the values of the following attributes:

a. Fuel_Type: “Petrol”: 0, “Diesel”: 1, “CNG”: 2
b. Seller_Type: “Dealer”: 0, “Individual”: 1

c. Transmission: “Manual”: 0, “Automatic”: 1

Show the conversion output of the specific attribute

A12: We can use the ‘==’ operator to check if the values in the column correspond to the given values and replace it with the given numeric values. To show the conversion output of the specific attribute we can use the unique() function to print all the unique values in the mentioned column. Fig 12 shows the outputs of Q12.

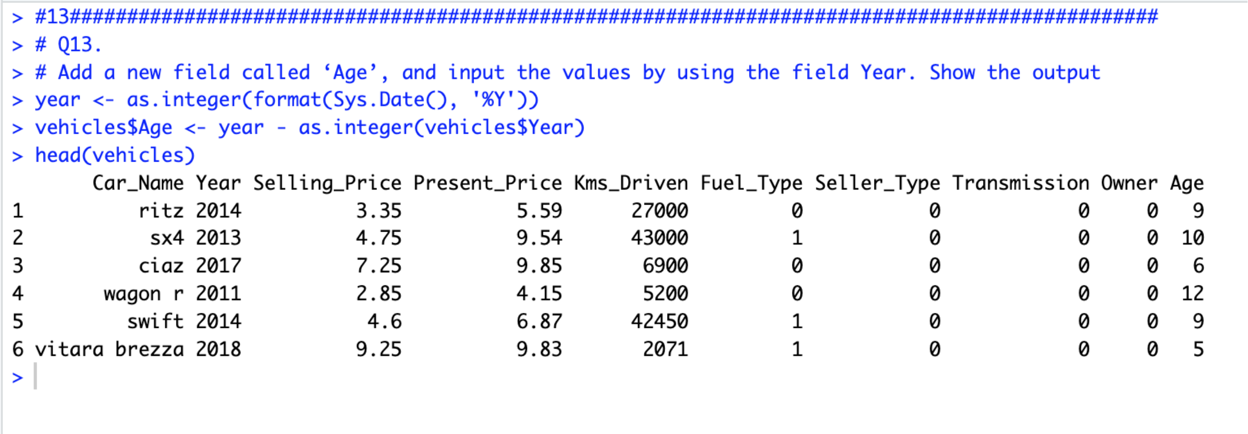


```
> #12###############################################################################################
> # Q12.
> # Replace the values of attributes
> vehicles$Fuel_Type[vehicles$Fuel_Type == 'Petrol'] <- 0
> vehicles$Fuel_Type[vehicles$Fuel_Type == 'Diesel'] <- 1
> vehicles$Fuel_Type[vehicles$Fuel_Type == 'CNG'] <- 2
> #head(vehicles)
> print(unique(vehicles$Fuel_Type))
[1] "0" "1" "2"
> vehicles$Seller_Type[vehicles$Seller_Type == 'Dealer'] <- 0
> vehicles$Seller_Type[vehicles$Seller_Type == 'Individual'] <- 1
> #head(vehicles)
> print(unique(vehicles$Seller_Type))
[1] "0" "1"
> vehicles$Transmission[vehicles$Transmission == 'Manual'] <- 0
> vehicles$Transmission[vehicles$Transmission == 'Automatic'] <- 1
> #head(vehicles)
> print(unique(vehicles$Transmission))
[1] "0" "1"
> 
```

Fig 12. The output of Q12.

Q13. Add a new field called ‘Age’, and input the values by using the field Year. Show the output.

A13: We can use the Sys.Date() function to find the current system data and we can use the format() function to extract the year from the system date. We can use this value and convert it into integer and subtract it from the year specified in the vehicle dataset to find the age. We can just specify the ‘dataframe_name$column_name’ to create a new column inside that dataframe. The outputs obtained are as shown in fig 13.

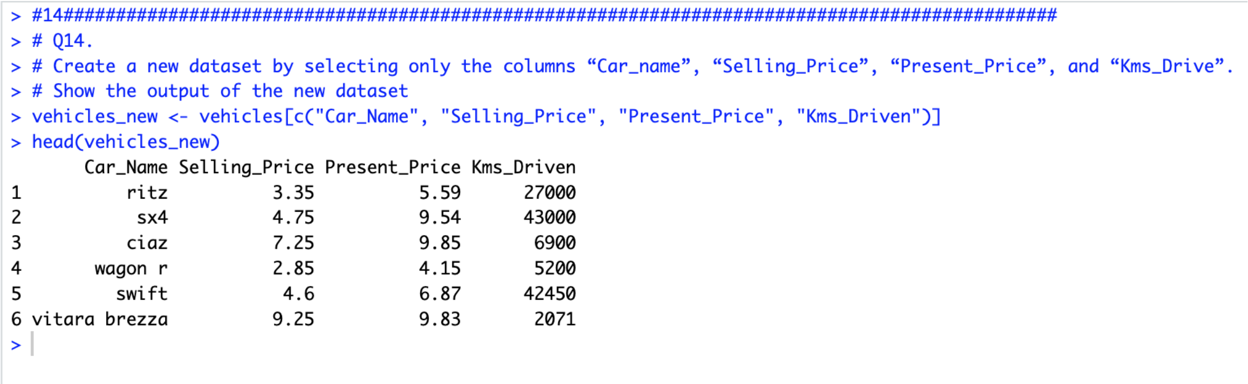


```
> #13############################################################################################
> # Q13.
> # Add a new field called ‘Age’, and input the values by using the field Year. Show the output
> year <- as.integer(format(Sys.Date(), '%Y'))
> vehicles$Age <- year - as.integer(vehicles$Year)
> head(vehicles)
       Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner Age
1          ritz 2014          3.35          5.59      27000         0           0            0     0   9
2           sx4 2013          4.75          9.54      43000         1           0            0     0  10
3          ciaz 2017          7.25          9.85       6900         0           0            0     0   6
4       wagon r 2011          2.85          4.15       5200         0           0            0     0  12
5         swift 2014           4.6          6.87      42450         1           0            0     0   9
6 vitara brezza 2018          9.25          9.83       2071         1           0            0     0   5
> 
```

Fig 13.The output of Q13.

Q14. Create a new dataset by selecting only the columns “Car_name”, “Selling_Price”, “Present_Price”, and “Kms_Drive”. Show the output of the new dataset.

A14: We can create a new dataset from the previous dataset using dataframes and c() function. The c() function combines all the specified column names and a new dataset can be created as shown in Fig 14. The output can be displayed either using the print function or the head() function which will display the first 6 rows of the dataset as shown in Fig. The outputs obtained are as shown in Fig 14.

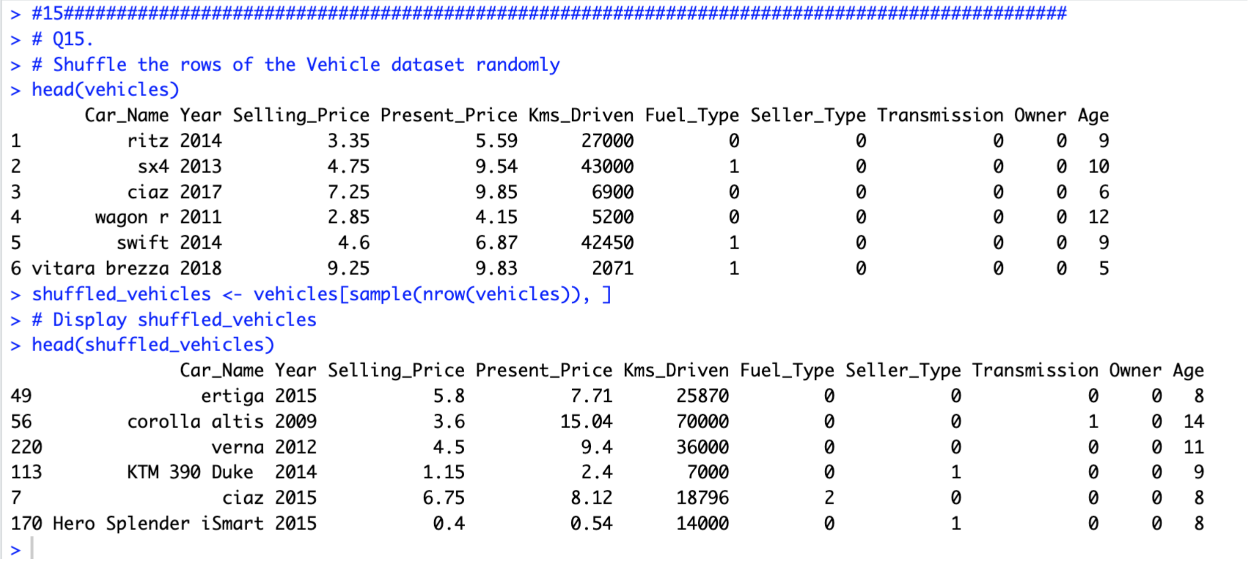


```
> #14############################################################################################
> # Q14.
> # Create a new dataset by selecting only the columns “Car_name”, “Selling_Price”, “Present_Price”, and “Kms_Drive”.
> # Show the output of the new dataset
> vehicles_new <- vehicles[c("Car_Name", "Selling_Price", "Present_Price", "Kms_Driven")]
> head(vehicles_new)
       Car_Name Selling_Price Present_Price Kms_Driven
1          ritz          3.35          5.59      27000
2           sx4          4.75          9.54      43000
3          ciaz          7.25          9.85       6900
4       wagon r          2.85          4.15       5200
5         swift           4.6          6.87      42450
6 vitara brezza          9.25          9.83       2071
> 
```

Fig 14. The output of Q14

Q15. Shuffle the rows of the Vehicle dataset randomly and show the output.

A15: The rows of the dataset can be shuffled using the sample() and nrow() functions. The shuffled output can be displayed either using the print function or the head() function which will display the first 6 rows of the dataset as shown in Fig 15. The outputs obtained are as shown in Fig 15.



```
> #15###############################################################################################
> # Q15.
> # Shuffle the rows of the Vehicle dataset randomly
> head(vehicles)
       Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner Age
1          ritz 2014          3.35          5.59      27000         0           0            0     0   9
2           sx4 2013          4.75          9.54      43000         1           0            0     0  10
3          ciaz 2017          7.25          9.85       6900         0           0            0     0   6
4       wagon r 2011          2.85          4.15       5200         0           0            0     0  12
5         swift 2014           4.6          6.87      42450         1           0            0     0   9
6 vitara brezza 2018          9.25          9.83       2071         1           0            0     0   5
> shuffled_vehicles <- vehicles[sample(nrow(vehicles)), ]
> # Display shuffled_vehicles
> head(shuffled_vehicles)
                Car_Name Year Selling_Price Present_Price Kms_Driven Fuel_Type Seller_Type Transmission Owner Age
49                ertiga 2015           5.8          7.71      25870         0           0            0     0   8
56         corolla altis 2009           3.6         15.04      70000         0           0            1     0  14
220                verna 2012           4.5           9.4      36000         0           0            0     0  11
113        KTM 390 Duke  2014          1.15           2.4       7000         0           1            0     0   9
7                   ciaz 2015          6.75          8.12      18796         2           0            0     0   8
170 Hero Splender iSmart 2015           0.4          0.54      14000         0           1            0     0   8
> 
```

Fig 15. The output of Q15.

Q16. Import the Vehicle dataset. Create a scatter plot of the Selling_Price Vs Present_Price. Color code the points based on the Transmission (5 marks).

a. Add labels, title and color to the plot. The color should be red for Transmission type '0' and blue for '1'.
b. Add open triangles to the plot.
c. What do you understand from the output (5 marks)?

A16: We can use the plot() function in R to plot a scatter plot. We can specify the columns to be given as x-axis and y-axis in this function and it will provide the plot diagram accordingly as shown in Fig 16. The xlab, ylab, main and pch arguments are used to specify the label in x-axis, label in y-axis, title of the diagram and the shape of the plot respectively. The scatter plot helps understand the selling price and current price of the different vehicles in the dataset according to their transmission type. All the plots in 'red' correspond to the selling price and present price of that vehicle who's transmission type is manual. Similarly, 'blue' corresponds to the vehicles with automatic as its transmission type. The points with red color (Manual Transmission) and blue color (Automatic Transmission) provide insights into the distribution and correlation between the two variables within each transmission category. The obtained graph is as shown in Fig 16.

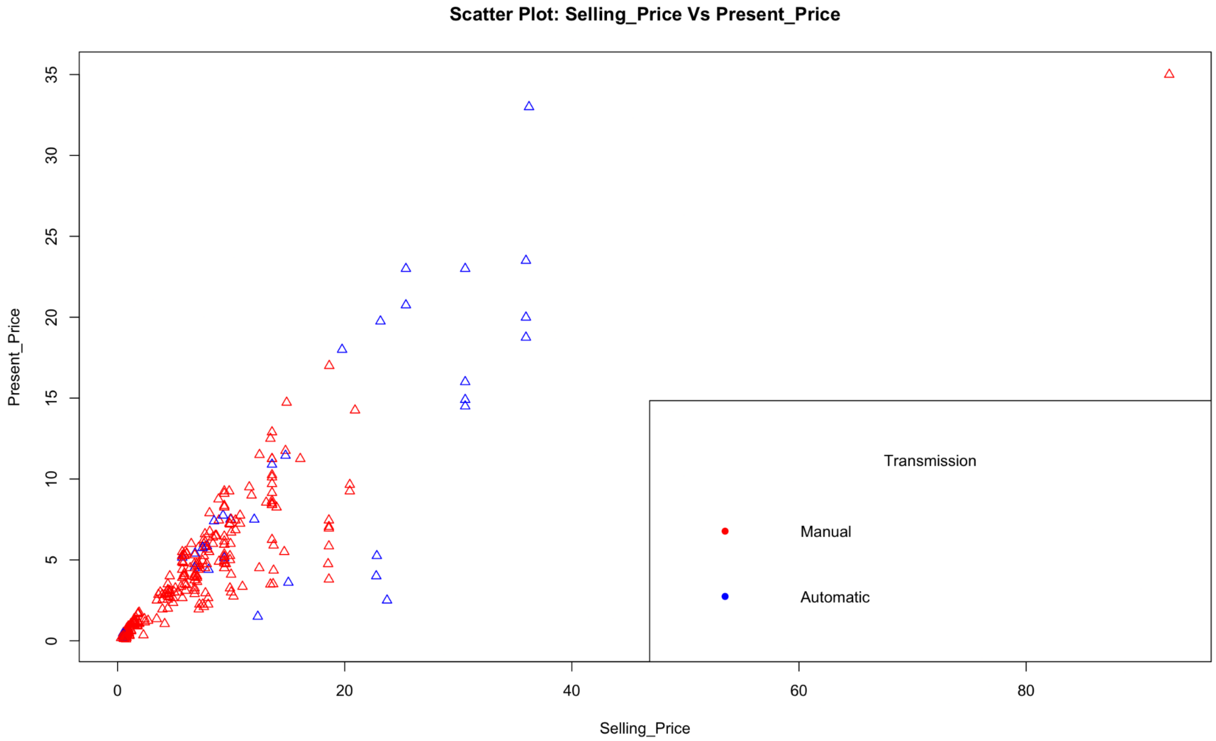


Fig 16. The output of Q16.

Q17. Create a box plot of the Selling_Price Vs Transmission and Fuel_Type.

A17: The boxplot() function in R can be used to create a boxplot diagram. In the first box plot, Selling_Price is plotted against Transmission. The formula 'Selling_Price ~ Transmission' specifies that Selling_Price is the dependent variable, and Transmission is the independent variable. The data argument is set to the 'vehicles' dataset.Similarly, in the second box plot, Selling_Price is plotted against Fuel_Type. The formula Selling_Price ~ Fuel_Type specifies the relationship between the variables. The outputs are as shown in Fig 17.

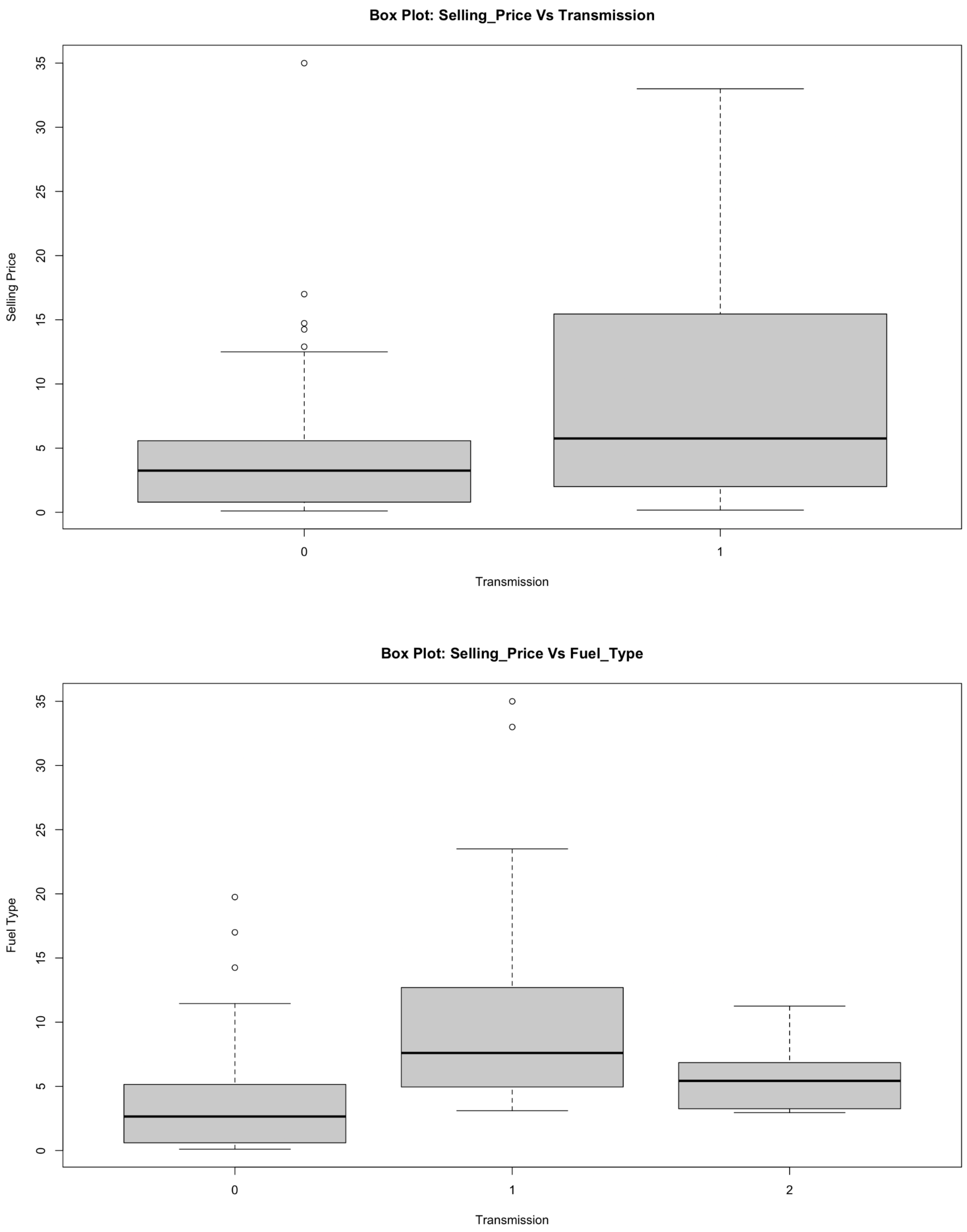


Fig 17. The output of Q17.

Q18. Create a scatter plot of the Selling_Price Vs Kms_Driven, and use k-means clustering to cluster the points into 4 clusters. Color-code based on the cluster they belong to.

A18: The kmeans() function is used to perform the k-means clustering on the selected data that contains the selling price and present price of vehicles from the dataset. We specify the center as 4 as the second argument of the kmeans() function. After the k-means clustering we use the plot() function to create the scatter plot of the resultant clustered data. This scatter plot is coloured according to the clusters (blue, red and green). We can add a legend to the diagram using the legend() function which is located at the top right of the diagram.

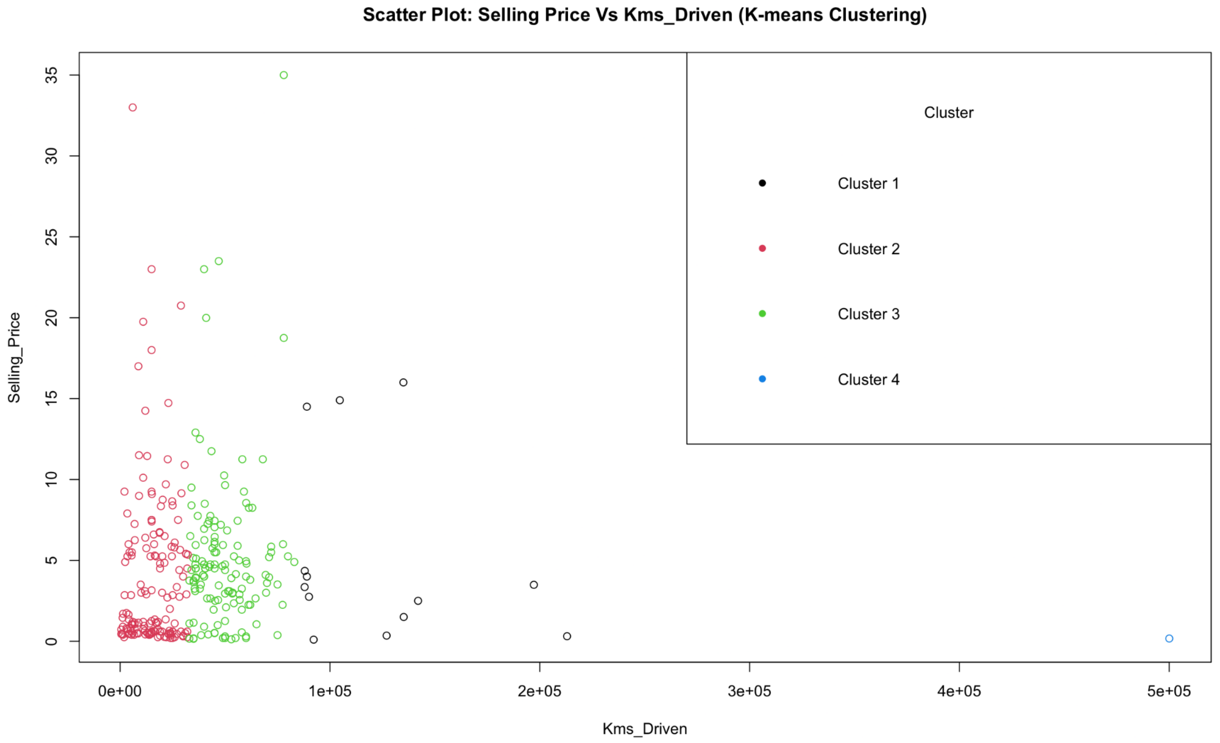


Fig 18. The output of Q18.

Q19. Create a scatter plot of the Selling_Price Vs Present_Price, and use hierarchical clustering to cluster the points into 3 clusters? Color-code the points based on the cluster they belong to.

A19: We select the two columns (selling_price and present_price) from the vehicle dataset and store it in a dataframe. We then compute the distance matrix with the dist() function, which measures the dissimilarity between points based on the euclidean distance. The hclust() function is used to perform the hierarchical clustering of the selected data. The method that we use in this clustering is 'average'. We then use the cutree() function to cut the dendrogram into 3 clusters based on a specified height which is 3. We plot this as a scatter-plot using the plot() and color them according to the cluster numbers. A legend is also added at the bottom right corner of the diagram using the legend() function.

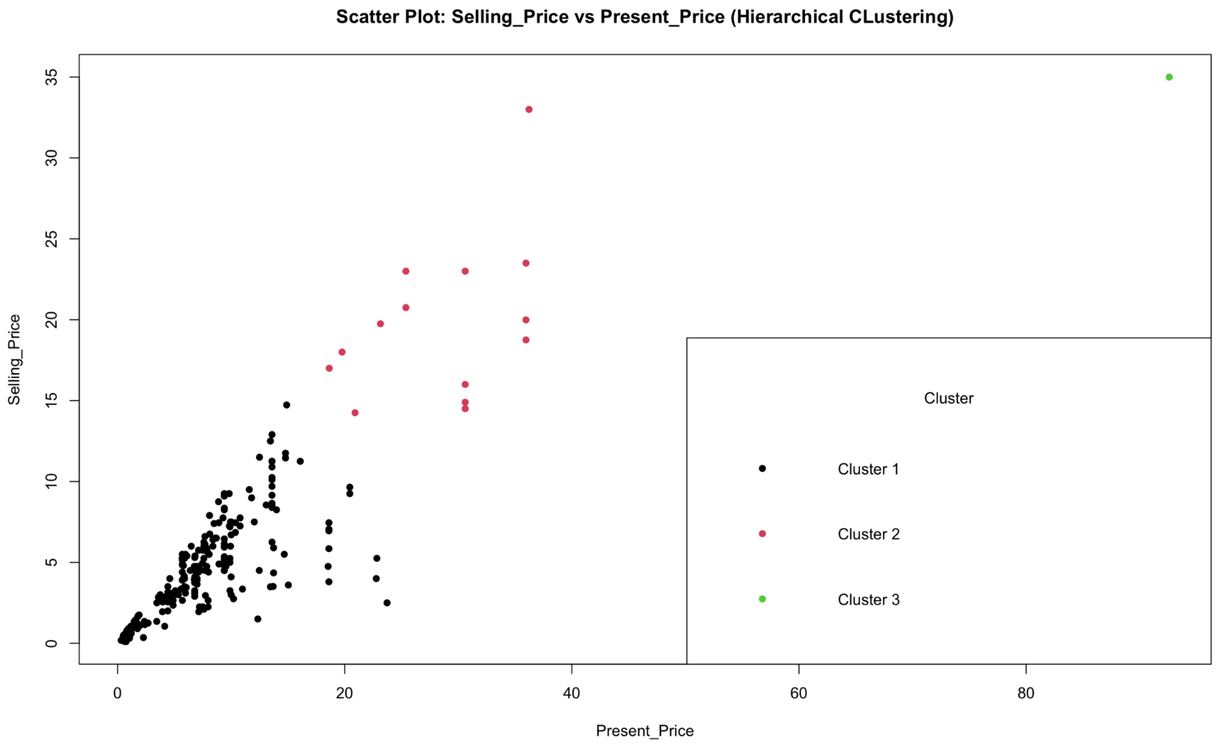
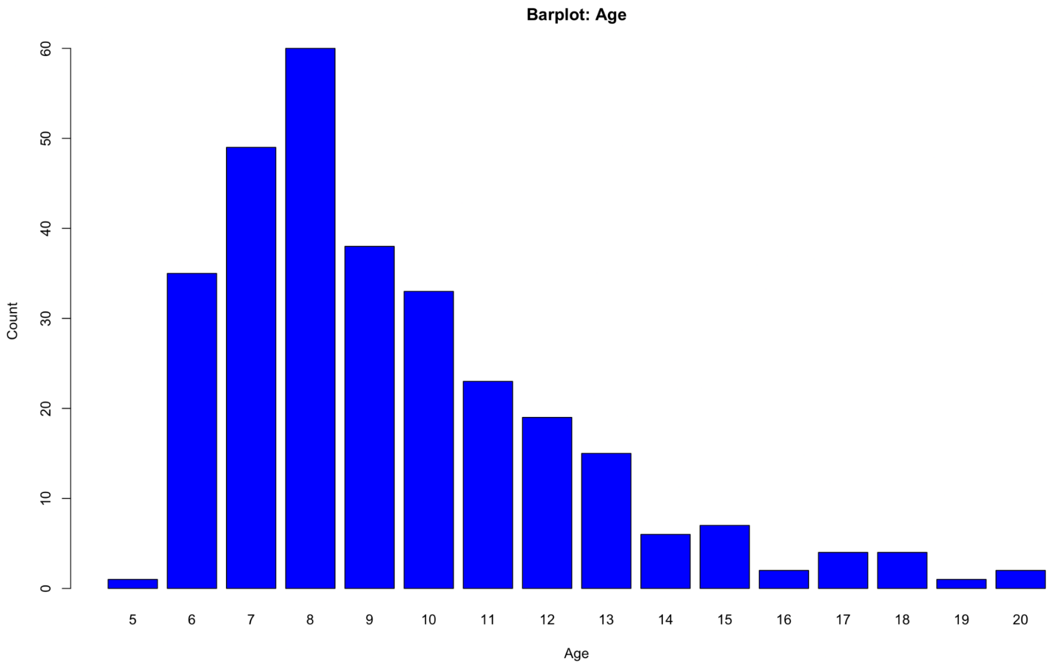


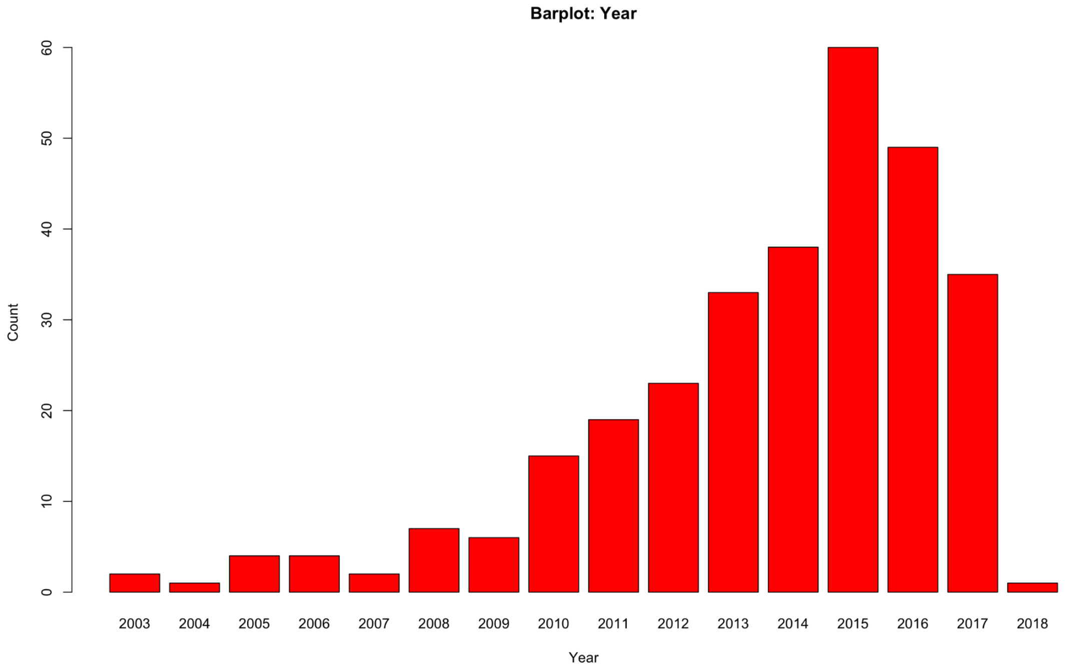
Fig 19. The output of Q19.

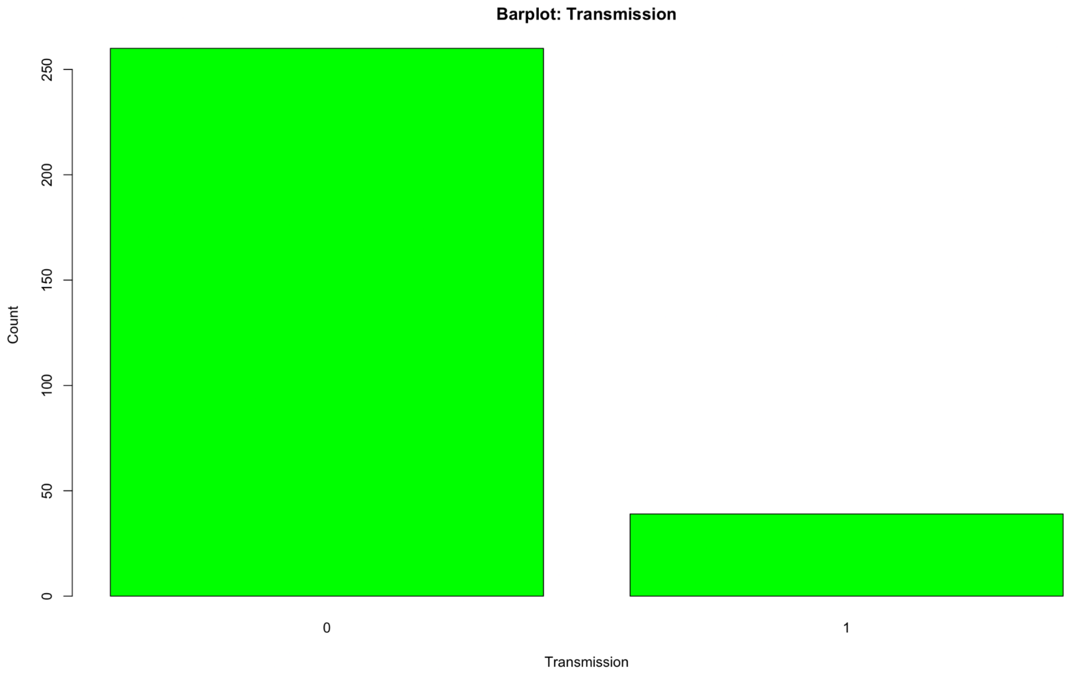
Q20. Add a new field called 'Age', and calculate it using the field 'Year'. Create a barplot for the following fields of the dataset: (10 marks)

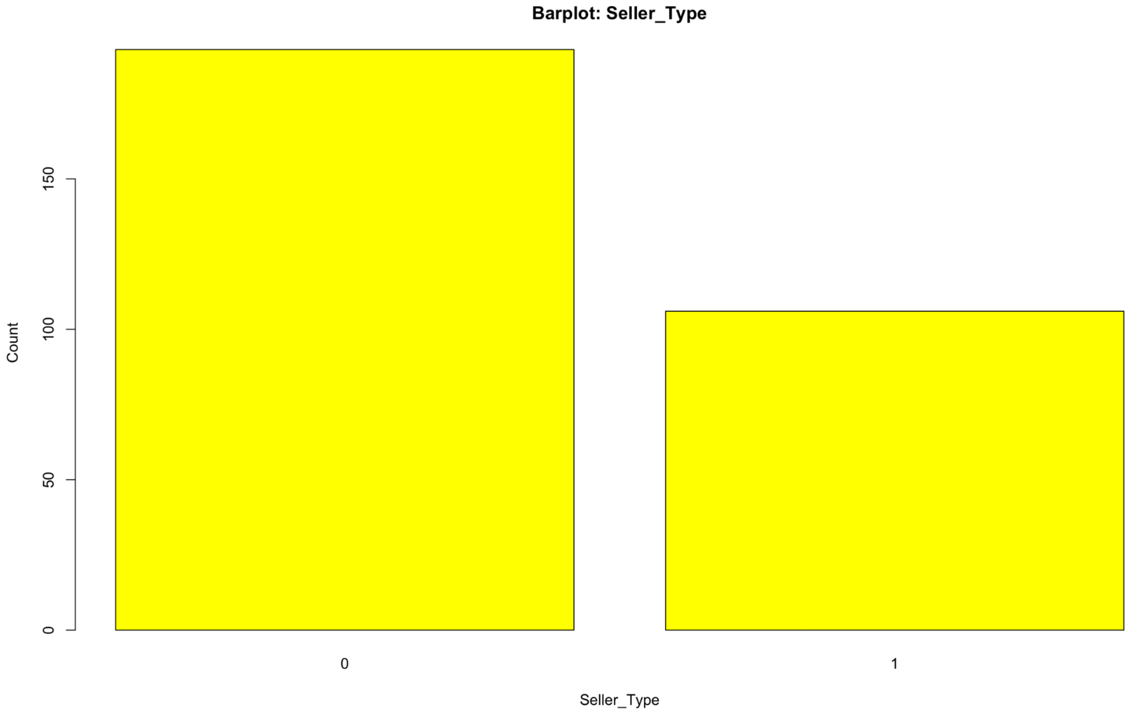
a. 'Age', 'Year', 'Transmission', 'Seller_Type', 'Fuel_Type' and 'Owner'
b. Add labels, titles, and colors to the plot.

A20: We use the Sys.Date() function to get the system date and use the format() function to get the year. We then subtract this year from the year given in the dataset and create a new column called ‘Age’. We use the barplot() to plot the graph accordingly for all the specified fields. The bar plots are also coloured accordingly in different colors.






Barplot: Transmission
Count
0
50
100
150
200
250
0
1
Transmission


Barplot: Seller_Type
Count
0
50
100
150
0
1
Seller_Type

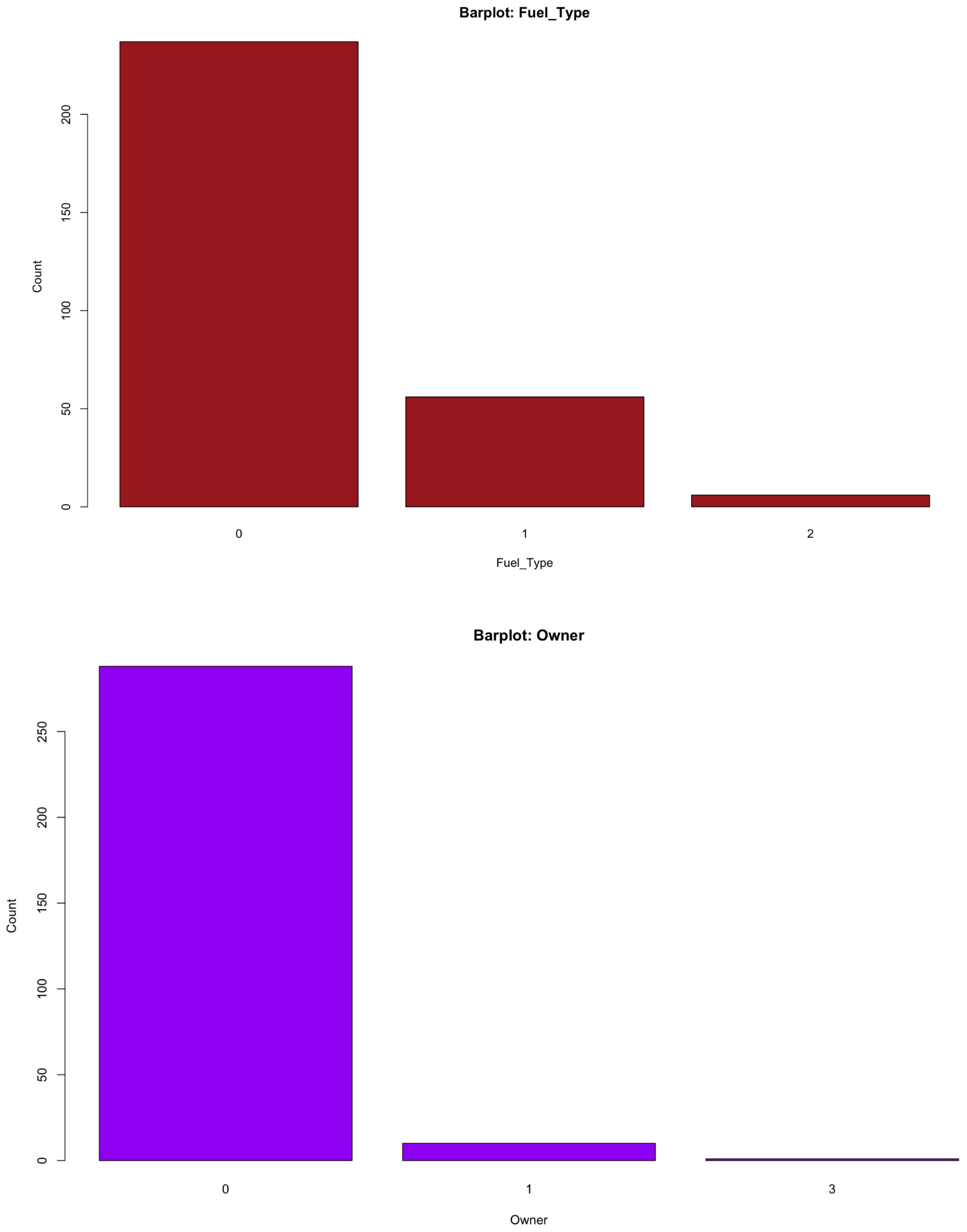


Fig 20. The output of Q20.

Q21. Create a correlation plot of the whole dataset variables and explain the output. Do not forget to convert some of the variable's data type if required and possible.

A21: First, we convert the values under the columns 'Selling_Price', 'Year', 'Present_Price', and 'Kms_Driven' into numeric data using the as.numeric() function. Now, we can plot the correlation plot using the pairs() function and plot() function. The pairs() function returns a plot matrix, consisting of scatterplots for each variable-combination of a data frame. We can plot the scatter-plots with this using the plot() function in R. The scatter-plots in a correlation plot shows the relationship between pairs of variables in the dataset. Each plot represents a pair of variables, where one variable will be on the x-axis while the other will be on the y-axis. By close examination of the correlation plot, we can identify patterns, associations, and dependencies between variables in the dataset. Here, we can see the correlation plot of the columns 'year', 'selling_price', 'present_price', and 'kms_driven' with each other. The Fig 21 shows the output graph of Q21.

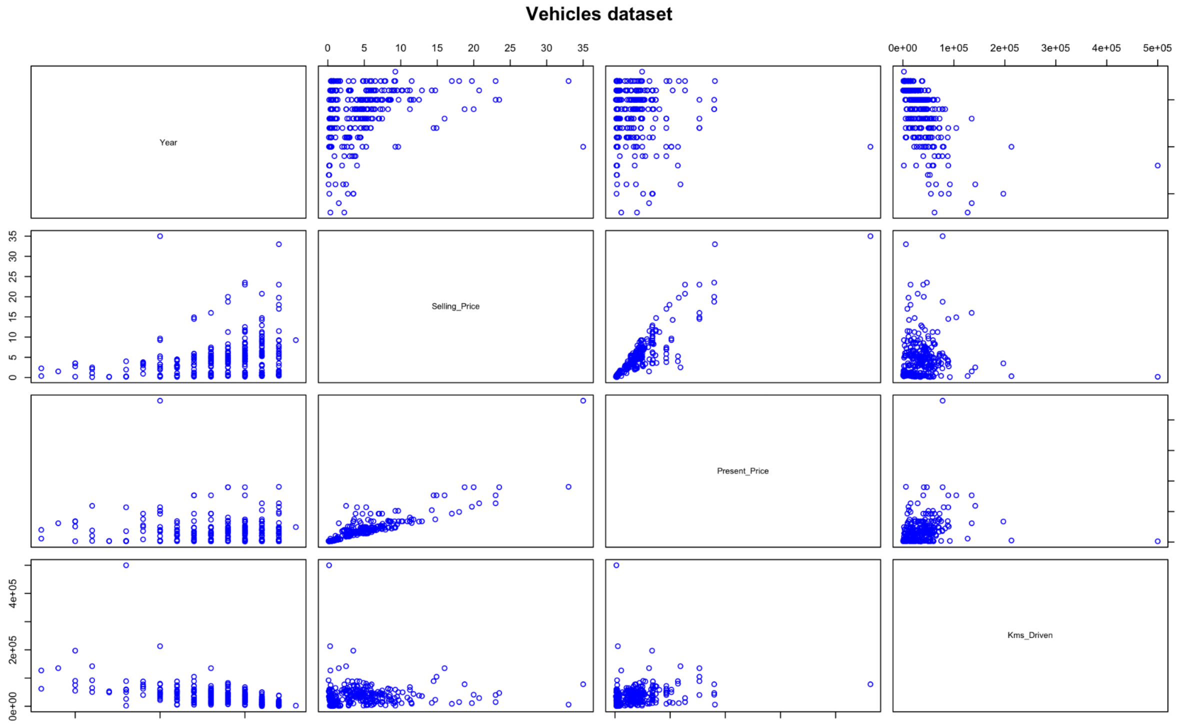


Fig 21. The output of Q21.

Q22. Create a scatter plot of the Selling_Price Vs Kms_Driven, and use DBSCAN clustering to cluster the points into 3 clusters. Color-code based on the cluster they belong to. Add a legend to the plot.

A22: The ‘dbscan’ library is installed and imported to solve this question. The Selling_price and Kms_Driven columns are extracted and clustered according to the DBSCAN algorithm. DBSCAN clustering is performed on the selected data using the dbscan() function, with specified parameters for 'eps' (the maximum distance between points) and 'MinPts' (the minimum number of points required to form a cluster). The results of the dbscan() function is used to plot the scatter-plot using the plot() function. The legend() function is used to include a legend in the plot. The legend's location is specified by the 'topright' parameter. Using the 'legend' and 'col' options, the distinct cluster labels and accompanying colors are shown in the legend. The 'pch' parameter changes the legend's point shape to a solid dot, while the 'title' argument changes the title.

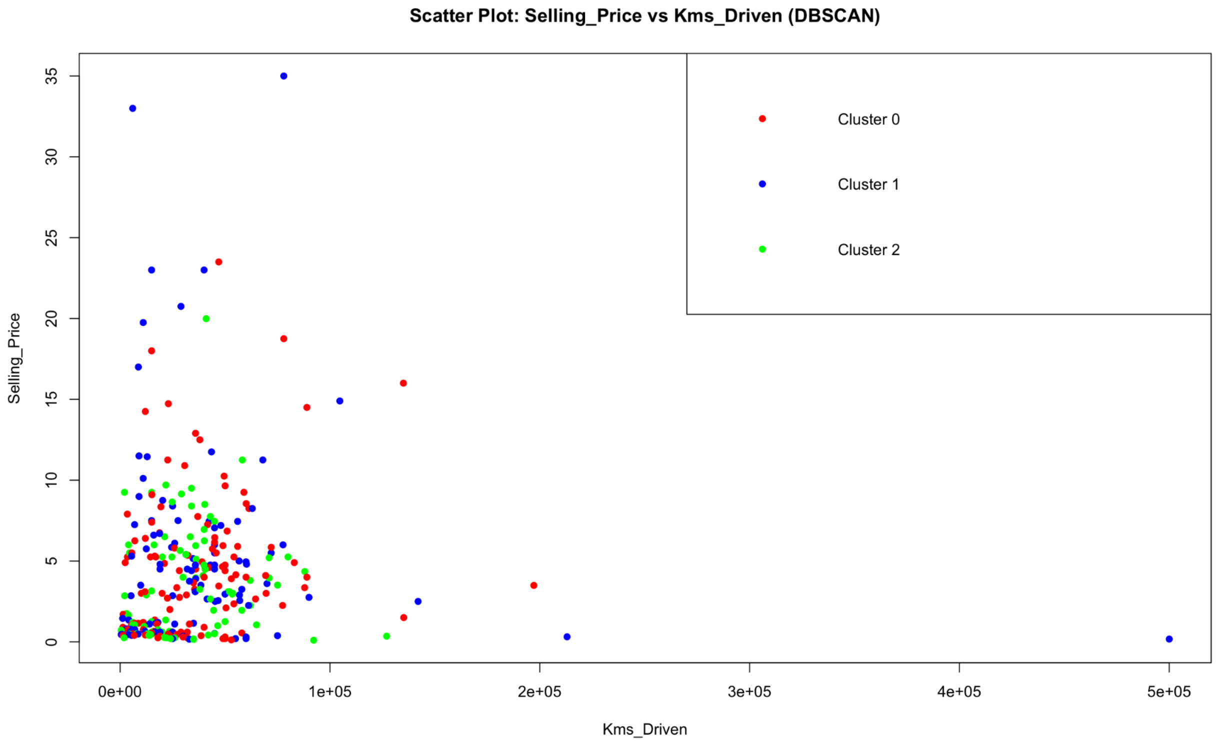


Fig 22. The output of Q22.