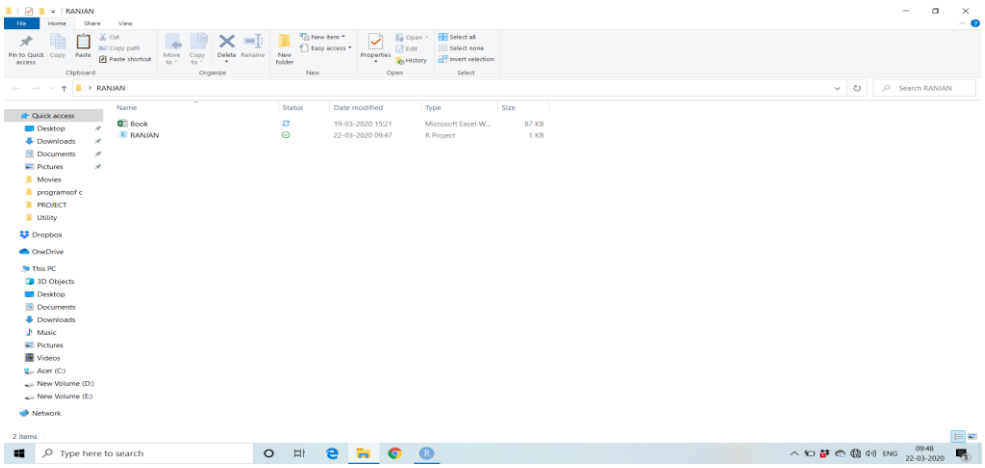
Phase-2

DATA CLEANING PROCESS

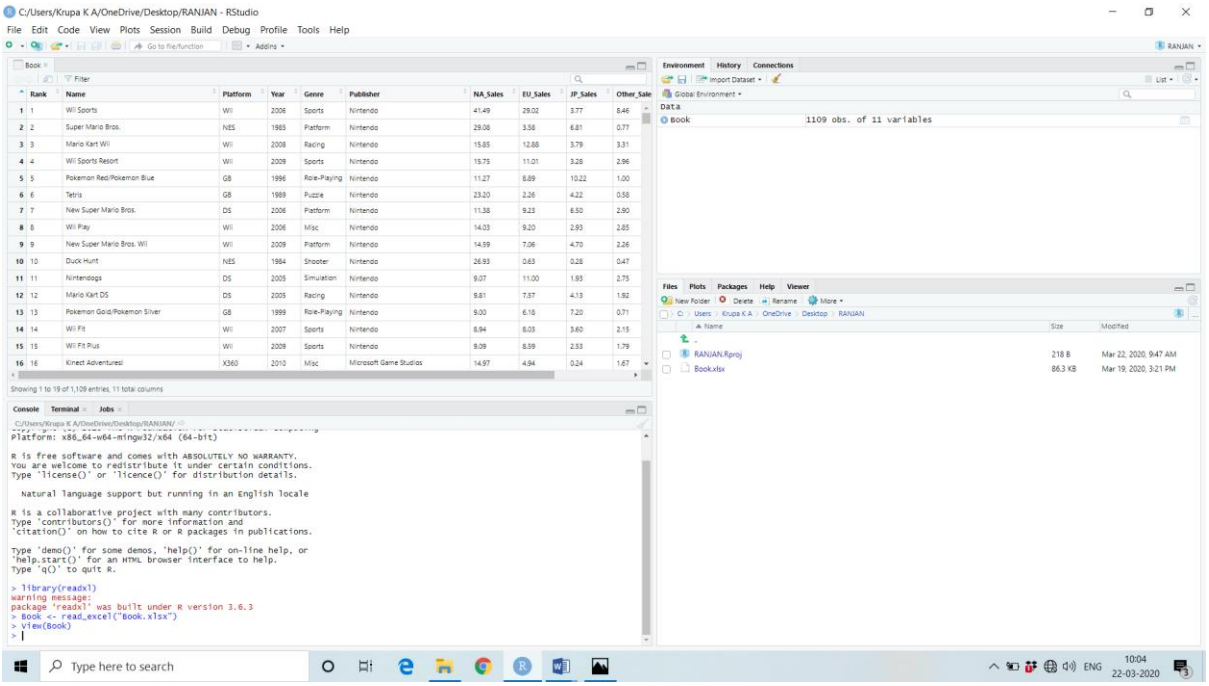
Objectives:-

1.First copy dataset into working directory/folder



2.Import the dataset present in the working directory into R-Studio:- Code:-

```
library(readxl)
 Book <- read_excel("Book.xlsx")
 View(Book)
```



summary(Book)

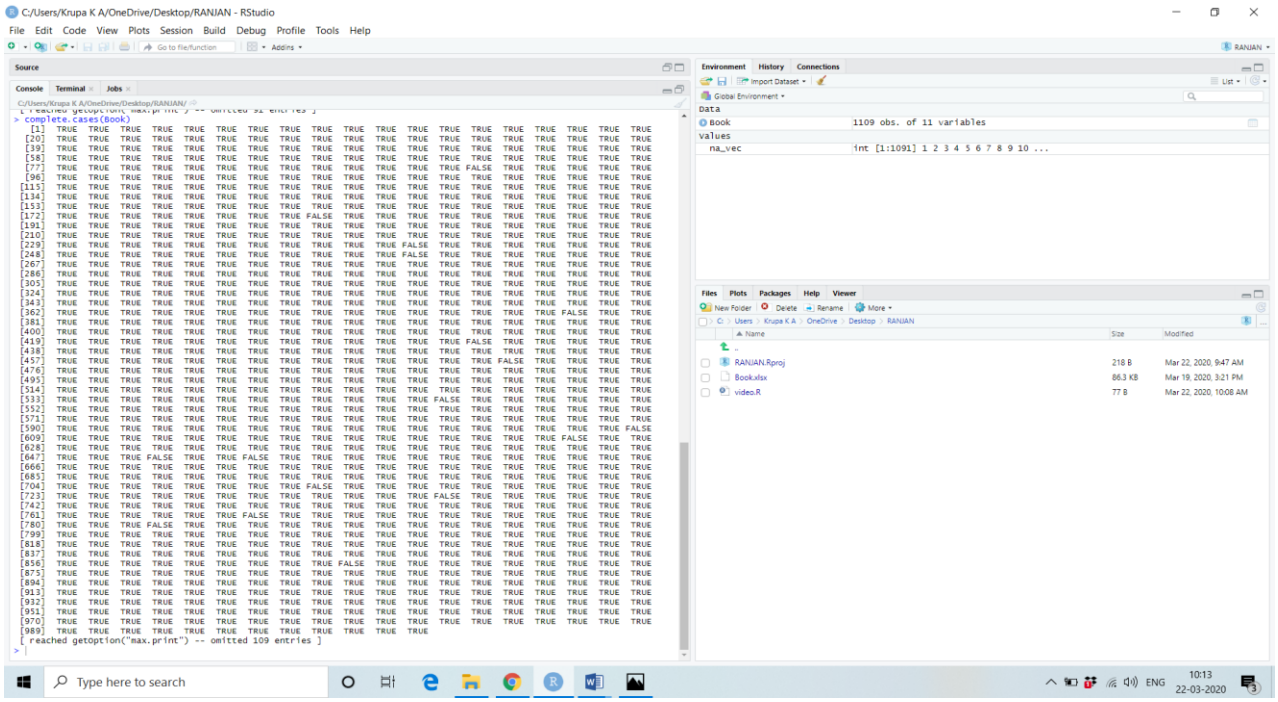
```
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.6.3
> Book <- read_excel("Book.xlsx")
> View(Book)
> View(Book)
> summary(Book)
      Rank            Name             Platform             Year         Genre             Publisher
 Min.   :   1.0   Length:1109        Length:1109        Min.   :1980   Length:1109        Length:1109
 1st Qu.: 278.0   Class :character   Class :character   1st Qu.:2001   Class :character   Class :character
 Median : 555.0   Mode  :character   Mode  :character   Median :2006   Mode  :character   Mode  :character
 Mean   : 555.4                                        Mean   :2005
 3rd Qu.: 833.0                                        3rd Qu.:2010
 Max.   :1110.0                                        Max.   :2016
                                                       NA's   :10
    NA_Sales          EU_Sales          JP_Sales        Other_Sales        Global_Sales
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 1.650
 1st Qu.: 0.920   1st Qu.: 0.460   1st Qu.: 0.0000   1st Qu.: 0.1200   1st Qu.: 2.040
 Median : 1.420   Median : 0.830   Median : 0.0600   Median : 0.2300   Median : 2.700
 Mean   : 1.965   Mean   : 1.183   Mean   : 0.5093   Mean   : 0.3777   Mean   : 4.035
 3rd Qu.: 2.120   3rd Qu.: 1.350   3rd Qu.: 0.5600   3rd Qu.: 0.4100   3rd Qu.: 4.160
 Max.   :41.490   Max.   :29.020   Max.   :10.2200   Max.   :10.5700   Max.   :82.740

>
```

3. Data cleaning process should be done after importing

3.1 Find Complete Cases. Return a logical vector indicating which cases are complete, i.e., have no missing values.

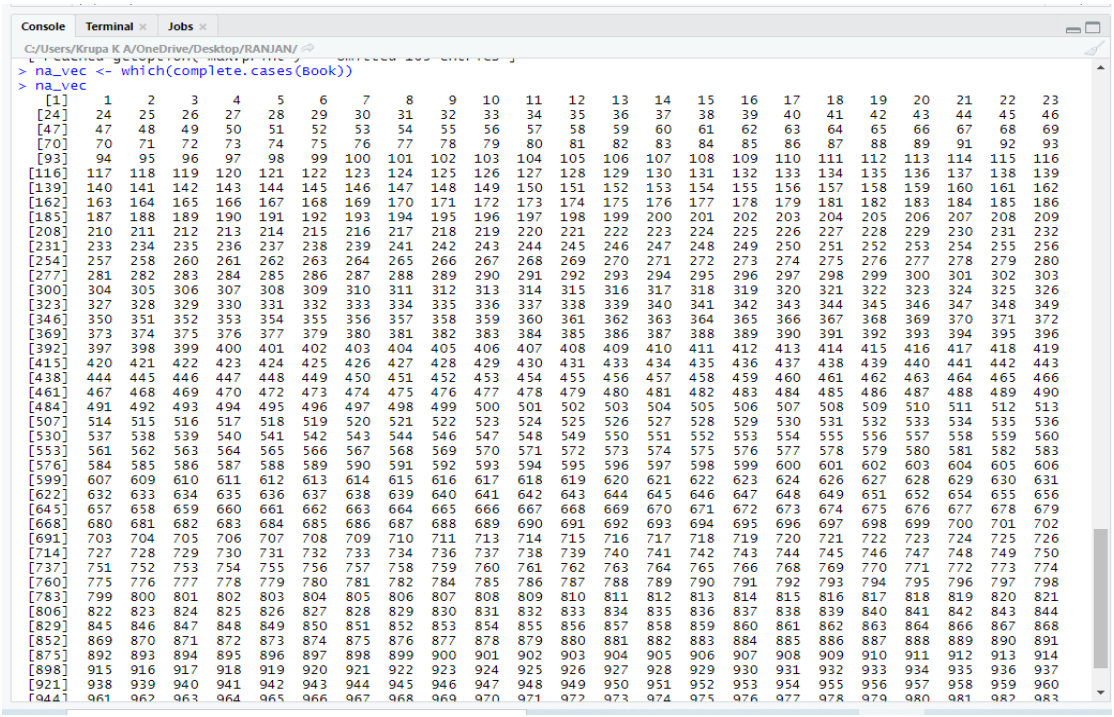
Complete.cases(Book)



3.2 Using **which(complete.cases())** return a numerical values indicating which cases are complete, i.e., have no missing values.

na_vec <- which(complete.cases(Book))

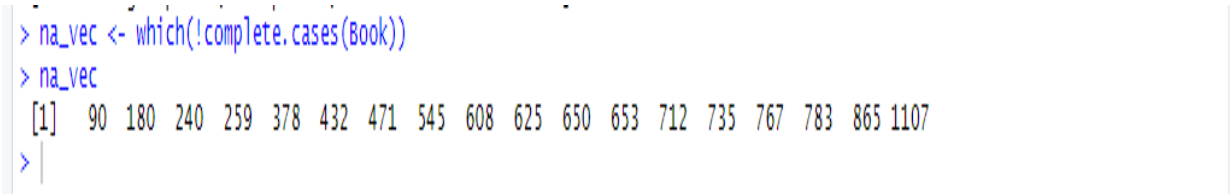
na_vec



3.3 Using **which(!complete.cases())** return a numerical values indicating which cases are not complete or incomplete , i.e., have missing values.

na_vec <- which(!complete.cases(Book))

na_vec



3.4 The **na**. **omit R** function removes all incomplete cases of a data object (typically of a data frame, matrix or vector). The syntax below illustrates the basic programming code for **na**.

Book_na_omit <- na.omit(Book)

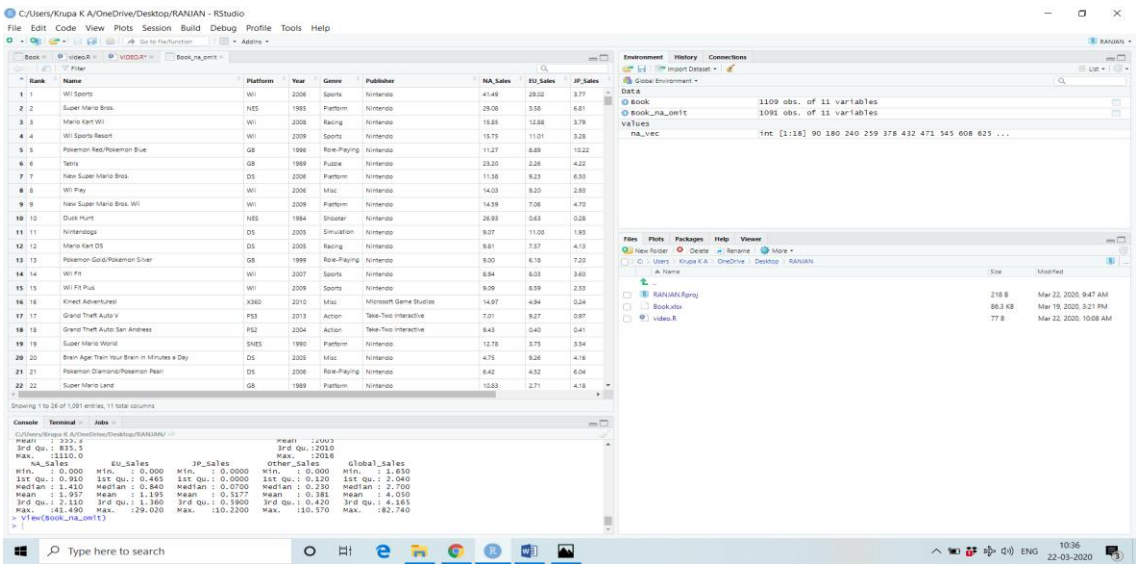
Book_na_omit

```
> Book_na_omit <- na.omit(Book)
> Book_na_omit
# A tibble: 1,091 x 11
    Rank Name                 Platform  Year Genre      Publisher NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
   <dbl> <chr>                <chr>     <dbl> <chr>      <chr>       <dbl>    <dbl>    <dbl>       <dbl>        <dbl>
 1     1 Wii Sports           Wii        2006 Sports     Nintendo     41.5     29.0     3.77        8.46         82.7
 2     2 Super Mario Bros.    NES        1985 Platform   Nintendo     29.1      3.58    6.81        0.77         40.2
 3     3 Mario Kart Wii       Wii        2008 Racing     Nintendo     15.8     12.9     3.79        3.31         35.8
 4     4 Wii Sports Resort    Wii        2009 Sports     Nintendo     15.8     11.0     3.28        2.96         33
 5     5 Pokemon Red/Pokemon Bl~ GB      1996 Role-Playi~ Nintendo     11.3      8.89   10.2         1            31.4
 6     6 Tetris               GB         1989 Puzzle     Nintendo     23.2      2.26    4.22        0.580        30.3
 7     7 New Super Mario Bros. DS        2006 Platform   Nintendo     11.4      9.23    6.5         2.9          30.0
 8     8 Wii Play             Wii        2006 Misc       Nintendo     14.0      9.2     2.93        2.85         29.0
 9     9 New Super Mario Bros. ~ Wii     2009 Platform   Nintendo     14.6      7.06    4.7         2.26         28.6
10    10 Duck Hunt            NES        1984 Shooter    Nintendo     26.9      0.63    0.28        0.47         28.3
# ... with 1,081 more rows
> |
```

summary(Book_na_omit)

```
> summary(Book_na_omit)
      Rank            Name              Platform              Year         Genre             Publisher
 Min.   :   1.0   Length:1091        Length:1091        Min.   :1983   Length:1091        Length:1091
 1st Qu.: 277.5   Class :character   Class :character   1st Qu.:2001   Class :character   Class :character
 Median : 554.0   Mode  :character   Mode  :character   Median :2006   Mode  :character   Mode  :character
 Mean   : 555.3                                        Mean   :2005
 3rd Qu.: 835.5                                        3rd Qu.:2010
 Max.   :1110.0                                        Max.   :2016
    NA_Sales         EU_Sales         JP_Sales        Other_Sales       Global_Sales
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000   Min.   : 1.650
 1st Qu.: 0.910   1st Qu.: 0.465   1st Qu.: 0.0000   1st Qu.: 0.120   1st Qu.: 2.040
 Median : 1.410   Median : 0.840   Median : 0.0700   Median : 0.230   Median : 2.700
 Mean   : 1.957   Mean   : 1.195   Mean   : 0.5177   Mean   : 0.381   Mean   : 4.050
 3rd Qu.: 2.110   3rd Qu.: 1.360   3rd Qu.: 0.5900   3rd Qu.: 0.420   3rd Qu.: 4.165
 Max.   :41.490   Max.   :29.020   Max.   :10.2200   Max.   :10.570   Max.   :82.740
> |
```

View(Book_na_omit)



The above dataset is cleaned data with no null value(no empty rows and empty column).

3.5 The **R** function **duplicated**() returns a logical vector where TRUE specifies which elements of a vector or data frame are **duplicates**

`duplicated(Book_na_omit)`

```
C:/Users/Krupa K A/OneDrive/Desktop/RANJAN/
> View(Book_na_omit)
> duplicated(Book_na_omit)
   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [20] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [39] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [58] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [77] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [96] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [115] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [134] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [153] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [172] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [191] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [210] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [248] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [267] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [286] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [305] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [324] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [343] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [362] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [381] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [400] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [419] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [438] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [476] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [495] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [514] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [533] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [552] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [571] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [590] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [609] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [628] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [647] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [666] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [685] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [704] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [723] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [742] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [761] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [780] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [799] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [818] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [837] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [856] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [875] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [894] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [913] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [932] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [951] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [970] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [989] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [ reached getOption("max.print") -- omitted 91 entries ]
>
```

3.6 ! is a logical negation. ! **duplicated**() means that we don't want **duplicate** rows

which(!duplicated(Book_na_omit))

```
Console   Terminal   Jobs
C:/Users/Krupa K A/OneDrive/Desktop/RANJAN/
> which(!duplicated(Book_na_omit))
   [1]    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
  [24]   24   25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46
  [47]   47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69
  [70]   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92
  [93]   93   94   95   96   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115
 [116]  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138
 [139]  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161
 [162]  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184
 [185]  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207
 [208]  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230
 [231]  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
 [254]  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276
 [277]  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299
 [300]  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322
 [323]  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345
 [346]  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368
 [369]  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391
 [392]  392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414
 [415]  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437
 [438]  438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460
 [461]  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483
 [484]  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506
 [507]  507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529
 [530]  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550  551  552
 [553]  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575
 [576]  576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596  597  598
 [599]  599  600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616  617  618  619  620  621
 [622]  622  623  624  625  626  627  628  629  630  631  632  633  634  635  636  637  638  639  640  641  642  643  644
 [645]  645  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663  664  665  666  667
 [668]  668  669  670  671  672  673  674  675  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690
 [691]  691  692  693  694  695  696  697  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713
 [714]  714  715  716  717  718  719  720  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736
 [737]  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751  752  753  754  755  756  757  758  759
 [760]  760  761  762  763  764  765  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780  781  782
 [783]  783  784  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800  801  802  803  804  805
 [806]  806  807  808  809  810  811  812  813  814  815  816  817  818  819  820  821  822  823  824  825  826  827  828
 [829]  829  830  831  832  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849  850  851
 [852]  852  853  854  855  856  857  858  859  860  861  862  863  864  865  866  867  868  869  870  871  872  873  874
 [875]  875  876  877  878  879  880  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896  897
 [898]  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920
 [921]  921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943
 [944]  944  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960  961  962  963  964  965  966
 [967]  967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989
 [990]  990  991  992  993  994  995  996  997  998  999 1000
 [ reached getOption("max.print") -- omitted 91 entries ]
>
```

3.7 Using **which(duplicated())** return a numerical values indicating which cases are duplicated.

which(duplicated(Book_na_omit))

```
> which(duplicated(Book_na_omit))
integer(0)
>
```

As per the above output there is no duplicated value in dataset

Conclusion:-

The dataset we have taken is cleaned by eliminating null values and duplicated values.