Business Intelligence

Coursework 2

Lukas Kopecky

4th January 2021

Q1 MySQL/R Queries

Task 1 Import data into SQL

Create table Cars:

```
> dbSendQuery(con, "
+ CREATE TABLE cars (
+ ID INT PRIMARY KEY,
+ mpg FLOAT,
+ cylinders FLOAT,
+ displacement FLOAT,
+ horsepower FLOAT,
+ weight FLOAT,
+ acceleration FLOAT,
+ model FLOAT,
+ origin FLOAT,
+ car_name VARCHAR(50),
+ price FLOAT);")
<MySQLResult:-153783080,5,90>
> dbListTables(con)
[1] "Cars"        "Order_Line" "Orders"      "Product"     "Users"
> |
```

Figure 1 – Create table ‘Cars’

Copy Data from CSV Table:

> cars_info <- read.csv("C:/Windows/Temp/cars_info.csv")

Attribute ‘horsepower’ has missing values in some records. (Those missing data are represented by Question Mark). For Purpose of this coursework, I have deleted those data entries. However, there are other methods to deal with missing data, they may be replaced by average value of the column or left as *null*. **Note:** Not all data structures accept null as a value.

Write data into the SQL table:

> dbWriteTable(con, name='Cars', value=cars_info, overwrite=TRUE)

Overview attributes:

> dbListFields(mydb, 'Cars')

[1] "ID" "mpg" "cylinders" "displacement" "horsepower" "weight" "acceleration" "model" "origin" "car_name"

[11] "price"

Display all data:

```
> data
   ID mpg cylinders displacement horsepower weight acceleration model origin                      car_name    price
1   1  18         8        307.0        130   3504         12.0    70      1     chevrolet chevelle malibu 25561.60
2   2  15         8        350.0        165   3693         11.5    70      1             buick skylark 320 24221.40
3   3  18         8        318.0        150   3436         11.0    70      1            plymouth satellite 27240.80
4   4  16         8        304.0        150   3433         12.0    70      1                 amc rebel sst 33685.00
5   5  17         8        302.0        140   3449         10.5    70      1                   ford torino 20000.00
6   6  15         8        429.0        198   4341         10.0    70      1              ford galaxie 500 30000.00
7   7  14         8        454.0        220   4354          9.0    70      1              chevrolet impala 35764.30
8   8  14         8        440.0        215   4312          8.5    70      1             plymouth fury iii 25899.50
9   9  14         8        455.0        225   4425         10.0    70      1              pontiac catalina 32882.50
10 10  15         8        390.0        190   3850          8.5    70      1            amc ambassador dpl 32617.10
11 11  15         8        383.0        170   3563         10.0    70      1           dodge challenger se 30000.00
12 12  14         8        340.0        160   3609          8.0    70      1            plymouth 'cuda 340 33034.90
13 13  15         8        400.0        150   3761          9.5    70      1           chevrolet monte carlo 30000.00
14 14  14         8        455.0        225   3086         10.0    70      1       buick estate wagon (sw) 26608.30
15 15  24         4        113.0         95   2372         15.0    70      3          toyota corona mark ii 36927.10
16 16  22         6        198.0         95   2833         15.5    70      1               plymouth duster 12561.10
17 17  18         6        199.0         97   2774         15.5    70      1                    amc hornet 24260.60
18 18  21         6        200.0         85   2587         16.0    70      1                 ford maverick 14228.30
19 19  27         4         97.0         88   2130         14.5    70      3                  datsun pl510 15802.20
20 20  26         4         97.0         46   1835         20.5    70      2  volkswagen 1131 deluxe sedan 27471.30
21 21  25         4        110.0         87   2672         17.5    70      2                   peugeot 504 40000.00
22 22  24         4        107.0         90   2430         14.5    70      2                   audi 100 ls 35897.40
23 23  25         4        104.0         95   2375         17.5    70      2                      saab 99e 16905.80
24 24  26         4        121.0        113   2234         12.5    70      2                      bmw 2002 15048.00
25 25  21         6        199.0         90   2648         15.0    70      1                   amc gremlin 14045.90
26 26  10         8        360.0        215   4615         14.0    70      1                     ford f250 25797.50
27 27  10         8        307.0        200   4376         15.0    70      1                     chevy c20 40000.00
28 28  11         8        318.0        210   4382         13.5    70      1                    dodge d200 24928.40
29 29   9         8        304.0        193   4732         18.5    70      1                      hi 1200d 45072.60
30 30  27         4         97.0         88   2130         14.5    71      3                  datsun pl510 16992.10
31 31  28         4        140.0         90   2264         15.5    71      1           chevrolet vega 2300 13206.40
32 32  25         4        113.0         95   2228         14.0    71      3                 toyota corona 27078.80
33 34  19         6        232.0        100   2634         13.0    71      1                   amc gremlin 30000.00
34 35  16         6        225.0        105   3439         15.5    71      1     plymouth satellite custom 44535.00
35 36  17         6        250.0        100   3329         15.5    71      1     chevrolet chevelle malibu 40000.00
36 37  19         6        250.0         88   3302         15.5    71      1               ford torino 500 30000.00
37 38  18         6        232.0        100   3288         15.5    71      1                   amc matador 33970.70
38 39  14         8        350.0        165   4209         12.0    71      1              chevrolet impala 40000.00
39 40  14         8        400.0        175   4464         11.5    71      1     pontiac catalina brougham 33793.70
40 41  14         8        351.0        153   4154         13.5    71      1              ford galaxie 500 30000.00
41 42  14         8        318.0        150   4096         13.0    71      1             plymouth fury iii 15358.90
42 43  12         8        383.0        180   4955         11.5    71      1            dodge monaco (sw) 23414.40
43 44  13         8        400.0        170   4746         12.0    71      1      ford country squire (sw) 44211.90
44 45  13         8        400.0        175   5140         12.0    71      1           pontiac safari (sw) 20000.00
45 46  18         6        258.0        110   2962         13.5    71      1    amc hornet sportabout (sw) 33328.40
46 47  22         4        140.0         72   2408         19.0    71      1           chevrolet vega (sw) 20000.00
```

Figure 2 – Display Imported data

```
# Display all data:
result <- dbSendQuery(mydb, 'SELECT * From Cars')

data <- fetch(result, n=-1)

data
```

Note: This screenshot covers only the first 46 rows. See the max number of rows is 46 but last ID is 47. This is caused by missing record ID 33 which was deleted as value horsepower was missing.

Task 2: RMySQL/DBI Queries

1 Get the first 10 rows in the imported table

Query: 'SELECT * FORM Cars WHERE ID <11'

```
> #Get the first 10 rows in the imported table
> result <- dbSendQuery(con, 'SELECT * From Cars WHERE ID < 11')
> data <- fetch(result, n=-1)
> data
   ID mpg cylinders displacement horsepower weight acceleration model origin                  car_name   price
1   1  18         8          307        130   3504         12.0    70      1 chevrolet chevelle malibu 25561.6
2   2  15         8          350        165   3693         11.5    70      1         buick skylark 320 24221.4
3   3  18         8          318        150   3436         11.0    70      1        plymouth satellite 27240.8
4   4  16         8          304        150   3433         12.0    70      1             amc rebel sst 33685.0
5   5  17         8          302        140   3449         10.5    70      1               ford torino 20000.0
6   6  15         8          429        198   4341         10.0    70      1          ford galaxie 500 30000.0
7   7  14         8          454        220   4354          9.0    70      1          chevrolet impala 35764.3
8   8  14         8          440        215   4312          8.5    70      1         plymouth fury iii 25899.5
9   9  14         8          455        225   4425         10.0    70      1          pontiac catalina 32882.5
10 10  15         8          390        190   3850          8.5    70      1        amc ambassador dpl 32617.1
> 
```

Figure 3 – The first 10 rows in the imported table

NOTE: This query is only valid if Primary Key (ID) is ascending.

2 Get all eight-cylinder cars with miles per gallon greater than 18

Query: 'SELECT * From Cars WHERE cylinders = 8 AND mpg >18'

```
> #Get all eight-cylinder cars with miles per gallon greater than 18
> result <- dbSendQuery(con, 'SELECT * From Cars WHERE cylinders = 8 AND
+                       mpg > 18')
> data <- fetch(result, n=-1)
> data
    ID  mpg cylinders displacement horsepower weight acceleration model origin                        car_name    price
1  166 20.0         8          262        110   3221         13.5    75      1             chevrolet monza 2+2 30000.00
2  250 19.9         8          260        110   3365         15.5    78      1    oldsmobile cutlass salon brougham 30000.00
3  251 19.4         8          318        140   3735         13.2    78      1                  dodge diplomat 40000.00
4  252 20.2         8          302        139   3570         12.8    78      1            mercury monarch ghia 34088.80
5  263 19.2         8          305        145   3425         13.2    78      1      chevrolet monte carlo landau 50024.70
6  265 18.1         8          302        139   3205         11.2    78      1                     ford futura 35375.50
7  289 18.2         8          318        135   3830         15.2    79      1                 dodge st. regis  8766.44
8  292 19.2         8          267        125   3605         15.0    79      1     chevrolet malibu classic (sw) 26157.20
9  293 18.5         8          360        150   3940         13.0    79      1 chrysler lebaron town @ country (sw) 14944.50
10 299 23.0         8          350        125   3900         17.4    79      1               cadillac eldorado 44274.40
11 301 23.9         8          260         90   3420         22.2    79      1    oldsmobile cutlass salon brougham 33062.10
12 365 26.6         8          350        105   3725         19.0    81      1             oldsmobile cutlass ls 30000.00
> 
```

Figure 4 - All eight-cylinder cars with miles per gallon greater than 18

3 Get the average horsepower and mpg by number of cylinder groups

Query: 'SELECT cylinders, AVG(horsepower), AVG(mpg) From Cars Group by cylinders'

```
> #Get the average horsepower and mpg by number of cylinder groups
> result <- dbSendQuery(con, 'SELECT cylinders, AVG(horsepower), AVG(mpg) From Cars Group by cylinders')
> data <- fetch(result, n=-1)
> data
  cylinders AVG(horsepower) AVG(mpg)
1         3        99.25000 20.55000
2         4        78.28141 29.28392
3         5        82.33333 27.36667
4         6       101.50602 19.97349
5         8       158.30097 14.96311
> 
```

Figure 5 - The average horsepower and mpg by number of cylinder groups

4 Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive)

Query: 'SELECT * From Cars WHERE cylinders < 8 AND (acceleration >= 11 and acceleration <= 13)'

```
> #Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive both limits)
> result <- dbSendQuery(con, 'SELECT * From Cars WHERE cylinders < 8 AND (acceleration >= 11 and acceleration <= 13)')
> data <- fetch(result, n=-1)
> data
    ID  mpg cylinders displacement horsepower weight acceleration model origin                car_name   price
1   24 26.0         4          121        113   2234         12.5    70      2               bmw 2002 15048.0
2   34 19.0         6          232        100   2634         13.0    71      1             amc gremlin 30000.0
3  204 29.5         4           97         71   1825         12.2    76      2      volkswagen rabbit 23640.7
4  243 21.5         4          121        110   2600         12.8    77      2               bmw 320i 23927.4
5  307 28.8         6          173        115   2595         11.3    79      1      chevrolet citation 30000.0
6  308 26.8         6          173        115   2700         12.9    79      1 oldsmobile omega brougham 45923.5
7  334 32.7         6          168        132   2910         11.4    80      3          datsun 280-zx 25944.3
8  335 23.7         3           70        100   2420         12.5    80      3           mazda rx-7 gs 44730.2
9  342 23.5         6          173        110   2725         12.6    81      1      chevrolet citation 40000.0
10 343 30.0         4          135         84   2385         12.9    81      1        plymouth reliant 33838.4
11 362 25.4         6          168        116   2900         12.6    81      3         toyota cressida 30000.0
12 392 36.0         4          135         84   2370         13.0    82      1      dodge charger 2.2 17421.6
13 396 32.0         4          135         84   2295         11.6    82      1          dodge rampage 47800.0
> 
```

Figure 6 - All cars with less than eight-cylinder and with acceleration from 11

Note: Inclusive are only value 11 and 13 of acceleration. Cars with 8 cylinders aren't included.

5 Select the car names and horsepower of the cars with 3 cylinders

Query: 'SELECT car_name, horsepower From Cars WHERE cylinders = 3'

```
> #Get the car names and horsepower of the cars with 3 cylinders
> result <- dbSendQuery(con, 'SELECT car_name, horsepower From Cars WHERE cylinders = 3 ')
> data <- fetch(result, n=-1)
> data
        car_name horsepower
1 mazda rx2 coupe         97
2       maxda rx3         90
3      mazda rx-4        110
4   mazda rx-7 gs        100
> 
```

Figure 7 - The car names and horsepower of the cars with 3 cylinders

TASK 3: dbplyr/dplyr queries

Save data structure to data frame cars_db

cars_db <- tbl(con, "Cars")

1 Get the first 10 row in the imported table

dplyr syntax: cars_db %>% filter(ID < 11)

```
> #Get the first 10 rows in the imported table
> cars_db %>% filter(ID < 11)
# Source:   lazy query [?? x 11]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
      ID   mpg cylinders displacement horsepower weight acceleration model origin car_name                  price
   <int> <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <chr>                     <dbl>
 1     1    18         8          307        130   3504         12      70      1 chevrolet chevelle malibu 25562.
 2     2    15         8          350        165   3693         11.5    70      1 buick skylark 320         24221.
 3     3    18         8          318        150   3436         11      70      1 plymouth satellite        27241.
 4     4    16         8          304        150   3433         12      70      1 amc rebel sst             33685
 5     5    17         8          302        140   3449         10.5    70      1 ford torino               20000
 6     6    15         8          429        198   4341         10      70      1 ford galaxie 500          30000
 7     7    14         8          454        220   4354          9      70      1 chevrolet impala          35764.
 8     8    14         8          440        215   4312          8.5    70      1 plymouth fury iii         25900.
 9     9    14         8          455        225   4425         10      70      1 pontiac catalina          32882.
10    10    15         8          390        190   3850          8.5    70      1 amc ambassador dpl        32617.
> 
```

Figure 8 - The first 10 row in the imported table

2 Get all eight-cylinder cars with miles per gallon greater than 18

Dplyr syntax: cars_db %>% filter(cylinders == 8, mpg > 18)

```
> #Get all eight-cylinder cars with miles per gallon greater than 18
> cars_db %>% filter(cylinders == 8, mpg > 18)
# Source:   lazy query [?? x 11]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
      ID   mpg cylinders displacement horsepower weight acceleration model origin car_name                         price
   <int> <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <chr>                            <dbl>
 1   166  20           8          262        110   3221         13.5    75      1 chevrolet monza 2+2              30000
 2   250  19.9         8          260        110   3365         15.5    78      1 oldsmobile cutlass salon brougham 30000
 3   251  19.4         8          318        140   3735         13.2    78      1 dodge diplomat                   40000
 4   252  20.2         8          302        139   3570         12.8    78      1 mercury monarch ghia             34089.
 5   263  19.2         8          305        145   3425         13.2    78      1 chevrolet monte carlo landau     50025.
 6   265  18.1         8          302        139   3205         11.2    78      1 ford futura                      35376.
 7   289  18.2         8          318        135   3830         15.2    79      1 dodge st. regis                   8766.
 8   292  19.2         8          267        125   3605         15      79      1 chevrolet malibu classic (sw)    26157.
 9   293  18.5         8          360        150   3940         13      79      1 chrysler lebaron town @ country (sw) 14944.
10   299  23           8          350        125   3900         17.4    79      1 cadillac eldorado                44274.
11   301  23.9         8          260         90   3420         22.2    79      1 oldsmobile cutlass salon brougham 33062.
12   365  26.6         8          350        105   3725         19      81      1 oldsmobile cutlass ls            30000
> 
```

Figure 9 – All eight-cylinder cars with miles per gallon greater than 18

3 Get the average horsepower and mpg by number of cylinder groups

Dplyr syntax: cars_db %>% group_by(cylinders) %>% summarise(avg(mpg), avg(horsepower))

```
> #Get the average horsepower and mpg by number of cylinder groups
> cars_db %>% group_by(cylinders) %>% summarise(avg(mpg), avg(horsepower))
# Source:   lazy query [?? x 3]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
  cylinders `avg(mpg)` `avg(horsepower)`
      <dbl>      <dbl>             <dbl>
1         3       20.6              99.2
2         4       29.3              78.3
3         5       27.4              82.3
4         6       20.0             102. 
5         8       15.0             158. 
> 
```

Figure 10 - The average horsepower and mpg by number of cylinder groups

4 Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive)

Dplyr syntax: cars_db %>% filter(cylinders < 8, acceleration <= 13, acceleration >=11)

```
> #Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive both limits)
> cars_db %>% filter(cylinders < 8, acceleration <= 13, acceleration >=11 )
# Source:   lazy query [?? x 11]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
      ID   mpg cylinders displacement horsepower weight acceleration model origin car_name                   price
   <int> <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <chr>                      <dbl>
 1    24  26           4          121        113   2234         12.5    70      2 bmw 2002                  15048
 2    34  19           6          232        100   2634         13      71      1 amc gremlin               30000
 3   204  29.5         4           97         71   1825         12.2    76      2 volkswagen rabbit         23641.
 4   243  21.5         4          121        110   2600         12.8    77      2 bmw 320i                  23927.
 5   307  28.8         6          173        115   2595         11.3    79      1 chevrolet citation        30000
 6   308  26.8         6          173        115   2700         12.9    79      1 oldsmobile omega brougham 45924.
 7   334  32.7         6          168        132   2910         11.4    80      3 datsun 280-zx             25944.
 8   335  23.7         3           70        100   2420         12.5    80      3 mazda rx-7 gs             44730.
 9   342  23.5         6          173        110   2725         12.6    81      1 chevrolet citation        40000
10   343  30           4          135         84   2385         12.9    81      1 plymouth reliant          33838.
11   362  25.4         6          168        116   2900         12.6    81      3 toyota cressida           30000
12   392  36           4          135         84   2370         13      82      1 dodge charger 2.2         17422.
13   396  32           4          135         84   2295         11.6    82      1 dodge rampage             47800
>
```

Figure 11 - All cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive)

5 Select the car names and horsepower of the cars with 3 cylinders

Dplyr syntax: cars_db %>% filter(cylinders == 3) %>% select(car_name, horsepower)

```
> #Get the car names and horsepower of the cars with 3 cylinders
> cars_db %>% filter(cylinders == 3) %>% select(car_name, horsepower)
# Source:   lazy query [?? x 2]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
  car_name         horsepower
  <chr>                 <dbl>
1 mazda rx2 coupe          97
2 maxda rx3                90
3 mazda rx-4              110
4 mazda rx-7 gs           100
>
```

Figure 12 – The car names and horsepower of the cars with 3 cylinders

TASK 4: Visualisation

1.1 Distribution of values for cylinders

> cylinders<-cars_db %>% select(cylinders) %>% data.frame

> hist(cylinders$cylinders, main = "Histogram of Cylinders", xlab = "Number of cylinders", col='light blue')

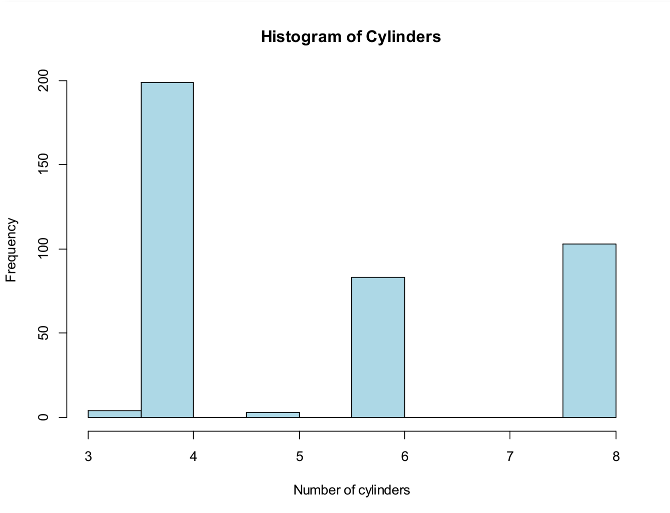

Figure 13 – Histogram of Cylinders

The chart "Histogram of Cylinders" represents the frequency of the number of cylinders in cars manufactured in years 1970 to 1982. Majority of cars has an even number of cylinders although, a few models had 3 or 5 cylinders too. There were nearly 200 models of cars with 4 cylinders manufactured in the 13-year period. Around 80 models were manufactured with 6 cylinders and around 100 models with 8-cylinder engines. The dataset doesn't contain any cars with 7-cylinder engine.

For more exact number of cars, we can display table using query:

> cars_db %>% group_by(cylinders) %>% summarise(count(cylinders))

```
> #Display table showing number of models by cylinder
> cars_db %>% group_by(cylinders) %>% summarise(count(cylinders))
# Source:   lazy query [?? x 2]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
  cylinders `count(cylinders)`
      <dbl>              <dbl>
1         3                  4
2         4                199
3         5                  3
4         6                 83
5         8                103
```

Figure 14 – Number of models per cylinder count

Note: Figure 14 showing actual number of cylinders per cylinder count showed in histogram in the figure 13.

```
> cars_db %>% group_by(model) %>% summarise(avg(cylinders))
# Source:   lazy query [?? x 2]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
   model `avg(cylinders)`
   <dbl>            <dbl>
 1    70             6.76
 2    71             5.63
 3    72             5.82
 4    73             6.38
 5    74             5.23
 6    75             5.6
 7    76             5.65
 8    77             5.46
 9    78             5.36
10    79             5.83
11    80             4.15
12    81             4.64
13    82             4.2
```

Figure 15 – Average number of engine cylinders in cars manufactured per year

The table in figure 15 show descending trend of the number of cylinders in years 1970 to 1982. An average model of car that was made in 1976 would have at least 6 cylinders however, a car made in 1982 had only 4 engines in average which is decrease of one third in cylinder count.

1.2 Distribution of values for miles per gallon

```
> mpg<-cars_db %>% select(mpg) %>% data.frame
```

```
> hist(mpg$mpg, main = "Histogram of Miles per gallon",  xlab = "Number of Miles per gallon", col='light blue')
```

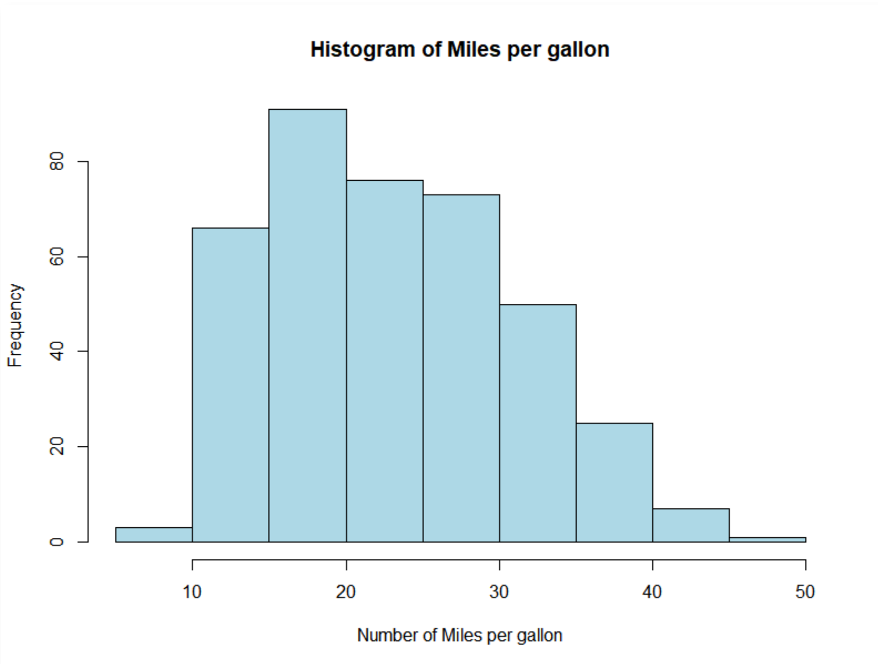

Figure 16 – Histogram of Miles per gallon

The figure 15 shows a histogram of fuel consumption of all the cars from the dataset. Only 3 cars from the dataset have higher consumption than 10 mpg. Majority of the car within the dataset has fuel consumption ranging between 10 to 30 mpg.

```
> cars_db %>% group_by(model) %>% summarise(avg(mpg))
# Source:   lazy query [?? x 2]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
   model `avg(mpg)`
   <dbl>      <dbl>
 1    70       17.7
 2    71       21.1
 3    72       18.7
 4    73       17.1
 5    74       22.8
 6    75       20.3
 7    76       21.6
 8    77       23.4
 9    78       24.1
10    79       25.1
11    80       33.8
12    81       30.2
13    82       32
```

Figure 17 – Average fuel consumption of cars models manufactured per year

There is increasing trend in maximum range from 17.7 to 32 miles per gallon within the 13 years covered by the sample.

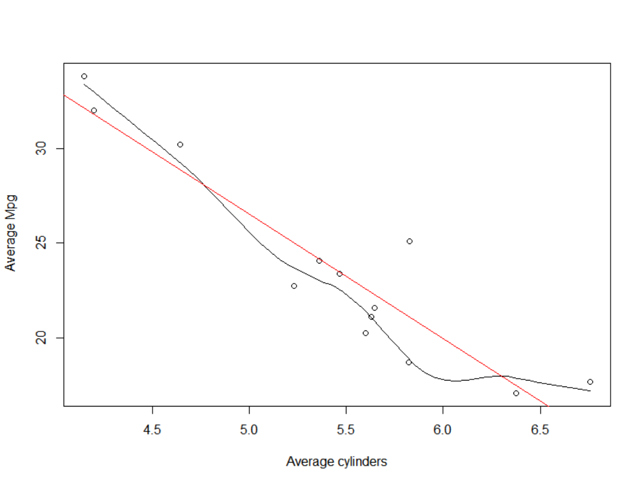

Figure 18 – Relationship between Number of Cylinders and fuel consumption

Figure 18 shows negative correlation between average mpg and cylinders. As the number of cylinders growing the maximum range per gallon is decreasing. We can see that average of millage per gallon decreases linearly with average of cylinders. Each step in the graph is and average of particular year.

2 Mean and distribution of Mpg measurements for each year

Filter values for a year (1970):

```
> y70<-cars_db %>% filter(model == 70) %>% select(mpg) %>% data.frame
```

Repeat operation above for every year in the dataset.

Display boxplot:

```
> boxplot( y70$mpg, y71$mpg, y72$mpg, y73$mpg, y74$mpg, y75$mpg, y76$mpg, y77$mpg, y78$mpg, y79$mpg,
y80$mpg,y81$mpg,y82$mpg,
    xlab = "Mean and distribution of mpg for each yaer",
    ylab = "Miles per gallon",
    names=c(1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982),
    col="light blue")
```

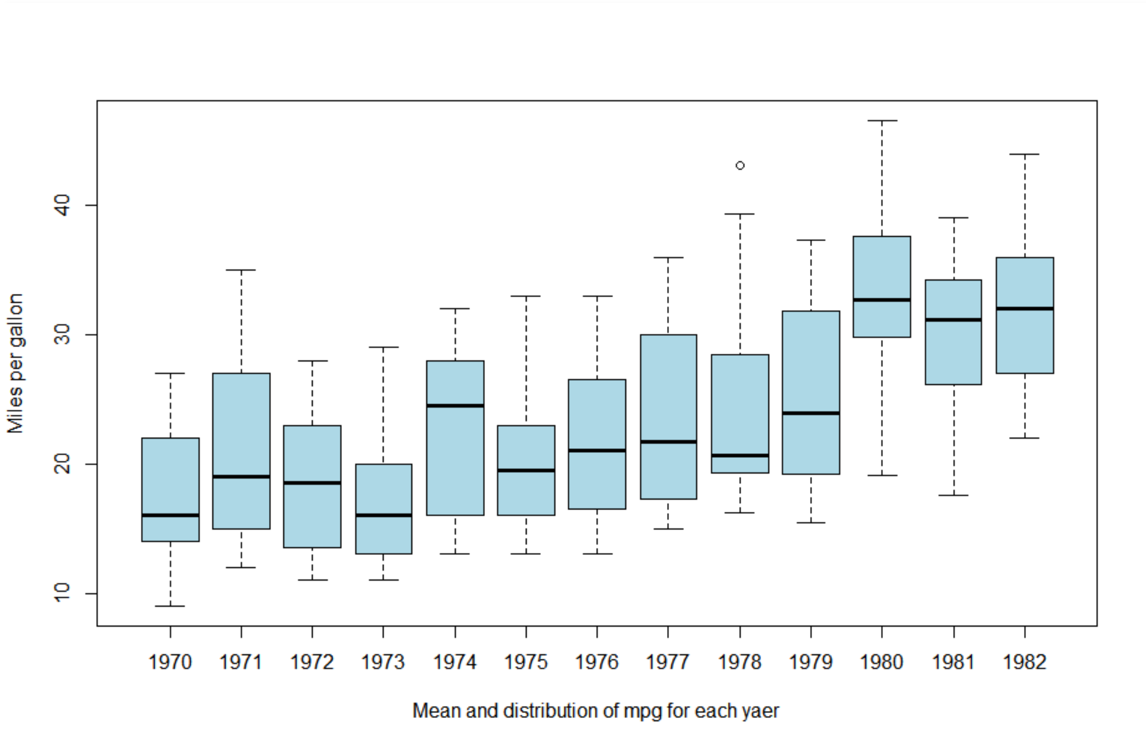

Figure 19 – Mean and distribution of Mpg for each year of dataset

It's apparent form the chart that mean mpg range was increasing over time. Models from early 70's have mean of maximum range below 20 Mpg compare to early 80's where the mean maximum range was over 30 Mpg.

2 Scatter Plot Showing the relationship between weight and Mpg

```
> cars_table <- dbReadTable(con, "Cars")

> scatter.smooth( cars_table$weight, cars_table$mpg, xlab= "Weight (Lbs.)", ylab = "Consuption Mpg", col="blue")

> abline(lm(cars_table$mpg ~cars_table$weight ), col="red")
```

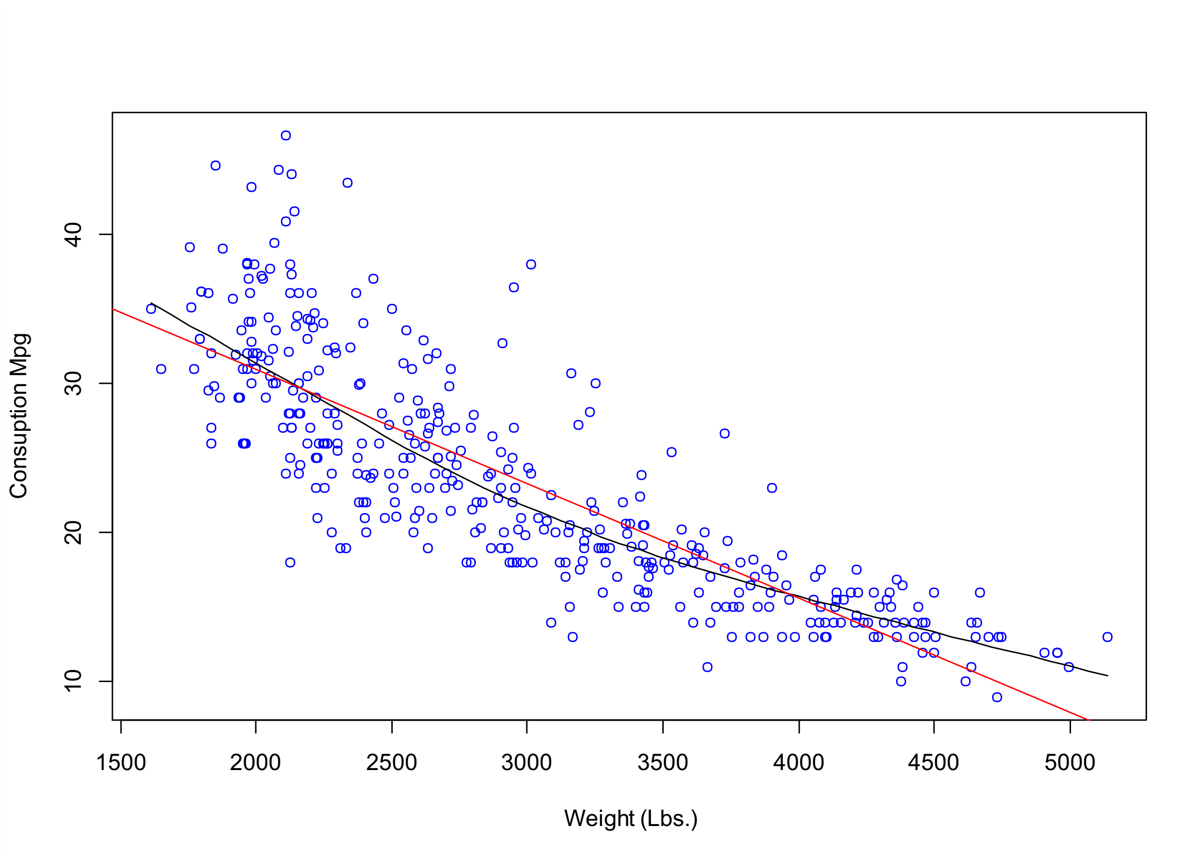

Figure 20 – Relationship between weight and millage per gallon

The scatter plot chart is showing negative correlation between weight of vehicles and millage per gallon. It means that lighter vehicles have lover consumption of fuel hence can make higher range per one gallon. If we metric standards were applied and instead of MPG we considered consumption per 100 kilometres, the trend of the graph would be opposite as the heavy cars would consume higher amount of fuel per 100 Km. (E.g. 30 Mpg ~ 7.8 l/100 km and 10 Mpg ~ 23.5 l/100 km).

Q2 OLAP Operations

Task 1: Generate Sales Function

Create State table

```
state_table <- data.frame(key=c("FR", "LA", "SY", "SE", "CT"),

                  name=c("Frankfurt", "Los Angeles", "Sydney", "Seoul", "Cape Town"),

                  country=c("Germany", "USA", "Australia", "S. Korea", "South Africa"))
```

Create Moth Table

```
month_table <- data.frame(key=1:12,

          desc=c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),

         quarter=c("Q1","Q1","Q1","Q2","Q2","Q2","Q3","Q3","Q3","Q4","Q4","Q4"))
```

Create Product Table

```
prod_table <-

          data.frame(key=c("Washing Machine", "Fridge", "Vacuum Cleaner", "Microwave Oven"),

          price=c(200, 500, 400, 150),

          stringsAsFactors = TRUE)
```

I identified a bug in the tutorial function gen_sales as the prices assigned were shifted by one. The tutorial was easy to fix by assigning the index +1. However, the in the case of the coursework the prices were shifted too, apart of the Vacuum Cleaner. I decided to rearrange the prod_table inot 'price=c(500, 150, 400, 200)' which fixes the problem in sales fact and the prices are assigned in correct order.

Define function gen_sales

Function 'gen_sales' generates a random sample of records based on 'prod_table', 'month_table', 'state_table' and parameter 'no_of_recs' that determines the number of records in the sample.

```
gen_sales <- function(no_of_recs) {
  # Generate transaction data randomly
  loc <- sample(state_table$key, no_of_recs, replace=T, prob=c(2,2,1,1,1))
  time_month <- sample(month_table$key, no_of_recs, replace=T)
  time_year <- sample(c(2015, 2016, 2017, 2018, 2019, 2020), no_of_recs, replace=T)
  prod <- sample(prod_table$key, no_of_recs, replace=T, prob=c(1, 3, 2, 4))
  unit <- sample(c(1,2), no_of_recs, replace=T, prob=c(10, 4))
  amount <- unit*prod_table[prod,]$price

  sales <- data.frame(month=time_month,
                 year=time_year,
                 loc=loc,
                 prod=prod,
                 unit=unit,
                 amount=amount)
  # Sort the records by time order
```

```
    sales <- sales[order(sales$year, sales$month),]
    row.names(sales) <- NULL
    return(sales)
  }
```

```
#Generate Samples

sales_fact <- gen_sales(500)
```

```
head(sales_facts)
```

	month	year	loc	prod	unit	amount
1	1	2015	SY	Vacuum Cleaner	1	400
2	1	2015	LA	Vacuum Cleaner	1	400
3	1	2015	LA	Microwave Oven	1	150
4	2	2015	SY	Microwave Oven	2	300
5	2	2015	SE	Microwave Oven	2	300
6	2	2015	LA	Microwave Oven	1	150

Save data into database

```
dbWriteTable(con, name='Sales_CWK2', value=sales_fact, overwrite=TRUE)
```

Saving generated data into the database allows us to reuse the same dataset within in the future.

Task 2: Revenue Cube

```
revenue_cube <- tapply(sales_table$amount, sales_fact[,c("prod", "month", "year", "loc")],
                FUN=function(x){return(sum(x))})
```

```
> dimnames(revenue_cube)
$prod
[1] "Fridge"          "Microwave Oven"  "Vacuum Cleaner"  "Washing Machine"

$month
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12"

$year
[1] "2015" "2016" "2017" "2018" "2019" "2020"

$loc
[1] "CT" "FR" "LA" "SE" "SY"

> |
```

Figure 21 – Dimension Names of the revenue cube

```
> revenue_cube
, , year = 2015, loc = CT

                 month
prod                1   2    3   4  5  6  7   8   9   10 11  12
  Fridge           NA 500 1000  NA NA NA NA 500  NA  500 NA  NA
  Microwave Oven   NA  NA  300 300 NA NA NA  NA  NA   NA NA 150
  Vacuum Cleaner   NA  NA  400 400 NA NA NA  NA  NA 1200 NA 800
  Washing Machine NA  NA   NA  NA NA NA NA  NA 200   NA NA  NA

, , year = 2016, loc = CT

                 month
prod                 1   2   3   4   5   6   7   8  9  10  11  12
  Fridge           500  NA  NA  NA  NA  NA  NA  NA NA  NA 500  NA
  Microwave Oven    NA 150  NA  NA 150  NA 150  NA NA  NA 300 300
  Vacuum Cleaner    NA  NA 400  NA  NA  NA  NA 400 NA 800  NA  NA
  Washing Machine   NA  NA  NA 200  NA 200  NA 200 NA  NA  NA  NA

, , year = 2017, loc = CT

                 month
prod                 1  2   3  4   5   6   7   8   9 10  11  12
  Fridge           500 NA 500 NA 500  NA 500 500 500 NA  NA  NA
  Microwave Oven    NA NA 300 NA  NA 150  NA  NA 600 NA 150 150
  Vacuum Cleaner    NA NA  NA NA  NA 800  NA  NA  NA NA 400  NA
  Washing Machine 400 NA  NA NA  NA 400  NA  NA 200 NA  NA 200

, , year = 2018, loc = CT

                 month
prod                1   2  3   4  5   6   7   8   9  10 11 12
  Fridge           NA  NA NA  NA NA  NA 500 500  NA  NA NA NA
  Microwave Oven   NA  NA NA 150 NA  NA  NA  NA  NA  NA NA NA
  Vacuum Cleaner   NA  NA NA  NA NA 800  NA 400  NA  NA NA NA
  Washing Machine NA 200 NA 200 NA  NA 600  NA 200 200 NA NA

, , year = 2019, loc = CT

                 month
prod                 1  2   3  4   5  6   7   8   9  10  11  12
  Fridge            NA NA 500 NA 500 NA 500  NA  NA  NA 500  NA
  Microwave Oven    NA NA  NA NA  NA NA 150  NA 150 300  NA 150
  Vacuum Cleaner   400 NA  NA NA 400 NA  NA 800  NA  NA  NA  NA
  Washing Machine   NA NA  NA NA  NA NA  NA 200  NA  NA  NA  NA
```

Figure 22 – Print of part of the revenue cube

Task 3: OLAP Functions

Slice

revenue_cube[, "1", "2016",]

```
> revenue_cube[, "1", "2016",]
                 loc
prod               CT FR  LA  SE SY
  Fridge          500 NA  NA  NA NA
  Microwave Oven   NA NA  NA  NA NA
  Vacuum Cleaner   NA NA  NA  NA NA
  Washing Machine  NA NA 800 200 NA
```

Figure 23 – Slice example 1

revenue_cube["Fridge", "1", "2017",]

```
> revenue_cube["Fridge", "1", "2017",]
  CT    FR    LA    SE    SY
 500    NA   500    NA  1000
```

Figure 24 – Slice example 2

Example displayed in the figure 23 displaying sales from January 2016. There were sales of Fridge in Cape Town, South Africa in total of $500 and sales of Washing Machine in LA, USA $800 and $200 in Seoul, South Korea.

Example in the figure 24 Focuses Fridge sales in January 2017. There was one item sold in Cape Town, South Africa, one unit in Los Angeles, USA and two units in total price of $100 in Sydney, Australia.

Dice

revenue_cube[c("Washing Machine", "Fridge"),

c("1","2",'5'), ,

c("SY","LA")]

```
>   revenue_cube[c("Washing Machine", "Fridge"),
+              c("1","2",'5'), ,
+              c("SY","LA")]
, , year = 2015, loc = SY

                 month
prod                1   2  5
  Washing Machine NA 200 NA
  Fridge          NA  NA NA

, , year = 2016, loc = SY

                 month
prod                1   2  5
  Washing Machine NA 200 NA
  Fridge          NA  NA NA

, , year = 2017, loc = SY

                 month
prod                 1    2  5
  Washing Machine  200   NA NA
  Fridge          1000 1000 NA

, , year = 2018, loc = SY

                 month
prod                1  2  5
  Washing Machine NA NA NA
  Fridge          NA NA NA

, , year = 2019, loc = SY
```

Figure 25 – Dice Example 1 – partial print

revenue_cube[c("Washing Machine", "Vacuum Cleaner"),

c("1","2",'12'), ,

c("SE","LA", "CT")]

```
>  revenue_cube[c("Washing Machine", "Vacuum Cleaner"),
+              c("1","2",'12'), ,
+              c("SE","LA", "CT")]
, , year = 2015, loc = SE

                 month
prod               1   2 12
  Washing Machine NA 200 NA
  Vacuum Cleaner  NA  NA NA

, , year = 2016, loc = SE

                 month
prod                1   2  12
  Washing Machine 200  NA 400
  Vacuum Cleaner   NA 400  NA

, , year = 2017, loc = SE

                 month
prod                1  2 12
  Washing Machine 400 NA NA
  Vacuum Cleaner   NA NA NA

, , year = 2018, loc = SE

                 month
prod               1   2 12
  Washing Machine NA  NA NA
  Vacuum Cleaner  NA 400 NA

, , year = 2019, loc = SE
```

Figure 26 – Dice Example 2 – partial print

Dice operation focuses on certain values keeping the number of dimensions of the cube. Example in figure 26 focuses on Washing Machines and Vacuum Cleaners sales in January, February and December in Seoul, Los Angeles and Cape Town. Each year is displayed separately. Note: The figure contains only partial print of the operation.

Roll-up

apply(revenue_cube, c("year", "prod"),

FUN=function(x) {return(sum(x, na.rm=TRUE))})

```
> #######Roll-up
> apply(revenue_cube, c("year", "prod"),
+         FUN=function(x) {return(sum(x, na.rm=TRUE))})
      prod
year   Fridge Microwave Oven Vacuum Cleaner Washing Machine
  2015  11500           3600          13600            4800
  2016  11000           4350          10000            6200
  2017  15500           4950           6000            7000
  2018  14000           3750           7600            6600
  2019  11500           4050          12400            3600
  2020  12500           4050           8400            5400
> 
```

Figure 27 – Rollup

Roll-up focuses on annual revenue of each product and collapses the location dimension. The example in the figure 27 focuses on all product within the 6 years in the dataset.

Drill-down

```
apply(revenue_cube, c("year", "month", "prod"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
> #######Drill-down
> apply(revenue_cube, c("year", "month", "prod"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
, , prod = Fridge

      month
year      1    2    3    4    5    6    7    8    9   10   11   12
  2015    0  500 1500 1000  500 1000 1500  500  500 1500 1000 2000
  2016  500 1000    0 1000  500  500  500  500 1500 1500 2000 1500
  2017 2000 1000  500  500  500    0 3500 2500 3000 1000  500  500
  2018 1500 2000 1000 2000  500  500 2000  500 1000  500 2000  500
  2019 1000    0 2000    0 1000    0 1000    0    0 1500 2000 3000
  2020 1000    0  500  500 3000 2000 1000 1500    0 1000 1000 1000

, , prod = Microwave Oven

      month
year     1   2   3   4   5   6   7   8    9  10  11   12
  2015 150 750 750 450 150 300   0 300  150 150 150  300
  2016   0 600 150 300 450 450 300 300    0 150 450 1200
  2017 450   0 600 150 600 300 150   0 1050 450 150 1050
  2018 450 150 450 750 300 150 300 900  300   0   0    0
  2019 300 900 150 150   0 450 150 150  450 600   0  750
  2020 450 450 450 900 300   0 150   0  300 450 150  450

, , prod = Vacuum Cleaner

      month
year      1    2    3    4    5    6   7    8    9   10  11   12
  2015  800  800 1200 2000    0  800 400 1600 2400 1600 400 1600
  2016    0 1600  400    0  800    0 800 2000 2400  800   0 1200
  2017  400  400 1200    0    0 1200   0 1200  800    0 800    0
  2018    0  400    0 1200    0 1200 400 1200 1200  800 800  400
  2019 3200  800 1200 1200 1600  400 400  800  400  800 400 1200
  2020    0  400  800  800  800  800   0  800 2000    0 800 1200

, , prod = Washing Machine

      month
year      1   2   3    4   5   6   7   8    9   10  11   12
  2015    0 400 600  600   0 600   0 200  200  600 800  800
  2016 1000 400 200 1000 200 800 400 800  200  400 200  600
  2017 1600   0 400  400 600 400 400 800  400 1000   0 1000
  2018  400 800 200  400 400 600 600   0 1400 1200 400  200
  2019  400 400 400    0 400 800   0 200  600  400   0    0
```

Figure 28 – Drill-down

Drill-down works as a reverse of Roll-up function. Focuses on annual and monthly revenue for each product and collapses the local dimensions.

Pivot

```
apply(revenue_cube, c("year", "month"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
> #########Pivot
> apply(revenue_cube, c("year", "month"),
+       FUN=function(x) {return(sum(x, na.rm=TRUE))})
      month
year      1    2    3    4    5    6    7    8    9   10   11   12
  2015  950 2450 4050 4050  650 2700 1900 2600 3250 3850 2350 4700
  2016 1500 3600  750 2300 1950 1750 2000 3600 4100 2850 2650 4500
  2017 4450 1400 2700 1050 1700 1900 4050 4500 5250 2450 1450 2550
  2018 2350 3350 1650 4350 1200 2450 3300 2600 3900 2500 3200 1100
  2019 4900 2100 3750 1350 3000 1650 1550 1150 1450 3300 2400 4950
  2020 2050 1250 1950 2800 4500 3000 1550 2700 2700 2650 2150 3050
```

Figure 29 – Pivots

Pivot analyses the combination of pairs of selected dimensions. The example in the figure 29 focuses on analytics of revenue of a year and month. In the same way we could plot product and location, or product and month.

Q3 Decision Support System

Task 1: Import dataset

#Load data from a file

liver_file = read.csv("C:/Users/w1701833/Desktop/Business Intelligence/liver.csv")

#Upload data to database

dbWriteTable(con, name='Liver', value=liver_file)

#Display table names

dbListTables(con)

[1] "Cars"   "Liver"   "Order_Line" "Orders"   "Product"   "Sales"   "TABLE 10"   "TABLE 8"

[9] "TABLE 9"   "Users"   "cars_info"

#Display Fileds in Liver table

dbListFields(con, 'Liver')

[1] "row_names"   "age"   "gender"   "tot_bilirubin"   "direct_bilirubin"

[6] "tot_proteins"   "albumin"   "ag_ratio"   "sgpt"   "sgot"

[11] "alkphos"   "is_patient"

liver_table <- tbl(con, "Liver")

head(liver_table)

```
> ################ LOAD DATA FROM DATABASE ########
> liver_table <- tbl(con, "Liver")
> head(liver_table)
# Source:   lazy query [?? x 12]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
  row_names   age gender tot_bilirubin direct_bilirubin tot_proteins albumin ag_ratio  sgpt  sgot alkphos
  <chr>     <dbl> <chr>          <dbl>            <dbl>        <dbl>   <dbl>    <dbl> <dbl> <dbl>   <dbl>
1 1            65 Female           0.7              0.1          187      16       18   6.8   3.3    0.9
2 2            62 Male            10.9              5.5          699      64      100   7.5   3.2    0.74
3 3            62 Male             7.3              4.1          490      60       68   7     3.3    0.89
4 4            58 Male             1                0.4          182      14       20   6.8   3.4    1
5 5            72 Male             3.9              2            195      27       59   7.3   2.4    0.4
6 6            46 Male             1.8              0.7          208      19       14   7.6   4.4    1.3
# ... with 1 more variable: is_patient <dbl>
> 
```

Figure 30 – Mean and distribution of Mpg for each year of dataset

Task 2.1: Data preparation

# Load ‘Liver’ table as data frame
liver_table <- tbl(con, "Liver") %>% data.frame

#Drop all Observations with NULL value
liver_filtered <- liver_table %>% drop_na()

# Replace gender with numerical values
liver_filtered[liver_filtered == "Male"] <-1

liver_filtered[liver_filtered == "Female"] <-2
Represent male subjects as 1 and female subject as 2.

Save serialised data into database as 'Liver_worksample'

dbWriteTable(con, name='Liver_worksample', value=liver_filtered[-1], overwrite=TRUE)

Task 2.2: Data Exploration

Mean age value of subjects

```
> #Mean(avg) of subjects
> liverDB %>% group_by(gender) %>% summarise(avg(age))
# Source:   lazy query [?? x 2]
# Database: mysql 5.7.32-0ubuntu0.18.04.1 [w1701833@127.0.0.1:/w1701833_0]
  gender `avg(age)`
  <chr>       <dbl>
1 1            45.3
2 2            43.2
> 
```

Figure 31 – Mean age of subjects grouped by gender. One represents male subjects two female subjects

Median age and quantiles value of subjects

```
#Median and quartiles of subject ages
female_age <-liverDB %>% select(age, gender) %>% filter(gender == 2) %>% data.frame
male_age <-liverDB %>% select(age, gender) %>% filter(gender == 1) %>% data.frame

female_Q <- quantile(female_age$age, probs=c(.25, .5, .75), na.rm = FALSE)
male_Q <- quantile(male_age$age, probs=c(.25, .5, .75), na.rm = FALSE)
```

```
> #Median  and quartiles of subject ages
> female_age <-liverDB %>% select(age, gender) %>% filter(gender == 2) %>% data.frame
> male_age <-liverDB %>% select(age, gender) %>% filter(gender == 1) %>% data.frame
> female_Q <- quantile(female_age$age, probs=c(.25, .5, .75), na.rm = FALSE)
> male_Q <- quantile(male_age$age, probs=c(.25, .5, .75), na.rm = FALSE)
> female_Q
25% 50% 75% 
 31  45  53 
> male_Q
25% 50% 75% 
 33  45  60 
> 
```

Figure 32 – Median values and quantiles for male and female subjects

From the figure __ is apparent that median age for female subjects is 45 years, the lower quantile age is 31 years and upper quantile is 53 years. In case of male subjects, the median value is also 45 years, but lower quantile is 33 years and upper quantile is 60 years.

Task 2.3: Data visualisation

Histogram of the frequency of patients per age

```
#extract age data
agehist<-liverDB %>% select(age) %>% data.frame#


#display histogram (step 1)
hist(agehist$age, breaks = seq(1,100, by=1), xlim = c(0,100),
col="light blue", xlab="Age (Step by 1)")

#display histogram (step 3)
hist(agehist$age, breaks = seq(1,100, by=3), xlim = c(0,100),  ylim = c(0,50),
col="light blue", xlab="Age (Step by 3)")
```

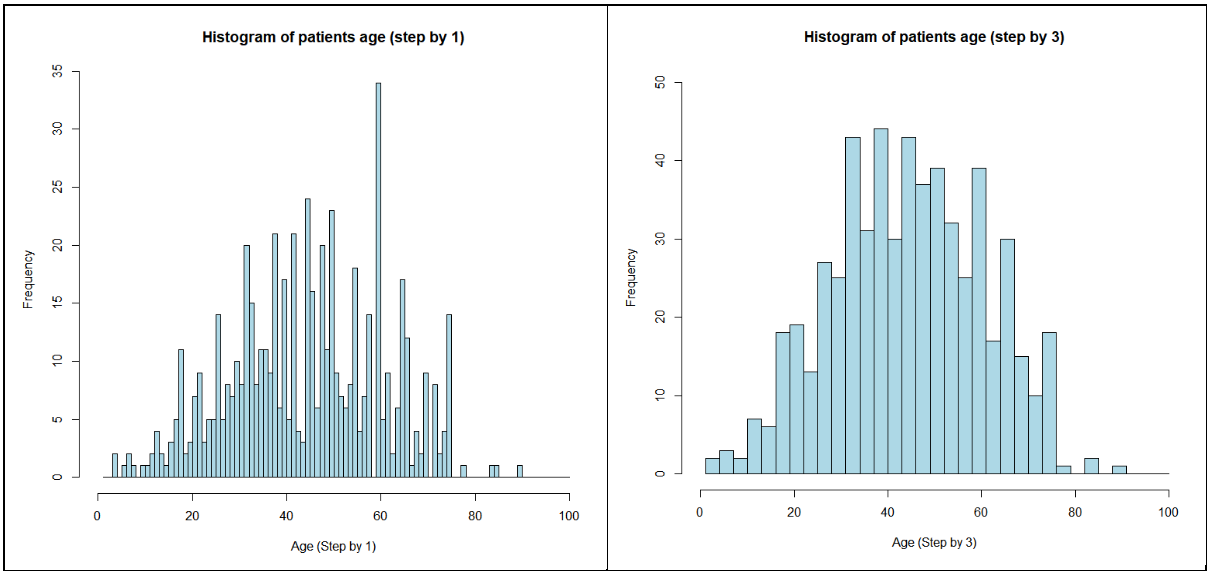

Figure 33 – Frequency of age of age by step of 1 and 3 years

Histogram of the frequency of patients per sgpt

```
#extract data
sgpthist<-liverDB %>% select(sgpt) %>% data.frame

#display histogram (step by 0.1)
hist(sgpthist$sgpt, breaks = seq(1,10, by=.1), xlim = c(2,10), ylim = c(0,35),
col="light blue", xlab="Alamine Aminotransferase (Step by 0.1)",
main = "Histogram of Alamine Aminotransferase (step by 0.1)")

#display histogram (step by 0.2)
hist(sgpthist$sgpt, breaks = seq(1,10, by=.2), xlim = c(2,10), ylim = c(0,60),
col="light blue", xlab="Alamine Aminotransferase (Step by 0.2)",
main = "Histogram of Alamine Aminotransferase (step by 0.2)")
```

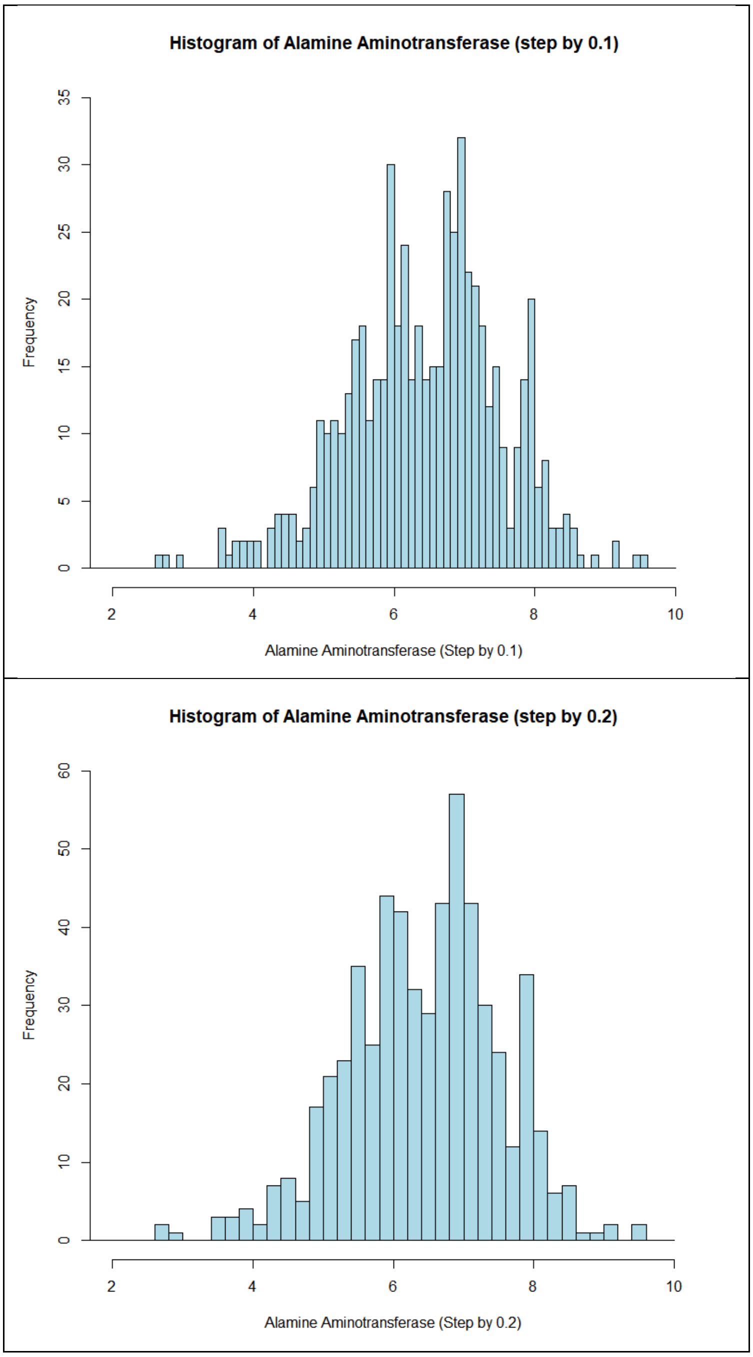

Figure 34 – Histogram of the frequency of patients per sgpt

Boxplot of gender of the subjects to age

```
###Perform a boxplot of Gender (Male & Female) vs age
boxplot(female_age$age, male_age$age, names=c("Female","Male"),
col = "light blue", horizontal = TRUE)
```

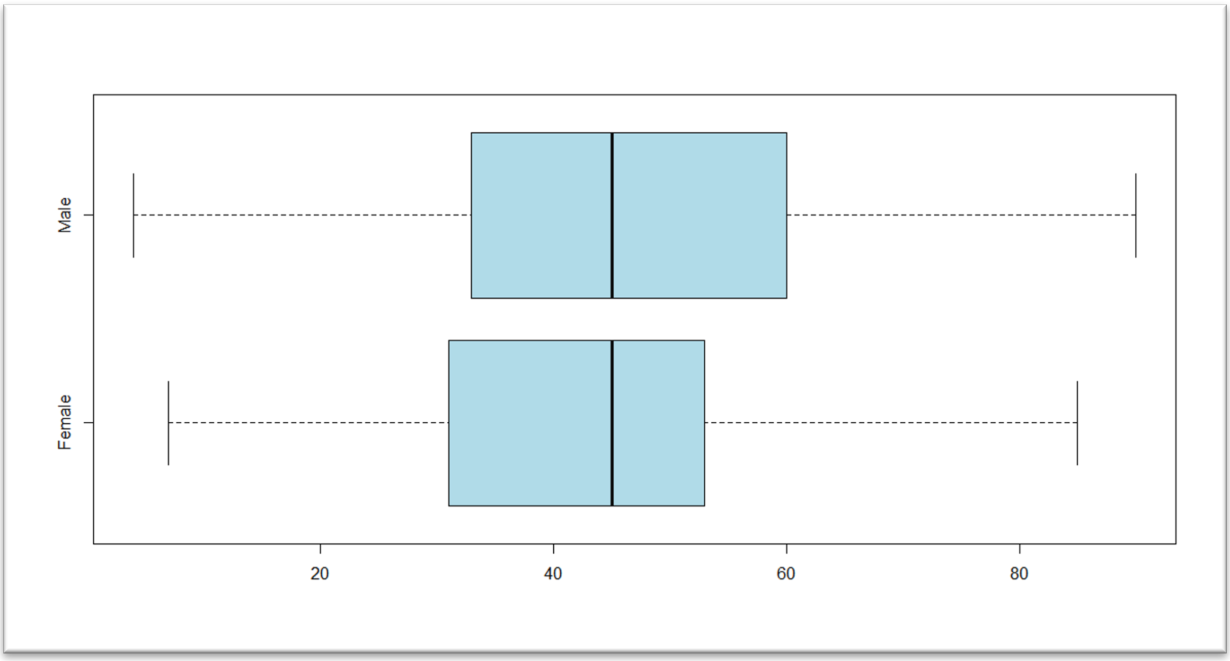

Figure 35 – Boxplot of gender of the subjects to age

Task 2.4: Generating Training and testing dataset

```
#table from database
liverDB <- tbl(con, 'Liver_worksample')

#save to dataframe
liver_ <- liverDB %>% data.frame

#Slice data for testing and training sample
set.seed(3003)
intrain <- createDataPartition(y = liver_$is_patient, p= 0.8, list = FALSE)
training <- liver_[intrain,]
testing <- liver_[-intrain,]

#save training and testing samples to the database
dbWriteTable(con, name='Liver_training', value=training[-1], overwrite=TRUE)
dbWriteTable(con, name='Liver_testing', value=testing[-1], overwrite=TRUE)
```

Code above splits data into 2 separate sets: training and testing by random. The testing dataset contains 20 per cent of the original data i.e. 115 samples of observed data and the training dataset contains 464 samples which are the other 80 per cent of the original 579 serialised samples. Although generating of the dataset is repeatable as the seed 3003 is used, the data are saved in database in tables ‘Liver_training’ and ‘Liver_testing’. This allows us reusing of the dataset in different sections of the code or even in different files without the need of regenerating the data from the original dataset.

Task 3: Decision Tree Model based on C5.0 algorithm

Loading data

Training and testing data are being saved in database, so they can be reused in various different parts of the code without the need of generating them again.

```
#Load data
liver_train <- tbl(con, 'Liver_training')
liver_test <- tbl(con, 'Liver_testing')

training_set <- liver_train %>% data.frame
training_set <- training_set[-1] #%>% data.frame

testing_set <- liver_test %>%
select(age,gender,tot_bilirubin,direct_bilirubin,tot_proteins,albumin,ag_ratio,sgpt,sgot,alkphos,is_patient) %>%
data.frame
```

Create Model

```
#Convert data to factor
training_set$is_patient <- as.factor(training_set$is_patient)

set.seed(2345)
model <- C5.0(is_patient ~., data=training_set )

plot(model)
```

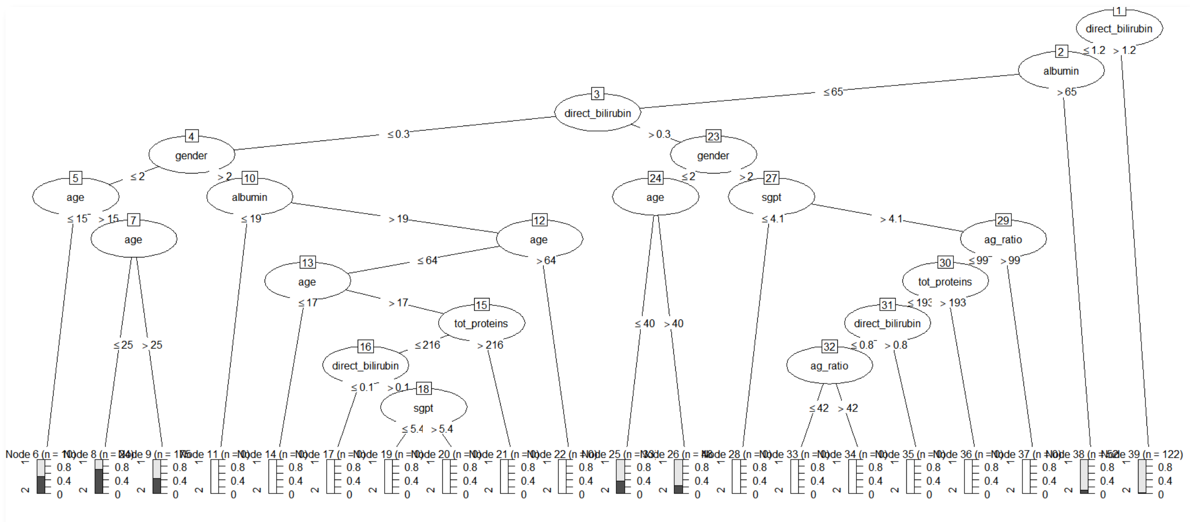

Figure 36 – Model of full DT using C5.0 algorithm

Test Decision Tree Classifier

```
#test tree
results <- predict(object=model, newdata=testing_set, type="class")
```

Evaliation

```
tt <- as.factor(testing_set$is_patient)

confusionMatrix(results, tt)
```

```
>  confusionMatrix(results, tt)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 68 20
         2 14 13

               Accuracy : 0.7043
                 95% CI : (0.6121, 0.7858)
    No Information Rate : 0.713
    P-Value [Acc > NIR] : 0.6264

                  Kappa : 0.236

 Mcnemar's Test P-Value : 0.3912

            Sensitivity : 0.8293
            Specificity : 0.3939
         Pos Pred Value : 0.7727
         Neg Pred Value : 0.4815
             Prevalence : 0.7130
         Detection Rate : 0.5913
   Detection Prevalence : 0.7652
      Balanced Accuracy : 0.6116

       'Positive' Class : 1

>
```

Figure 37 – Confusion Matrix of C5.0 DT classifier

F1 Score

> F1_Score(results, tt)

[1] 0.8

ROC and Area under the curve

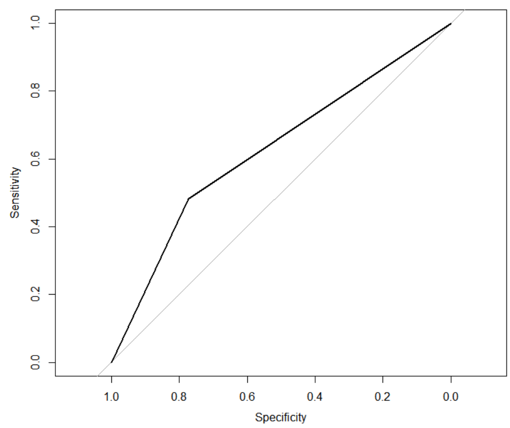

Figure 38 – ROC Curve for C5.0 Model Classifier

Area under the curve: 0.6271

Task 4: Decision Tree Model based on CART algorithm

In this task will be used function 'rpart'. In further improvement CART algorithm will use CARET package instead.

Loading data

```
###Load data
liver_train <- tbl(con, 'Liver_training')
liver_test <- tbl(con, 'Liver_testing')

training_set <- liver_train %>% data.frame
training_set <- training_set[-1] #%>% data.frame

testing_set <- liver_test %>%

select(age,gender,tot_bilirubin,direct_bilirubin,tot_proteins,albumin,ag_ratio,sgpt,sgot,alkphos,is_patient) %>% data.frame
```

Train Tree

```
set.seed(3333)

model <- rpart(is_patient~.,data=training_set,method="class")

model
```

```
>  model
n= 464

node), split, n, loss, yval, (yprob)
      * denotes terminal node

   1) root 464 132 1 (0.71551724 0.28448276)
     2) tot_bilirubin>=1.65 178  16 1 (0.91011236 0.08988764) *
     3) tot_bilirubin< 1.65 286 116 1 (0.59440559 0.40559441)
       6) tot_proteins>=211.5 96  21 1 (0.78125000 0.21875000) *
       7) tot_proteins< 211.5 190  95 1 (0.50000000 0.50000000)
        14) age>=24.5 171  80 1 (0.53216374 0.46783626)
          28) albumin>=66.5 8   1 1 (0.87500000 0.12500000) *
          29) albumin< 66.5 163  79 1 (0.51533742 0.48466258)
            58) sgot< 4.35 152  71 1 (0.53289474 0.46710526)
             116) albumin>=19.5 112  47 1 (0.58035714 0.41964286)
               232) age>=67 8   0 1 (1.00000000 0.00000000) *
               233) age< 67 104  47 1 (0.54807692 0.45192308)
                 466) alkphos>=0.93 66  25 1 (0.62121212 0.37878788)
                   932) age< 59 59  19 1 (0.67796610 0.32203390)
                    1864) sgot< 3.05 10   0 1 (1.00000000 0.00000000) *
                    1865) sgot>=3.05 49  19 1 (0.61224490 0.38775510)
                      3730) tot_proteins< 176.5 30   8 1 (0.73333333 0.26666667) *
                      3731) tot_proteins>=176.5 19   8 2 (0.42105263 0.57894737) *
                   933) age>=59 7   1 2 (0.14285714 0.85714286) *
                 467) alkphos< 0.93 38  16 2 (0.42105263 0.57894737)
                   934) age>=45.5 16   6 1 (0.62500000 0.37500000) *
                   935) age< 45.5 22   6 2 (0.27272727 0.72727273) *
             117) albumin< 19.5 40  16 2 (0.40000000 0.60000000)
               234) tot_bilirubin>=0.85 15   6 1 (0.60000000 0.40000000) *
               235) tot_bilirubin< 0.85 25   7 2 (0.28000000 0.72000000) *
            59) sgot>=4.35 11   3 2 (0.27272727 0.72727273) *
        15) age< 24.5 19   4 2 (0.21052632 0.78947368) *
> |
```

Figure 39 – CART tree classifier using rpart function

```
plot(model)
 text(model, digits = 3)
```

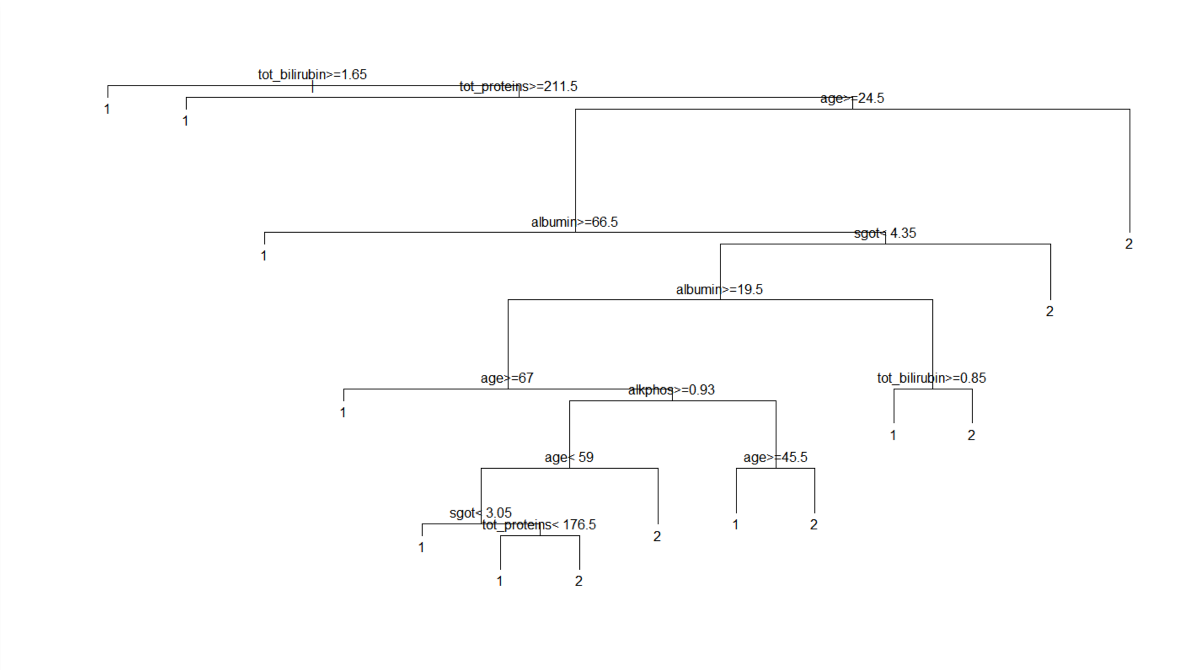

Figure 40 – Plot of CART tree classifier using rpart

Test Tree Classifier

```
# test tree
 predicted.classes <- model %>% predict(testing_set, type = "class")
```

Evaluation

```
#evaluate tree
 tt <- as.factor(testing_set$is_patient)
 confusionMatrix(predicted.classes, tt)

 #F1 Score
 F1_Score(predicted.classes, tt)

 #ROC
 cartROC <- roc(predict.classes, testing_set$is_patient, plot = TRUE)
 auc(cartROC)
```

```
>  tt <- as.factor(testing_set$is_patient)
>  confusionMatrix(predicted.classes, tt)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 72 19
         2 10 14

               Accuracy : 0.7478
                 95% CI : (0.6583, 0.8242)
    No Information Rate : 0.713
    P-Value [Acc > NIR] : 0.2375

                  Kappa : 0.3291

 Mcnemar's Test P-Value : 0.1374

            Sensitivity : 0.8780
            Specificity : 0.4242
         Pos Pred Value : 0.7912
         Neg Pred Value : 0.5833
             Prevalence : 0.7130
         Detection Rate : 0.6261
   Detection Prevalence : 0.7913
      Balanced Accuracy : 0.6511

       'Positive' Class : 1

>
```

Figure 41 – Confusion Matrix of CART DT classifier

F1 Score

> F1_Score(predicted.classes, tt)

[1] 0.8323699

ROC and AUC

cartROC <- roc(predicted.classes, testing_set$is_patient, plot = TRUE)
auc(cartROC)

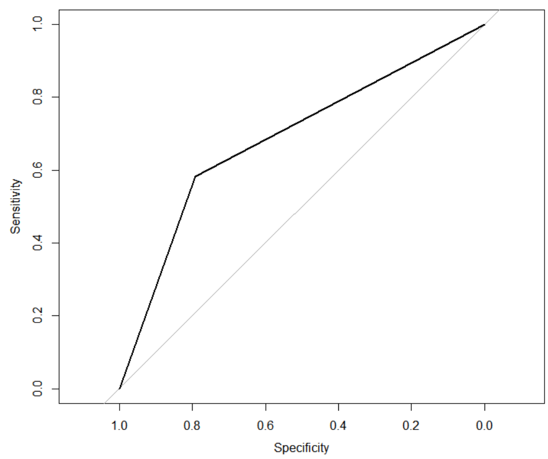

Figure 42 – ROC Curve for CART Model Classifier C

Area under the curve: 0.6873

Task 5: Evaluation

C5.0

Accuracy	0.7041	70%
F1 score	0.8	
Area Under Curve	0.6271	
Precision C1	0.7727	77%
Precision C2	0.4814	48%

CART

Accuracy	0.7478	75%
F1 score	0.8324	
Area Under Curve	0.6873	
Precision C1	0.7912	79%
Precision C2	0.5833	58%

F1 measures is another way to calculate overall accuracy. It is calculated in way of

$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$. In the same way as Accuracy, higher F1 score means higher result.

Precision focuses on each class separately and determines percentage of correctly assigned classes and false positives.

Area under the curve is area under ROC curve (Receiver operating characteristic), which is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. Larger area under the curve means better result of the model.

Overall accuracy of CART model is better by 5 per cent. F1 Score is higher by .03 compare to C5.0 and the are under the curve covers 6 per cent larger area. Precision for each class are higher too. Precision of recognition of Liver patients is higher by 3 per cent and non-liver patients as much as 10 per cent.

CART Model based on rpart package outperforms C5.0 in all aspects and is more suitable for this case study.

Task 6: Improvement of Current Models

I took several attempts of improvement of the winner CART model by pruning and boosting. Unfortunately, none of those attempts made had higher overall score compare to rpart CART model.

However, model with rctr <- rpart.control(maxdepth = 10 , minsplit = 10) control correctly classified more patients in the Liver Patients group compare to original CART model without pruning.

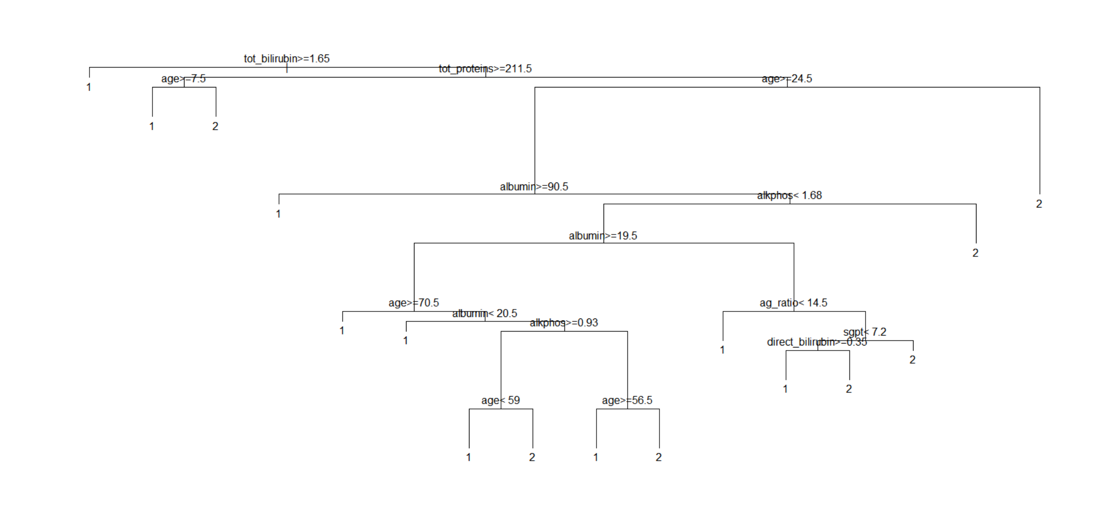

Figure 43 – Pruned Decision Tree (max depth 10 nodes), min number of cases to split =10

```
>  tt <- as.factor(testing_set$is_patient)
>  confusionMatrix(predicted.classes, tt)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 72 19
         2 10 14

               Accuracy : 0.7478
                 95% CI : (0.6583, 0.8242)
    No Information Rate : 0.713
    P-Value [Acc > NIR] : 0.2375

                  Kappa : 0.3291

 Mcnemar's Test P-Value : 0.1374

            Sensitivity : 0.8780
            Specificity : 0.4242
         Pos Pred Value : 0.7912
         Neg Pred Value : 0.5833
             Prevalence : 0.7130
         Detection Rate : 0.6261
   Detection Prevalence : 0.7913
      Balanced Accuracy : 0.6511

       'Positive' Class : 1

>
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 75 25
         2  7  8

               Accuracy : 0.7217
                 95% CI : (0.6305, 0.8013)
    No Information Rate : 0.713
    P-Value [Acc > NIR] : 0.464770

                  Kappa : 0.1876

 Mcnemar's Test P-Value : 0.002654

            Sensitivity : 0.9146
            Specificity : 0.2424
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.5333
             Prevalence : 0.7130
         Detection Rate : 0.6522
   Detection Prevalence : 0.8696
      Balanced Accuracy : 0.5785

       'Positive' Class : 1
```

Figure 44 – Comparison of original CART (left) and pruned model (right

CART

Accuracy	0.7478	75%
F1 score	0.8324	
Area Under Curve	0.6873	
Precision C1	0.7912	79%
Precision C2	0.5833	58%

Pruned CART

Accuracy	0.7217	72%
F1 score	0.8241758	
Area Under Curve	0.6417	
Precision C1	0.7947	79%
Precision C2	0.5333	54%

Overall, Original Cart model has better accuracy, larger area under the curve and higher F1 Score too. Nevertheless, Sensitivity of the second model higher (91% compare to original 88%) and the improved model classified correctly higher number of Liver patients (75 compare to original 72). Precision on the class 1 (positive) was higher too (0.7947 compare to 0.7912 in the original CART) original which means less false positive cases.

Although the Original CART model is better, I'd recommend using the Improved CART for medical use. Better sensitivity means more patients with liver disease will be classified correctly and those will take examined further by doctors.

Appendices

Appendix A: Database Connection

```
library(dplyr)

library(RMySQL)

library(ggplot2)

library(tidyr)

library(caret)

library(C50)

library(rpart.plot)

library(MLmetrics)

library(pROC)

library(adabag)


########## Database Connection #############

con <- DBI::dbConnect(RMySQL::MySQL(),

                host = "host.address.com",

                user = "username",

                dbname="dbname",

                port=2222,

                password = "passwords")

dbListTables(con)
```

Appendix B: Code Question 1

```
################################ QUESTION 1 ########################################

##### TASK 1 : ISPECT WHAT YOU HAVE LOADED ######

#Create table Cars
dbSendQuery(con, "
 CREATE TABLE Cars (
 ID INT PRIMARY KEY,
 mpg FLOAT,
 cylinders FLOAT,
 displacement FLOAT,
 horsepower FLOAT,
 weight FLOAT,
 acceleration FLOAT,
 model FLOAT,
 origin FLOAT,
 car_name VARCHAR(250),
 price FLOAT);")

#Load Data from CSV File
cars_info <- read.csv("C:/Windows/Temp/cars_info.csv")

#Write data into database
dbWriteTable(con, name='Cars', value=cars_info, overwrite=TRUE)

#Displays names of the columns in the table Car
dbListFields(con, 'Cars')

#Display all Data
```

```
result <- dbSendQuery(con, 'SELECT * From Cars')

data <- fetch(result, n=-1)

data


########### TASK 2 : RMySQL QUERIES #####################

#1 Get the first 10 rows in the imported table

result <- dbSendQuery(con, 'SELECT * From Cars WHERE ID < 11')

data <- fetch(result, n=-1)

data


#2 Get all eight-cylinder cars with miles per gallon greater than 18

result <- dbSendQuery(con, 'SELECT * From Cars WHERE cylinders = 8 AND
                    mpg > 18')

data <- fetch(result, n=-1)

data


#3 Get the average horsepower and mpg by number of cylinder groups

result <- dbSendQuery(con, 'SELECT cylinders, AVG(horsepower), AVG(mpg) From Cars Group by
cylinders')

data <- fetch(result, n=-1)

data


#4 Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive both
limits)

result <- dbSendQuery(con, 'SELECT * From Cars WHERE cylinders < 8 AND (acceleration >= 11 and
acceleration <= 13)')

data <- fetch(result, n=-1)

data


#5 Get the car names and horsepower of the cars with 3 cylinders

result <- dbSendQuery(con, 'SELECT car_name, horsepower From Cars WHERE cylinders = 3 ')

data <- fetch(result, n=-1)
```

```
data



########### TASK 2 dplyr QUERIES #####################

#saves table car to data structure cars_db

cars_db <- tbl(con, "Cars")


#set minimum of printed tibble rows to 100

options( tibble.print_min = 100)


#1 Get the first 10 rows in the imported table

cars_db %>% filter(ID < 11)


#2 Get all eight-cylinder cars with miles per gallon greater than 18

cars_db %>% filter(cylinders == 8, mpg > 18)


#3 Get the average horsepower and mpg by number of cylinder groups

cars_db %>% group_by(cylinders) %>% summarise(avg(mpg), avg(horsepower))


#4 Get all cars with less than eight-cylinder and with acceleration from 11 to 13 (inclusive both
limits)

cars_db %>% filter(cylinders < 8, acceleration <= 13, acceleration >=11 )


#5 Get the car names and horsepower of the cars with 3 cylinders

cars_db %>% filter(cylinders == 3) %>% select(car_name, horsepower)



#################### TASK 3 : VISUALISATION #########################

#saves table car to data structure cars_db

cars_db <- tbl(con, "Cars")
```

```
####### 1.1 distribution of values for cylinders
#retrieve data frame of cylinders
cylinders<-cars_db %>% select(cylinders) %>% data.frame
hist(cylinders$cylinders, main = "Histogram of Cylinders",
    xlab = "Number of cylinders", col='light blue')

#Display table showing number of models by cylinder
cars_db %>% group_by(cylinders) %>% summarise(count(cylinders))

#average per year
aCYL <- cars_db %>% group_by(model) %>% summarise(avg(cylinders)) %>% data.frame

####### 1.2 distribution of values for Mpg

mpg<-cars_db %>% select(mpg) %>% data.frame
hist(mpg$mpg, main = "Histogram of Miles per gallon",
    xlab = "Number of Miles per gallon", col='light blue')

#cars with higher fuel consumption than 10mpg
cars_db %>% filter(mpg <= 10)

#average per year
aMPG <- cars_db %>% group_by(model) %>% summarise(avg(mpg)) %>% data.frame

#Relationship between
CYLMPG <- merge( aCYL,aMPG,by="model")
scatter.smooth( CYLMPG$avg.cylinders., CYLMPG$avg.mpg., xlab= "Average cylinders",
            ylab = " Average Mpg")
abline(lm(CYLMPG$avg.mpg. ~ CYLMPG$avg.cylinders.), col="red")
```

```
####### 2 boxplot to show the mean and distribution of Mpg measurements for each year
cars_db <- tbl(con, "Cars")

#select value for each year
y70<-cars_db %>% filter(model == 70) %>% select(mpg) %>% data.frame
y71<-cars_db %>% filter(model == 71) %>% select(mpg) %>% data.frame
y72<-cars_db %>% filter(model == 72) %>% select(mpg) %>% data.frame
y73<-cars_db %>% filter(model == 73) %>% select(mpg) %>% data.frame
y74<-cars_db %>% filter(model == 74) %>% select(mpg) %>% data.frame
y75<-cars_db %>% filter(model == 75) %>% select(mpg) %>% data.frame
y76<-cars_db %>% filter(model == 76) %>% select(mpg) %>% data.frame
y77<-cars_db %>% filter(model == 77) %>% select(mpg) %>% data.frame
y78<-cars_db %>% filter(model == 78) %>% select(mpg) %>% data.frame
y79<-cars_db %>% filter(model == 79) %>% select(mpg) %>% data.frame
y80<-cars_db %>% filter(model == 80) %>% select(mpg) %>% data.frame
y81<-cars_db %>% filter(model == 81) %>% select(mpg) %>% data.frame
y82<-cars_db %>% filter(model == 82) %>% select(mpg) %>% data.frame

#display boxplot
boxplot(y70$mpg,y71$mpg,y72$mpg,y73$mpg,y74$mpg,y75$mpg,y76$mpg,y77$mpg,y78$mpg,y79$mpg,
    y80$mpg,y81$mpg,y82$mpg,
    xlab = "Mean and distribution of mpg for each yaer",
    ylab = "Miles per gallon",
    names=c(1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982),
    col="light blue")

####### 3 scatter plot showing the relationship between weight and Mpg
```

```
cars_db <- tbl(con, "Cars")


cars_table <- dbReadTable(con, "Cars")
scatter.smooth( cars_table$weight, cars_table$mpg, xlab= "Weight (Lbs.)",
          ylab = "Consuption Mpg", col="blue")
abline(lm(cars_table$mpg ~cars_table$weight ), col="red")
```

Appendix C: Code Question 2

```
########################## GENERATE DATA ########################################

#Create State Table

state_table <-

 data.frame(key=c("FR", "LA", "SY", "SE", "CT"),

      name=c("Frankfurt", "Los Angeles", "Sydney", "Seoul", "Cape Town"),

      country=c("Germany", "USA", "Australia", "S. Korea", "South Africa"))


# Create Month Table

month_table <-

 data.frame(key=1:12,

      desc=c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),

      quarter=c("Q1","Q1","Q1","Q2","Q2","Q2","Q3","Q3","Q3","Q4","Q4","Q4"))


# Create Product Table

prod_table <-

 data.frame(key=c("Washing Machine", "Fridge", "Vacuum Cleaner", "Microwave Oven"),

      price=c(500, 150, 400, 200), #Values are shifted (see reprot)

      stringsAsFactors = TRUE)



#########Generate randomly 500 samples within 6 years

# Define funcition

gen_sales <- function(no_of_recs) {

 # Generate transaction data randomly

 loc <- sample(state_table$key, no_of_recs, replace=T, prob=c(2,2,1,1,1))

 time_month <- sample(month_table$key, no_of_recs, replace=T)

 time_year <- sample(c(2015, 2016, 2017, 2018, 2019, 2020), no_of_recs, replace=T)

 prod <- sample(prod_table$key, no_of_recs, replace=T)

 unit <- sample(c(1,2), no_of_recs, replace=T, prob=c(10, 4))

 amount <- unit*prod_table[prod,]$price
```

```
  sales <- data.frame(month=time_month,
                     year=time_year,
                     loc=loc,
                     prod=prod,
                     unit=unit,
                     amount=amount)
  # Sort the records by time order
  sales <- sales[order(sales$year, sales$month),]
  row.names(sales) <- NULL
  return(sales)
}

#Generate Samples
sales_fact <- gen_sales(500)

head(sales_fact)



##### Save sales_fact to Database
dbWriteTable(con, name='Sales_CWK2', value=sales_fact, overwrite=TRUE)

####################### LOAD FROM DATABASE ####################################

salesDB <- tbl(con, 'Sales_CWK2')



sales_table <- salesDB %>% select(month, year, loc, prod, unit, amount) %>% data.frame



##### Create Revenue Cube ####
```

```
revenue_cube <-
  tapply(sales_table$amount,
         sales_table[,c("prod", "month", "year", "loc")],
         FUN=function(x){return(sum(x))})



# Display Revenue Cube
revenue_cube


# Display dimension names
dimnames(revenue_cube)



######################## OLAP OPERATIONS ########################################


#######Slice
  revenue_cube[, "1", "2016",]


  revenue_cube["Fridge", "1", "2017",]


#######Dice


  revenue_cube[c("Washing Machine", "Fridge"),
            c("1","2",'5'), ,
            c("SY","LA")]


    revenue_cube[c("Washing Machine", "Vacuum Cleaner"),
             c("1","2",'12'), ,
             c("SE","LA", "CT")]


#######Roll-up
```

```
apply(revenue_cube, c("year", "prod"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
apply(revenue_cube, c("year", "prod"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
#######Drill-down
apply(revenue_cube, c("year", "month", "prod"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
apply(revenue_cube, c("year",  "month", "loc"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

```
#########Pivot
apply(revenue_cube, c("year", "month"),
    FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

Appendix D: Code Question 3

```
################# IMPORT DATASET ##############

#Load data from a file
liver_file = read.csv("C:/Users/w1701833/Desktop/Business Intelligence/liver.csv")

#Upload data to database
dbWriteTable(con, name='Liver', value=liver_file)

#Display table names
dbListTables(con)

#Display Fileds in Liver table
dbListFields(con, 'Liver')

################# LOAD DATA FROM DATABASE ########
liver_table <- tbl(con, "Liver") %>% data.frame

head(liver_table)

################# DATA PREPARATION ################

#Drop all Observations with NULL value
liver_filtered <- liver_table %>% drop_na()



# Replace gender with numerical values
liver_filtered[liver_filtered == "Male"] <-1
liver_filtered[liver_filtered == "Female"] <-2

anyNA(liver_filtered)
```

```
# [1] FALSE

#Save working data set into database#
dbWriteTable(con, name='Liver_worksample', value=liver_filtered[-1], overwrite=TRUE)

################# DATA ANALYTICS ###################
liverDB <- tbl(con, 'Liver_worksample')

#Mean(avg) age of subjects
liverDB %>% group_by(gender) %>% summarise(avg(age))

#Median  and quartiles of subject ages
female_age <-liverDB %>% select(age, gender) %>% filter(gender == 2) %>% data.frame
male_age <-liverDB %>% select(age, gender) %>% filter(gender == 1) %>% data.frame

female_Q <- quantile(female_age$age, probs=c(.25, .5, .75), na.rm = FALSE)
male_Q <- quantile(male_age$age, probs=c(.25, .5, .75), na.rm = FALSE)

#Display Values
female_Q
male_Q

################# DATA VISUALISATION #############
liverDB <- tbl(con, 'Liver_worksample')

###Perform a histogram of the frequency of patients per age
  #check number of groups
  liverDB %>% group_by(age) %>% summarise(count(age))%>% data.frame
  #extract age data
  agehist<-liverDB %>% select(age) %>% data.frame
```

```
#display histogram (step 1)
hist(agehist$age, breaks = seq(1,100, by=1), xlim = c(0,100),
    col="light blue", xlab="Age (Step by 1)", main = "Histogram of patients age (step by 1)")

#display histogram (step 3)
hist(agehist$age, breaks = seq(1,100, by=3), xlim = c(0,100),  ylim = c(0,50),
    col="light blue", xlab="Age (Step by 3)", main = "Histogram of patients age (step by 3)")

#display ggplot histogram
ggplot(agehist, aes(age)) + geom_histogram(binwidth=3, fill="light blue", color="black")



###Perform a histogram of the frequency of patients per sgpt
#check number of groups
liverDB %>% group_by(sgpt) %>% summarise(count(sgpt))%>% data.frame

#extract data
sgpthist<-liverDB %>% select(sgpt) %>% data.frame

#display histogram
  hist(sgpthist$sgpt, breaks = seq(1,10, by=.1), xlim = c(2,10), ylim = c(0,35),
    col="light blue", xlab="Alamine Aminotransferase (Step by 0.1)",
    main = "Histogram of Alamine Aminotransferase (step by 0.1)")

hist(sgpthist$sgpt, breaks = seq(1,10, by=.2), xlim = c(2,10), ylim = c(0,60),
    col="light blue", xlab="Alamine Aminotransferase (Step by 0.2)",
    main = "Histogram of Alamine Aminotransferase (step by 0.2)")

#ggplot histogram
  ggplot(sgpthist, aes(sgpt)) + geom_histogram(binwidth=.1, fill="light blue", color="black")
```

```
###Perform a boxplot of Gender (Male & Female) vs age
 boxplot(female_age$age, male_age$age, names=c("Female","Male"),
        col = "light blue", horizontal = TRUE)



################# GENERATE TRAINING AND TESTING DATASET ########################
 #table from database
 liverDB <- tbl(con, 'Liver_worksample')

 #save to dataframe
 liver_ <- liverDB %>% data.frame

 #Slice data for testing and training sample
 set.seed(3003)
 intrain <- createDataPartition(y = liver_$is_patient, p= 0.8, list = FALSE)
 training <- liver_[intrain,]
 testing <- liver_[-intrain,]

 #save training and testing samples to the database
 dbWriteTable(con, name='Liver_training', value=training[-1], overwrite=TRUE)
 dbWriteTable(con, name='Liver_testing', value=testing[-1], overwrite=TRUE)

 dim(training) #464
 dim(testing)  #115

################# C5.0 MODEL DECISION TREE #####################################

 #Load data
 liver_train <- tbl(con, 'Liver_training')
 liver_test <- tbl(con, 'Liver_testing')
```

```
training_set <- liver_train %>% data.frame

training_set <- training_set[-1] #%>% data.frame


testing_set <- liver_test %>%

select(age,gender,tot_bilirubin,direct_bilirubin,tot_proteins,albumin,ag_ratio,sgpt,sgot,alkphos,is_patient) %>%
  data.frame



#Convert data to factor

training_set$is_patient <- as.factor(training_set$is_patient)


#set.seed(2345)

model <- C5.0(is_patient ~., data=training_set )


plot(model)



#test treee

results <- predict(object=model, newdata=testing_set, type="class")


#evaluation

#table(results, testing_set$is_patient)

tt <- as.factor(testing_set$is_patient)

confusionMatrix(results, tt)


#F1 Score

F1_Score(tt, results)
```

```
#ROC

C5ROC <- roc(results, testing_set$is_patient, plot = TRUE)


auc(C5ROC)


################ CART MODEL DECISION TREE ###################################


###Load data

liver_train <- tbl(con, 'Liver_training')

liver_test <- tbl(con, 'Liver_testing')




training_set <- liver_train %>% data.frame

training_set <- training_set[-1] #%>% data.frame


testing_set <- liver_test %>%

select(age,gender,tot_bilirubin,direct_bilirubin,tot_proteins,albumin,ag_ratio,sgpt,sgot,alkphos,is_p
atient) %>%
  data.frame


# Train tree

set.seed(3333)

model <- rpart(is_patient~.,data=training_set,method="class")

model


plot(model)

text(model, digits = 3)


# test tree

predicted.classes <- model %>% predict(testing_set, type = "class")

head(predicted.classes)
```

```
mean(predicted.classes == testing_set$is_patient)



#evaluate tree
tt <- as.factor(testing_set$is_patient)
confusionMatrix(predicted.classes, tt)

#F1 Score
F1_Score(predicted.classes, tt)

#ROC
cartROC <- roc(predicted.classes, testing_set$is_patient, plot = TRUE)

auc(cartROC)

################# PRUNING AND IMPROVEMENT  ################################

###Load data
liver_train <- tbl(con, 'Liver_training')
liver_test <- tbl(con, 'Liver_testing')



training_set <- liver_train %>% data.frame
training_set <- training_set[-1] #%>% data.frame

testing_set <- liver_test %>%
select(age,gender,tot_bilirubin,direct_bilirubin,tot_proteins,albumin,ag_ratio,sgpt,sgot,alkphos,is_patient) %>%
  data.frame
```

```
# Train tree
set.seed(3333)

rctr <- rpart.control(maxdepth = 10 , minsplit = 10)
model <- rpart(is_patient~.,data=training_set,method="class", control = rctr)



plot(model)
text(model, digits = 3)

# test tree
predicted.classes <- model %>% predict(testing_set, type = "class")
head(predicted.classes)

mean(predicted.classes == testing_set$is_patient)



#evaluate tree
tt <- as.factor(testing_set$is_patient)
confusionMatrix(predicted.classes, tt)

#F1 Score
F1_Score(predicted.classes, tt)

  #ROC
cartROC <- roc(predicted.classes, testing_set$is_patient, plot = TRUE)

auc(cartROC)
```