These are the points which then have to be processed into a cheat sheet.

1. table(Title)#group by and count/length
2. table(tit\_train$Sex,tit\_train$Title) #equivalent to groupby with implicit count/length using a case statement
3. tapply(tit\_train$Sex,tit\_train$Title,length) # this is equivalent to table(Title). The equivalent group has implicit count/length without case statement
4. update a column using a where clause

tit\_train$Title[tit\_train$Title == 'Mlle'] <- 'Miss'

tit\_train$Title[tit\_train$Title %in% rare\_title] <- 'Rare Title'

1. subset(tit\_train,subset = !is.na(tit\_train$Age)) # equivalent to filtering a table using where clause.
2. Projection or selecting some column (using column names) from a dataset:

subset(auto, select = -c(name))

if we were dropping just by using indexes, then this would have worked:

df[,-3]

1. Filtering a matrix (like on returned by cor) can be done in various ways:

auto = ISLR::Auto

mat = cor(auto\_sb)

mat>.4 # returns a matrix with True/False values in the matrix

mat[mat>.4] # this returns a vector of values satisfying the condition

subset(mat,mat>.4) # this returns a vector of values satisfying the condition

which(mat>.4, arr.ind = T) #This give a matrix with values being indexes satisfying the condition

mat[mat>.4] = 1 #i could also set the values like this which is equivalent to true/false.

mat[mat<=.4] = 0 #i could also set the values like this which is equivalent to true/false.

1. is.element(FALSE, is.na(Auto$year)) checks whether a value belongs to a set. In SQL one can use IN keyword to form a similar condition. Actually, in R as well there you have in:

FALSE %in% is.na(Auto$year)