<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Import packages</td><td>

```{r echo=FALSE}
#Import libraries
library(dplyr)
library(reshape2)
library(data.table)
library(tidyr)
library(reshape2)
```

</td><td>

```
#Import packaages
import pandas as pd
import numpy as np
```

</td></tr>
<tr><td>Read csv file as a dataframe</td><td>The head function returns the first 10 rows, by default. The tail function can be used to filter out the rows from the end.
read.csv(“student-mat.csv”)
head(student_score)

```{r}
#Read csv file
student_score = read.csv("student-mat.csv")
head(student_score)
```

	school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>
1	GP	F	18	U	GT3	A	4	4	at_home
2	GP	F	17	U	GT3	T	1	1	at_home
3	GP	F	15	U	LE3	T	1	1	at_home
4	GP	F	15	U	GT3	T	4	2	health
5	GP	F	16	U	GT3	T	3	3	other
6	GP	M	16	U	LE3	T	4	3	services

6 rows | 1-10 of 33 columns

</td><td>The head function returns the first 5 rows, by default. The tail function can be used to filter out the rows from the end.
pd.read_csv(“student-mat.csv”)
student_score.head()

```
#Read csv file
student_score = pd.read_csv('student-mat.csv')
student_score.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

5 rows × 33 columns

</td></tr>
<tr><td>Read a csv as a datatable</td><td>Data tables are used for large datasets and quick data wrangling operations. Data tables are faster than base R or dplyr functions.
fread(“student-mat.csv”)

```{r}
#Read csv file as a data table
student_score_dt = fread("student-mat.csv")
head(student_score_dt)
```

school <chr>	sex <chr>	age <int>	address <chr>	famsize <chr>	Pstatus <chr>	Medu <int>	Fedu <int>	Mjob <chr>	Fjob <chr>
GP	F	18	U	GT3	A	4	4	at_home	teacher
GP	F	17	U	GT3	T	1	1	at_home	other
GP	F	15	U	LE3	T	1	1	at_home	other
GP	F	15	U	GT3	T	4	2	health	services
GP	F	16	U	GT3	T	3	3	other	other
GP	M	16	U	LE3	T	4	3	services	other

6 rows | 1-10 of 33 columns

</td><td>Currently, Python does not support data tables for Windows.</td></tr>
<tr><td>Dataframe Dimension</td><td>Dim returns the no. of rows and columns.
dim(student_score)

```{r}
#Dataset dimension
dim(student_score)
```

```
[1] 395  33
```

</td><td>Shape returns the no. of rows and columns.
student_score.shape

```
#Dataset dimension
student_score.shape
```

```
(395, 33)
```

</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Dataset summary</td><td>str(student_score)

```{r}
#Dataset sumamry
str(student_score_dt)
```

```
Classes 'data.table' and 'data.frame':  395 obs. of  33 variables:
 $ school    : chr  "GP" "GP" "GP" "GP" ...
 $ sex       : chr  "F" "F" "F" "F" ...
 $ age       : int  18 17 15 15 16 16 16 17 15 15 ...
 $ address   : chr  "U" "U" "U" "U" ...
 $ famsize   : chr  "GT3" "GT3" "LE3" "GT3" ...
 $ Pstatus   : chr  "A" "T" "T" "T" ...
 $ Medu      : int  4 1 1 4 3 4 2 4 3 3 ...
 $ Fedu      : int  4 1 1 2 3 3 2 4 2 4 ...
 $ Mjob      : chr  "at_home" "at_home" "at_home" "health" ...
 $ Fjob      : chr  "teacher" "other" "other" "services" ...
 $ reason    : chr  "course" "course" "other" "home" ...
 $ guardian  : chr  "mother" "father" "mother" "mother" ...
 $ traveltime: int  2 1 1 1 1 1 1 2 1 1 ...
 $ studytime : int  2 2 2 3 2 2 2 2 2 2 ...
 $ failures  : int  0 0 3 0 0 0 0 0 0 0 ...
 $ schoolsup : chr  "yes" "no" "yes" "no" ...
 $ famsup    : chr  "no" "yes" "no" "yes" ...
 $ paid      : chr  "no" "no" "yes" "yes" ...
 $ activities: chr  "no" "no" "no" "no" ...
 $ nursery   : chr  "yes" "no" "yes" "yes" ...
 $ higher    : chr  "yes" "yes" "yes" "yes" ...
 $ internet  : chr  "no" "yes" "yes" "yes" ...
 $ romantic  : chr  "no" "no" "no" "yes" ...
 $ famrel    : int  4 5 4 3 4 5 4 4 4 5 ...
 $ freetime  : int  3 3 3 2 3 4 4 1 2 5 ...
 $ goout     : int  4 3 2 2 2 2 4 4 2 1 ...
 $ Dalc      : int  1 1 2 1 1 1 1 1 1 1 ...
 $ Walc      : int  1 1 3 1 2 2 1 1 1 1 ...
 $ health    : int  3 3 3 5 5 5 3 1 1 5 ...
 $ absences  : int  6 4 10 2 4 10 0 6 0 0 ...
```

</td><td>student_score.info

```
#Dataset sumamry
student_score.info
```

```
<bound method DataFrame.info of     school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob  \
0       GP   F   18       U     GT3       A     4     4   at_home   teacher
1       GP   F   17       U     GT3       T     1     1   at_home     other
2       GP   F   15       U     LE3       T     1     1   at_home     other
3       GP   F   15       U     GT3       T     4     2    health  services
4       GP   F   16       U     GT3       T     3     3     other     other
..     ...  ..  ...     ...     ...     ...   ...   ...       ...       ...
390     MS   M   20       U     LE3       A     2     2  services  services
391     MS   M   17       U     LE3       T     3     1  services  services
392     MS   M   21       R     GT3       T     1     1     other     other
393     MS   M   18       R     LE3       T     3     2  services     other
394     MS   M   19       U     LE3       T     1     1     other   at_home

     ... famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0    ...      4        3      4     1     1      3        6   5   6   6
1    ...      5        3      3     1     1      3        4   5   5   6
2    ...      4        3      2     2     3      3       10   7   8  10
3    ...      3        2      2     1     1      5        2  15  14  15
4    ...      4        3      2     1     2      5        4   6  10  10
..   ...    ...      ...    ...   ...   ...    ...      ...  ..  ..  ..
390  ...      5        5      4     4     5      4       11   9   9   9
391  ...      2        4      5     3     4      2        3  14  16  16
392  ...      5        5      3     3     3      3        3  10   8   7
```

</td></tr>
<tr><td>Summary Statistics</td><td>Summary statistics like Min, 1st quartile, Median, Mean, 3rd Quartile and Max values are returned.
summary(student_score)

```{r}
#Dataset statistics
summary(student_score)
```

```
     school   sex          age        address famsize   Pstatus      Medu            Fedu            Mjob     
 GP:349   F:208   Min.   :15.0   R: 88   GT3:281   A: 41   Min.   :0.000   Min.   :0.000   at_home : 59  
 MS: 46   M:187   1st Qu.:16.0   U:307   LE3:114   T:354   1st Qu.:2.000   1st Qu.:2.000   health  : 34  
                  Median :17.0                              Median :3.000   Median :2.000   other   :141  
                  Mean   :16.7                              Mean   :2.749   Mean   :2.522   services:103  
                  3rd Qu.:18.0                              3rd Qu.:4.000   3rd Qu.:3.000   teacher : 58  
                  Max.   :22.0                              Max.   :4.000   Max.   :4.000                 
      Fjob           reason      guardian     traveltime      studytime        failures      schoolsup 
 at_home : 20   course     :145   father: 90   Min.   :1.000   Min.   :1.000   Min.   :0.0000   no :344  
 health  : 18   home       :109   mother:273   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   yes: 51  
 other   :217   other      : 36   other : 32   Median :1.000   Median :2.000   Median :0.0000            
 services:111   reputation:105                Mean   :1.448   Mean   :2.035   Mean   :0.3342            
 teacher : 29                                 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.0000            
                                              Max.   :4.000   Max.   :4.000   Max.   :3.0000            
 famsup     paid     activities nursery   higher    internet  romantic      famrel         freetime    
 no :153   no :214   no :194    no : 81   no : 20   no : 66   no :263   Min.   :1.000   Min.   :1.000  
 yes:242   yes:181   yes:201    yes:314   yes:375   yes:329   yes:132   1st Qu.:4.000   1st Qu.:3.000  
                                                                       Median :4.000   Median :3.000  
                                                                       Mean   :3.944   Mean   :3.235  
                                                                       3rd Qu.:5.000   3rd Qu.:4.000  
                                                                       Max.   :5.000   Max.   :5.000  
     goout            Dalc            Walc          health         absences            G1       
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   : 0.000   Min.   : 3.00  
 1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:3.000   1st Qu.: 0.000   1st Qu.: 8.00  
 Median :3.000   Median :1.000   Median :2.000   Median :4.000   Median : 4.000   Median :11.00  
 Mean   :3.109   Mean   :1.481   Mean   :2.291   Mean   :3.554   Mean   : 5.709   Mean   :10.91  
 3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:5.000   3rd Qu.: 8.000   3rd Qu.:13.00  
 Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :75.000   Max.   :19.00  
       G2              G3       
 Min.   : 0.00   Min.   : 0.00  
```

</td><td>Summary statistics like Min, 1st quartile, Median, Mean, 3rd Quartile and Max values are returned.
student_score.describe()
student_score.mean() - #Returns only the mean of all the numerical columns.
Could use specific functions to return only the min, max or count of all the numeric columns.

```
#Dataset statistics
student_score.describe()
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	abs
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.0
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	3.235443	3.108861	1.481013	2.291139	3.554430	5.7
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	0.998862	1.113278	0.890741	1.287897	1.390303	8.0
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.0
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	3.000000	0.0
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	4.0
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	8.0
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	75.0

</td></tr>
</table>

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Type of object</td><td>class(student_score)

```r
#Type of object
class(student_score)
class(student_score_dt)
```

```
[1] "data.frame"
[1] "data.table" "data.frame"
```

</td><td>type(student_score)

```
#Type of object
type(student_score)
```

```
pandas.core.frame.DataFrame
```

</td></tr>
<tr><td>Columns / Series</td><td>ncol(student_score) - #No. of columns
colnames(student_score) - #Column names

```r
#Columns
ncol(student_score) #No. of columns
colnames(student_score) #Names of the columns
```

```
[1] 33
 [1] "school"     "sex"        "age"        "address"    "famsize"    "Pstatus"    "Medu"       "Fedu"
 [9] "Mjob"       "Fjob"       "reason"     "guardian"   "traveltime" "studytime"  "failures"   "schoolsup"
[17] "famsup"     "paid"       "activities" "nursery"    "higher"     "internet"   "romantic"   "famrel"
[25] "freetime"   "goout"      "Dalc"       "Walc"       "health"     "absences"   "G1"         "G2"
[33] "G3"
```

</td><td>Each column in a python dataframe is called “series”.
len(student_score.columns) - #No.of columns
student_score.columns.values - #Column names

```
#Columns
len(student_score.columns)
```

```
33
```

```
student_score.columns.values
```

```
array(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu',
       'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime',
       'studytime', 'failures', 'schoolsup', 'famsup', 'paid',
       'activities', 'nursery', 'higher', 'internet', 'romantic',
       'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health',
       'absences', 'G1', 'G2', 'G3'], dtype=object)
```

</td></tr>
<tr><td>Rows</td><td>The row index starts from 1.
nrow(student_score) - #No. of rows
rownames(student_score) - #Row indices

```r
#Rows
nrow(student_score) #No. of rows
rownames(student_score) #Row indices
```

```
[1] 395
  [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"  "13"  "14"  "15"  "16"  "17"  "18"
 [19] "19"  "20"  "21"  "22"  "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"
 [37] "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"
 [55] "55"  "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"  "72"
 [73] "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"
 [91] "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"  "100" "101" "102" "103" "104" "105" "106" "107" "108"
[109] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120" "121" "122" "123" "124" "125" "126"
[127] "127" "128" "129" "130" "131" "132" "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143" "144"
[145] "145" "146" "147" "148" "149" "150" "151" "152" "153" "154" "155" "156" "157" "158" "159" "160" "161" "162"
[163] "163" "164" "165" "166" "167" "168" "169" "170" "171" "172" "173" "174" "175" "176" "177" "178" "179" "180"
[181] "181" "182" "183" "184" "185" "186" "187" "188" "189" "190" "191" "192" "193" "194" "195" "196" "197" "198"
[199] "199" "200" "201" "202" "203" "204" "205" "206" "207" "208" "209" "210" "211" "212" "213" "214" "215" "216"
[217] "217" "218" "219" "220" "221" "222" "223" "224" "225" "226" "227" "228" "229" "230" "231" "232" "233" "234"
[235] "235" "236" "237" "238" "239" "240" "241" "242" "243" "244" "245" "246" "247" "248" "249" "250" "251" "252"
[253] "253" "254" "255" "256" "257" "258" "259" "260" "261" "262" "263" "264" "265" "266" "267" "268" "269" "270"
[271] "271" "272" "273" "274" "275" "276" "277" "278" "279" "280" "281" "282" "283" "284" "285" "286" "287" "288"
[289] "289" "290" "291" "292" "293" "294" "295" "296" "297" "298" "299" "300" "301" "302" "303" "304" "305" "306"
[307] "307" "308" "309" "310" "311" "312" "313" "314" "315" "316" "317" "318" "319" "320" "321" "322" "323" "324"
[325] "325" "326" "327" "328" "329" "330" "331" "332" "333" "334" "335" "336" "337" "338" "339" "340" "341" "342"
[343] "343" "344" "345" "346" "347" "348" "349" "350" "351" "352" "353" "354" "355" "356" "357" "358" "359" "360"
[361] "361" "362" "363" "364" "365" "366" "367" "368" "369" "370" "371" "372" "373" "374" "375" "376" "377" "378"
[379] "379" "380" "381" "382" "383" "384" "385" "386" "387" "388" "389" "390" "391" "392" "393" "394" "395"
```

</td><td>The rows are called index. The row index starts from 0.
len(student_score.index) = #No. of rows
student_score.index.values - #Row indices

```
#Rows
len(student_score.index)
```

```
395
```

```
student_score.index.values
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
```

</td></tr>
</table>

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Datatype of a column</td><td>class(student_score$G1)

```{r}
#Datatype of a specific column
class(student_score$G1)
```

```
[1] "integer"
```
</td><td>student_score['G1'].dtype

```
#Datatype of a specific column
student_score['G1'].dtype
```

```
dtype('int64')
```
</td></tr>
<tr><td>Select a few columns</td><td>The dplyr library is used.
student_score %>% select("school","age")

```{r}
#Select a few columns
library(dplyr)
student_score %>% select("school","age") %>% head()
```

	school <fctr>	age <int>
1	GP	18
2	GP	17
3	GP	15
4	GP	15
5	GP	16
6	GP	16

6 rows
</td><td>student_score[['school','age']] – #To select more than one columns, then double squares to be used.

```
#Select only a few columns
student_score[['school','age']].head()
```

	school	age
0	GP	18
1	GP	17
2	GP	15
3	GP	15
4	GP	16
</td></tr>
<tr><td>Select a few rows</td><td>The rows from 3 to 10 and all the columns, will be returned. Columns can be accessed only using the index (in this method).
student_score[3:10,]

```{r}
#Select custom rows
student_score[3:10,]
```

	school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>
3	GP	F	15	U	LE3	T	1	1	at_home
4	GP	F	15	U	GT3	T	4	2	health
5	GP	F	16	U	GT3	T	3	3	other
6	GP	M	16	U	LE3	T	4	3	services
7	GP	M	16	U	LE3	T	2	2	other
8	GP	F	17	U	GT3	A	4	4	other
9	GP	M	15	U	LE3	A	3	2	services
10	GP	M	15	U	GT3	T	3	4	other

8 rows | 1-10 of 33 columns
</td><td>Row indices cannot be accessed directly. The 'loc' function is used for this purpose.
The actual rows 3 to 10 and all columns will be returned, since the index starts from 0.
student_score.loc[2:9,]
student_score.loc[2:9,'school':'Pstatus'] - #Rows 3 to 10 and columns 'school' to 'Pstatus' will be displayed
student_score.loc[2:9,['school','Pstatus']] - #Rows 3 to 10 and columns 'school' and 'Pstatus' will be displayed

```
#Select a few rows
student_score.loc[2:9,]
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10
5	GP	M	16	U	LE3	T	4	3	services	other	...	5	4	2	1	2	5	10	15	15	15
6	GP	M	16	U	LE3	T	2	2	other	other	...	4	4	4	1	1	3	0	12	12	11
7	GP	F	17	U	GT3	A	4	4	other	teacher	...	4	1	4	1	1	1	6	6	5	6
8	GP	M	15	U	LE3	A	3	2	services	other	...	4	2	2	1	1	1	0	16	18	19
9	GP	M	15	U	GT3	T	3	4	other	other	...	5	5	1	1	1	5	0	14	15	15

8 rows × 33 columns
</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Add a column</td><td>student_score[‘cust_keys’] = custom_keys - #in a dataframe

```{r}
#Adding a column in a dataframe
custom_keys = 1000:1394
student_score['cust_keys'] = custom_keys
student_score %>% head()
```

freetime <int>	goout <int>	Dalc <int>	Walc <int>	health <int>	absences <int>	G1 <int>	G2 <int>	G3 <int>	cust_keys <int>
3	4	1	1	3	6	5	6	6	1000
3	3	1	1	3	4	5	5	6	1001
3	2	2	3	3	10	7	8	10	1002
2	2	1	1	5	2	15	14	15	1003
3	2	1	2	5	4	6	10	10	1004
4	2	1	2	5	10	15	15	15	1005

6 rows | 26-35 of 34 columns</td><td>student_score[‘cust_keys’] = custom_keys

```
#Add index values as a series in the data frame
custom_keys = list(range(1000,1395))
student_score['cust_keys']=custom_keys
```

```
student_score.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	cust_keys
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	1000
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	1001
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	1002
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1	1	5	2	15	14	15	1003
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	1004

5 rows × 34 columns</td></tr>
<tr><td>Custom row index</td><td>Custom indices can be set only for data tables, not data frames. In a data table, only secondary indices can be set.
setindex(student_score_dt,cust_keys)

```{r}
#Custom Index
setindex(student_score_dt,cust_keys)
indices(student_score_dt)
```

```
[1] "cust_keys"
```</td><td>Add the custom index values as a series in the dataframe, if not already present.<br>The column ‘cust_keys’ will act as secondary index. The original row index remains as-is.<br><b><i>student_score.set_index(‘cust_keys’)</i></b><br>

```
#Custom Index
student_score.set_index('cust_keys', inplace=True)
student_score.loc[1000:1009]
```

cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2
1000	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6
1001	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5
1002	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8
1003	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14
1004	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10
1005	GP	M	16	U	LE3	T	4	3	services	other	...	5	4	2	1	2	5	10	15	15
1006	GP	M	16	U	LE3	T	2	2	other	other	...	4	4	4	1	1	3	0	12	12
1007	GP	F	17	U	GT3	A	4	4	other	teacher	...	4	1	4	1	1	1	6	6	5
1008	GP	M	15	U	LE3	A	3	2	services	other	...	4	2	2	1	1	1	0	16	18
1009	GP	M	15	U	GT3	T	3	4	other	other	...	5	5	1	1	1	5	0	14	15

10 rows × 33 columns</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Access rows using custom index</td><td>student_score_dt[.(1002:1009)]

```r
#Custom Index
setindex(student_score_dt,cust_keys)
indices(student_score_dt)
student_score_dt[.(1002:1009)]
```

freetime <int>	goout <int>	Dalc <int>	Walc <int>	health <int>	absences <int>	G1 <int>	G2 <int>	G3 <int>	cust_keys <int>
3	2	2	3	3	10	7	8	10	1002
2	2	1	1	5	2	15	14	15	1003
3	2	1	2	5	4	6	10	10	1004
4	2	1	2	5	10	15	15	15	1005
4	4	1	1	3	0	12	12	11	1006
1	4	1	1	1	6	6	5	6	1007
2	2	1	1	1	0	16	18	19	1008
5	1	1	1	5	0	14	15	15	1009

8 rows | 25-34 of 34 columns</td><td>student_score.loc[1002:1009]

```
student_score.loc[1002:1009]
```

cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2
1002	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8
1003	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14
1004	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10
1005	GP	M	16	U	LE3	T	4	3	services	other	...	5	4	2	1	2	5	10	15	15
1006	GP	M	16	U	LE3	T	2	2	other	other	...	4	4	4	1	1	3	0	12	12
1007	GP	F	17	U	GT3	A	4	4	other	teacher	...	4	1	4	1	1	1	6	6	5
1008	GP	M	15	U	LE3	A	3	2	services	other	...	4	2	2	1	1	1	0	16	18
1009	GP	M	15	U	GT3	T	3	4	other	other	...	5	5	1	1	1	5	0	14	15

8 rows × 33 columns</td></tr>
<tr><td>Access rows with row indices</td><td>Accessing the rows with the row indices is similar to the previous example above.
student_score[3:10,]

```r
#Select custom rows
student_score[3:10,]
```

	school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>
3	GP	F	15	U	LE3	T	1	1	at_home
4	GP	F	15	U	GT3	T	4	2	health
5	GP	F	16	U	GT3	T	3	3	other
6	GP	M	16	U	LE3	T	4	3	services
7	GP	M	16	U	LE3	T	2	2	other
8	GP	F	17	U	GT3	A	4	4	other
9	GP	M	15	U	LE3	A	3	2	services
10	GP	M	15	U	GT3	T	3	4	other

8 rows | 1-10 of 33 columns</td><td>When custom indices are introduced, accessing the row indices is done with the ‘iloc’ function.
student_score.iloc[2:10]

```
student_score.iloc[2:10]
```

cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2
1002	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8
1003	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14
1004	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10
1005	GP	M	16	U	LE3	T	4	3	services	other	...	5	4	2	1	2	5	10	15	15
1006	GP	M	16	U	LE3	T	2	2	other	other	...	4	4	4	1	1	3	0	12	12
1007	GP	F	17	U	GT3	A	4	4	other	teacher	...	4	1	4	1	1	1	6	6	5
1008	GP	M	15	U	LE3	A	3	2	services	other	...	4	2	2	1	1	1	0	16	18
1009	GP	M	15	U	GT3	T	3	4	other	other	...	5	5	1	1	1	5	0	14	15

8 rows × 33 columns</td></tr>
<tr><td>Reset custom row index</td><td>The secondary indices will be deleted.
setindex(student_score_dt,NULL)

```r
#Remove secondary index
setindex(student_score_dt,NULL)
indices(student_score_dt)
```

NULL</td><td>The secondary indices will be deleted.
student_score.reset_index()

```
student_score.reset_index().head()
```

	cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	1000	GP	F	18	U	GT3	A	4	4	at_home	...	4	3	4	1	1	3	6	5	6	6
1	1001	GP	F	17	U	GT3	T	1	1	at_home	...	5	3	3	1	1	3	4	5	5	6
2	1002	GP	F	15	U	LE3	T	1	1	at_home	...	4	3	2	2	3	3	10	7	8	10
3	1003	GP	F	15	U	GT3	T	4	2	health	...	3	2	2	1	1	5	2	15	14	15
4	1004	GP	F	16	U	GT3	T	3	3	other	...	4	3	2	1	2	5	4	6	10	10

5 rows × 34 columns</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Add column with values computed from existing columns</td><td>student_score['Total']=student_score$G1 + student_Score$G2
<pre>```{r}
#Adding a column with values computed from two columns in the existing dataset
student_score['Total']=student_score$G1+student_score$G2
student_score %>% head()
```</pre>
<table>
<tr><th></th><th>goout<br><int></th><th>Dalc<br><int></th><th>Walc<br><int></th><th>health<br><int></th><th>absences<br><int></th><th>G1<br><int></th><th>G2<br><int></th><th>G3<br><int></th><th>cust_keys<br><int></th><th>Total<br><int></th></tr>
<tr><td></td><td>4</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td><td>6</td><td>1000</td><td>11</td></tr>
<tr><td></td><td>3</td><td>1</td><td>1</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td><td>1001</td><td>10</td></tr>
<tr><td></td><td>2</td><td>2</td><td>3</td><td>3</td><td>10</td><td>7</td><td>8</td><td>10</td><td>1002</td><td>15</td></tr>
<tr><td></td><td>2</td><td>1</td><td>1</td><td>5</td><td>2</td><td>15</td><td>14</td><td>15</td><td>1003</td><td>29</td></tr>
<tr><td></td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>6</td><td>10</td><td>10</td><td>1004</td><td>16</td></tr>
<tr><td></td><td>2</td><td>1</td><td>2</td><td>5</td><td>10</td><td>15</td><td>15</td><td>15</td><td>1005</td><td>30</td></tr>
</table>
6 rows | 27-36 of 35 columns</td><td><b><i>student_score['Total'] = student_score['G1'] + student_score['G2']</i></b>
<pre>student_score['Total']=student_score['G1']+student_score['G2']
student_score.head()</pre>
<table>
<tr><th></th><th>school</th><th>sex</th><th>age</th><th>address</th><th>famsize</th><th>Pstatus</th><th>Medu</th><th>Fedu</th><th>Mjob</th><th>Fjob</th><th>...</th><th>freetime</th><th>goout</th><th>Dalc</th><th>Walc</th><th>health</th><th>absences</th><th>G1</th><th>G2</th><th>G3</th><th>Total</th><th>ust_keys</th></tr>
<tr><td>1000</td><td>GP</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>at_home</td><td>teacher</td><td>...</td><td>3</td><td>4</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td><td>6</td><td>11</td></tr>
<tr><td>1001</td><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>3</td><td>3</td><td>1</td><td>1</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td><td>10</td></tr>
<tr><td>1002</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>LE3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>3</td><td>2</td><td>2</td><td>3</td><td>3</td><td>10</td><td>7</td><td>8</td><td>10</td><td>15</td></tr>
<tr><td>1003</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>4</td><td>2</td><td>health</td><td>services</td><td>...</td><td>2</td><td>2</td><td>1</td><td>1</td><td>5</td><td>2</td><td>15</td><td>14</td><td>15</td><td>29</td></tr>
<tr><td>1004</td><td>GP</td><td>F</td><td>16</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>3</td><td>other</td><td>other</td><td>...</td><td>3</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>6</td><td>10</td><td>10</td><td>16</td></tr>
</table>
rows × 34 columns</td></tr>
<tr><td>Drop Column(s)</td><td>All columns except the ones mentioned below will be selected.<br><b><i>select(student_score, -c(Total,cust_keys)</i></b>
<pre>```{r}
#Drop column(s)
select(student_score,-c(Total,cust_keys)) %>% head()
```</pre>
<table>
<tr><th></th><th>famrel
<int></th><th>freetime
<int></th><th>goout
<int></th><th>Dalc
<int></th><th>Walc
<int></th><th>health
<int></th><th>absences
<int></th><th>G1
<int></th><th>G2
<int></th><th>G3
<int></th></tr>
<tr><td></td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td><td>6</td></tr>
<tr><td></td><td>5</td><td>3</td><td>3</td><td>1</td><td>1</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td></tr>
<tr><td></td><td>4</td><td>3</td><td>2</td><td>2</td><td>3</td><td>3</td><td>10</td><td>7</td><td>8</td><td>10</td></tr>
<tr><td></td><td>3</td><td>2</td><td>2</td><td>1</td><td>1</td><td>5</td><td>2</td><td>15</td><td>14</td><td>15</td></tr>
<tr><td></td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>6</td><td>10</td><td>10</td></tr>
<tr><td></td><td>5</td><td>4</td><td>2</td><td>1</td><td>2</td><td>5</td><td>10</td><td>15</td><td>15</td><td>15</td></tr>
</table>
6 rows | 25-34 of 33 columns</td><td>The series specified in the function will be dropped. Axis = 1 represents the columns.
By specifying axis = 0 and row indices (primary or secondary), the specific rows will be dropped.
student_score.drop(['Total','cust_keys'],axis=1)
<pre>student_score.drop(['Total','cust_keys'],axis=1).head()</pre>
<table>
<tr><th></th><th>school</th><th>sex</th><th>age</th><th>address</th><th>famsize</th><th>Pstatus</th><th>Medu</th><th>Fedu</th><th>Mjob</th><th>Fjob</th><th>...</th><th>famrel</th><th>freetime</th><th>goout</th><th>Dalc</th><th>Walc</th><th>health</th><th>absences</th><th>G1</th><th>G2</th><th>G3</th></tr>
<tr><td>0</td><td>GP</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>at_home</td><td>teacher</td><td>...</td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td><td>6</td></tr>
<tr><td>1</td><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>5</td><td>3</td><td>3</td><td>1</td><td>1</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td></tr>
<tr><td>2</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>LE3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>2</td><td>3</td><td>3</td><td>10</td><td>7</td><td>8</td><td>10</td></tr>
<tr><td>3</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>4</td><td>2</td><td>health</td><td>services</td><td>...</td><td>3</td><td>2</td><td>2</td><td>1</td><td>1</td><td>5</td><td>2</td><td>15</td><td>14</td><td>15</td></tr>
<tr><td>4</td><td>GP</td><td>F</td><td>16</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>3</td><td>other</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>6</td><td>10</td><td>10</td></tr>
</table>
5 rows × 33 columns</td></tr>
<tr><td>Statistic of one column</td><td>Other statistics like min, max, sum, etc. can be calculated the same way, as below.
mean(student_score$G1)
<pre>```{r}
#Statistic of one column
mean(student_score$G1)
```</pre>
<pre>[1] 10.90886</pre></td><td>Other statistics like min, max, sum, etc. can be calculated the same way, as below.<br><b><i>student_score['G1'].mean()</i></b>
<pre>student_score['G1'].mean()</pre>
<pre>10.90886075949367</pre></td></tr>
<tr><td>Finding null values</td><td><b><i>is.na(student_score$address)</i></b><br>Without the sum (as in the image below), the is.na() returns binary values – True/False.<br>Sum being zero indicates that there are no null values in the given column.
<pre>```{r}
#Sum of null values in a column
sum(is.na(student_score$address))
```</pre>
<pre>[1] 0</pre></td><td>student_score['address'].isna()
Without the sum (as in the image below), the isna() returns binary values – True/False.
Sum being zero indicates that there are no null values in the given column.
<pre>student_score['address'].isna().sum()</pre>
<pre>0</pre></td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Drop / Omit null values</td><td>This function drops out the rows that has missing/na/null values.
na.omit(student_score)

```r
#Omit null values
na.omit(student_score)
```

</td><td>This function drops out the rows that has missing/na/null values.
student_score.dropna()

```python
#Drop null values
student_score.dropna()
```

</td></tr>
<tr><td>No. of unique values in a column</td><td>nlevels(student_score$Mjob) - #No. of unique values
levels(student_score$Mjob) - #Returns unique values

```r
#Unique values in a column
nlevels(student_score$Mjob) # No. of unique values
levels(student_score$Mjob)  # Returns unqiue values
```

```
[1] 5
[1] "at_home"  "health"   "other"    "services" "teacher"
```

</td><td>student_score['Mjob'].nunique() - #No. of unique values
student_score['Mjob'].unique() - #Returns unique values

```python
#Unique values in a series
student_score['Mjob'].nunique() #No. of unique values
student_score['Mjob'].unique() #Returns unique values
```

```
5

array(['at_home', 'health', 'other', 'services', 'teacher'], dtype=object)
```

</td></tr>
<tr><td>Count of unique values in a column</td><td>student_score %>% group_by %>% summarise(MJob_count = n()) %>% arrange(-MJob_count)
The column has to be grouped for similar values and the count is calculated using n(). Without the arrange function, the output will be arranged in ascending order of Mjob (the column in discussion).

```r
#Count of unique values
student_score %>% group_by(Mjob) %>% summarise(MJob_count = n()) %>% arrange(-MJob_count)
```

Mjob <fctr>	MJob_count <int>
other	141
services	103
at_home	59
teacher	58
health	34

5 rows

</td><td>The output is automatically arranged in descending order of the values.
student_score['Mjob'].value_counts()

```python
#Count of unique values in a series
student_score['Mjob'].value_counts()
```

```
other       141
services    103
at_home      59
teacher      58
health       34
Name: Mjob, dtype: int64
```

</td></tr>
<tr><td>Sort dataframe based on values of a column</td><td>The minus(-) symbol in arrange function will arrange the values of the column in the descending order. The same can be extended to categorical columns too.
student_score %>% arrange(-Total)

```r
#Sort dataframe based on values of one column
student_score %>% arrange(-Total) %>% head()
```

goout <int>	Dalc <int>	Walc <int>	health <int>	absences <int>	G1 <int>	G2 <int>	G3 <int>	cust_keys <int>	Total <int>
2	1	1	2	4	19	19	20	1047	38
3	1	1	5	2	19	18	18	1042	37
3	1	1	4	6	18	19	19	1110	37
2	1	1	3	10	18	19	19	1113	37
4	1	1	1	0	19	18	19	1374	37
5	2	5	4	8	18	18	18	1129	36

6 rows | 27-36 of 35 columns

</td><td>The ‘by’ attritube is to specify the column. ‘ascending’ = False arranges in descending order. Without this attribute, the values will be arranged in ascending order by default. The same can be extended to categorical columns too.
student_score.sort_values(by='Total',ascending=False)

```python
#Sort dataframe based on values of one column
student_score.sort_values(by='Total', ascending=False).head()
```

	cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	Total
47	1047	GP	M	16	U	GT3	T	4	3	health	...	2	2	1	1	2	4	19	19	20	38
113	1113	GP	M	15	U	LE3	T	4	2	teacher	...	5	2	1	1	3	10	18	19	19	37
110	1110	GP	M	15	U	LE3	A	4	4	teacher	...	5	3	1	1	4	6	18	19	19	37
374	1374	MS	F	18	R	LE3	T	4	4	other	...	4	4	1	1	1	0	19	18	19	37
42	1042	GP	M	15	U	GT3	T	4	4	services	...	3	3	1	1	5	2	19	18	18	37

5 rows × 35 columns

</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

Function	R	Python
Rank values of a column	The rank function automatically ranks the mentioned column in ascending order. The same can be extended to categorical columns too. The method attiribute spcifies how to handle the same values. ***student_score['rank'] = rank(student_score$absences, ties.method='min')***	The rank function ranks the mentioned column in ascending order, by default. By default, the same is applicable to categorical columns too. The method attiribute spcifies how to handle the same values. The rank function also include attributes to rank percentages and handle null values. ***student_score['rank']=student_score['absences'].rank(method='min')***
View rank	The specific column or the rank column has to be sorted, to view the rank in the correct order. Since method is specified as min, all the same values will carry the same rank. ***student_score[order(-student_score$absences),]***	The specific column or the rank column has to be sorted, to view the rank in the correct order. Since method is specified as min, all the same values will carry the same rank. ***student_score.sort_values(by'absences', ascending=True)***

```r
#Adding a rank column based on values of another column
student_score['rank']=rank(student_score$absences, ties.method='min')
student_score %>% head()
```

Dalc	Walc	health	absences	G1	G2	G3	cust_keys	Total	rank
1	1	3	6	5	6	6	1000	11	250
1	1	3	4	5	5	6	1001	10	192
2	3	3	10	7	8	10	1002	15	313
1	1	5	2	15	14	15	1003	29	119
1	2	5	4	6	10	10	1004	16	192
1	2	5	10	15	15	15	1005	30	313

6 rows | 28-37 of 36 columns

```python
#Adding a rank column based on values of another column
student_score['rank']=student_score['absences'].rank(method='min')
student_score.head()
```

	cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	...	goout	Dalc	Walc	health	absences	G1	G2	G3	Total	rank
0	1000	GP	F	18	U	GT3	A	4	4	at_home	...	4	1	1	3	6	5	6	6	11	250.0
1	1001	GP	F	17	U	GT3	T	1	1	at_home	...	3	1	1	3	4	5	5	6	10	192.0
2	1002	GP	F	15	U	LE3	T	1	1	at_home	...	2	2	3	3	10	7	8	10	15	313.0
3	1003	GP	F	15	U	GT3	T	4	2	health	...	2	1	1	5	2	15	14	15	29	119.0
4	1004	GP	F	16	U	GT3	T	3	3	other	...	2	1	2	5	4	6	10	10	16	192.0

5 rows × 36 columns

```r
#To view the rank values in the correct order
student_score[order(-student_score$absences),]
```

Dalc	Walc	health	absences	G1	G2	G3	cust_keys	Total	rank
1	2	5	23	13	13	13	1320	26	385
2	4	1	22	6	6	4	1216	12	382
1	4	3	22	9	9	9	1277	18	382
1	2	1	22	13	10	11	1313	23	382
1	3	2	21	17	18	18	1260	35	381
1	4	5	20	9	7	8	1118	16	377
1	1	5	20	13	12	12	1237	25	377
1	1	3	20	15	14	13	1304	29	377
1	1	2	20	14	12	13	1311	26	377
3	4	3	19	11	9	10	1281	20	376

11-20 of 395 rows | 28-37 of 36 columns Previous 1 2 3 4 5 6 ... 40 Next

```python
#To view the rank values in the correct order
student_score.sort_values(by='absences', ascending=True).tail(20)
```

	cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	...	goout	Dalc	Walc	health	absences	G1	G2	G3	Total	rank
281	1281	GP	M	17	U	LE3	A	3	2	teacher	...	4	3	4	3	19	11	9	10	20	376.0
237	1237	GP	F	16	U	GT3	T	2	1	other	...	2	1	1	5	20	13	12	12	25	377.0
118	1118	GP	M	17	R	GT3	T	1	3	other	...	4	1	4	5	20	9	7	8	16	377.0
311	1311	GP	F	19	U	GT3	T	2	1	at_home	...	1	1	1	2	20	14	12	13	26	377.0
304	1304	GP	M	19	U	GT3	T	3	3	other	...	4	1	1	3	20	15	14	13	29	377.0
260	1260	GP	F	18	U	GT3	T	4	3	services	...	2	1	3	2	21	17	18	18	35	381.0
313	1313	GP	F	19	U	LE3	T	3	2	services	...	2	1	2	1	22	13	10	11	23	382.0
216	1216	GP	F	17	U	GT3	T	4	3	other	...	5	2	4	1	22	6	6	4	12	382.0
277	1277	GP	M	18	U	GT3	T	4	4	teacher	...	4	1	4	3	22	9	9	9	18	382.0

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Convert numerical column into categorical buckets</td><td>The cut function automatically divides the numerical column into 6 (breaks = 6 or as mentioned inside the function) buckets, depending on the values of the numerical column. The count of values in each bucket is found using the group_by and summarise functions.
The number of breaks can be an integer, asking R to divide the values automatically (as below). Or, we can specify the intervals. For eg., the intervals can be taken as “0-10”,”10-15”,”15-20”..and so on (right exclusive). And hence, the labels should be one less than the number of bins given.
student_score[‘Total_Bucket’] = cut(student_score$Total, 6 ,labels=c(“A”,”B”,”C”,”D”,”E”,”F”))
```{r}
#Numerical column into automatic categorical buckets
student_score['Total_Bucket']=cut(student_score$Total,6,labels=c("A","B","C","D","E","F"))
student_score %>% group_by(Total_Bucket) %>% summarise(Total_Bucket_count=n())
```
R Console
tbl_df
6 x 2
<table><tr><th>Total_Bucket
<fctr></th><th>Total_Bucket_count
<int></th></tr><tr><td>A</td><td>11</td></tr><tr><td>B</td><td>62</td></tr><tr><td>C</td><td>124</td></tr><tr><td>D</td><td>101</td></tr><tr><td>E</td><td>74</td></tr><tr><td>F</td><td>23</td></tr></table>6 rows</td><td>The cut function is very similar in R and Python. Only change is, instead of breaks, in Python, it’s called bins. The number of bins can be an integer, asking python to divide the values automatically. Or, we can specify the intervals (as below). Here, the intervals are taken as “0-10”,”10-15”,”15-20”..and so on (right exclusive). And hence, the labels should be one less than the number of bins given.
student_score[‘Total_Bucket_custom’] = pd.cut(student_score[‘Total’],bins=[0,10,15,20,25,30,40],labels=[“A”,”B”,”C”,”D”,”E”,”F”])
#Numerical column into custom categorical buckets
student_score['Total_Bucket_custom']=pd.cut(student_score['Total'],bins=[0,10,15,20,25,30,40],labels=["A","B","C","D","E","F"])
student_score['Total_Bucket_custom'].value_counts()
C 111
D 91
E 77
B 59
F 43
A 14
Name: Total_Bucket_custom, dtype: int64</td></tr>
<tr><td>Select specific rows and columns</td><td>Filter functions is used to filter out the rows with the specific values or logical functions. Select function is used to filter out the required columns.
student_score %>% filter(sex == “F” & age>=18) %>% select(school, sex, age, Mjob, Fjob, G1, G2)
```{r}
#Select columns and rows
student_score %>% filter(sex=="F" & age>=18) %>% select(school,sex,age,Mjob,Fjob,G1,G2) %>% head()
```
<table><tr><th></th><th>school
<fctr></th><th>sex
<fctr></th><th>age
<int></th><th>Mjob
<fctr></th><th>Fjob
<fctr></th><th>G1
<int></th><th>G2
<int></th></tr><tr><td>1</td><td>GP</td><td>F</td><td>18</td><td>at_home</td><td>teacher</td><td>5</td><td>6</td></tr><tr><td>2</td><td>GP</td><td>F</td><td>19</td><td>at_home</td><td>other</td><td>7</td><td>8</td></tr><tr><td>3</td><td>GP</td><td>F</td><td>18</td><td>at_home</td><td>other</td><td>9</td><td>8</td></tr><tr><td>4</td><td>GP</td><td>F</td><td>19</td><td>other</td><td>other</td><td>8</td><td>8</td></tr><tr><td>5</td><td>GP</td><td>F</td><td>18</td><td>other</td><td>other</td><td>9</td><td>8</td></tr><tr><td>6</td><td>GP</td><td>F</td><td>18</td><td>other</td><td>other</td><td>7</td><td>0</td></tr></table>6 rows</td><td>To access the rows, the loc function is used along with the required conditions or logicals. The second set of double square brackets lets us filter out the required columns.
student_score.loc[(student_score[‘age’]>=18) & (student_score[‘sex’]==”F”)] [[‘school’, ‘ sex’, ‘age’, ‘Mjob’, ‘Fjob’, ‘ G1’, ‘G2]]
#Select columns and rows
student_score.loc[(student_score['age']>=18) & (student_score['sex']=="F")][['school','sex','age','Mjob','Fjob','G1','G2']].head
<table><tr><th></th><th>school</th><th>sex</th><th>age</th><th>Mjob</th><th>Fjob</th><th>G1</th><th>G2</th></tr><tr><td>0</td><td>GP</td><td>F</td><td>18</td><td>at_home</td><td>teacher</td><td>5</td><td>6</td></tr><tr><td>127</td><td>GP</td><td>F</td><td>19</td><td>at_home</td><td>other</td><td>7</td><td>8</td></tr><tr><td>157</td><td>GP</td><td>F</td><td>18</td><td>at_home</td><td>other</td><td>9</td><td>8</td></tr><tr><td>210</td><td>GP</td><td>F</td><td>19</td><td>other</td><td>other</td><td>8</td><td>8</td></tr><tr><td>225</td><td>GP</td><td>F</td><td>18</td><td>other</td><td>other</td><td>9</td><td>8</td></tr></table></td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Select rows based on more than one value of a column</td><td>

student_score %>% filter(Mjob %in% c("at_home", " teacher"))

```{r}
#Select rows based on more than one value of a column
student_score %>% filter(Mjob %in% c("at_home","teacher")) %>% head()
```

	school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>
1	GP	F	18	U	GT3	A	4	4	at_home
2	GP	F	17	U	GT3	T	1	1	at_home
3	GP	F	15	U	LE3	T	1	1	at_home
4	GP	F	15	U	GT3	T	4	4	teacher
5	GP	M	15	U	GT3	T	4	3	teacher
6	GP	M	15	U	GT3	T	4	3	teacher

6 rows | 1-10 of 38 columns

</td><td>

student_score.loc[student_score['Mjob'].isin(["at_home", "teacher"])]

```
#Select rows based on more than one value of a column
student_score.loc[student_score['Mjob'].isin(["at_home","teacher"])].head()
```

	cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	...	Walc	health	absences	G1	G2	G3	Total	rank	Total_Bucket	Tot
0	1000	GP	F	18	U	GT3	A	4	4	at_home	...	1	3	6	5	6	6	11	250.0	B	
1	1001	GP	F	17	U	GT3	T	1	1	at_home	...	1	3	4	5	5	6	10	192.0	A	
2	1002	GP	F	15	U	LE3	T	1	1	at_home	...	3	3	10	7	8	10	15	313.0	B	
10	1010	GP	F	15	U	GT3	T	4	4	teacher	...	2	2	0	10	8	9	18	1.0	C	
13	1013	GP	M	15	U	GT3	T	4	3	teacher	...	2	3	2	10	10	11	20	119.0	C	

5 rows × 38 columns

</td></tr>
<tr><td>Select rows based on a partial string match</td><td>

The %like% will parse the "reason" column and filter out all the values that has a substring "our". In the following example, the only value is "course".

student_score %>% filter (reason %like% "our")

```{r}
#Select rows based on a partial string match
student_score %>% filter(reason %like% "our") %>% head()
```

Fjob <fctr>	reason <fctr>	guardian <fctr>	traveltime <int>	studytime <int>	failures <int>	schoolsup <fctr>	famsup <fctr>	paid <fctr>	activities <fctr>
teacher	course	mother	2	2	0	yes	no	no	no
other	course	father	1	2	0	no	yes	no	no
services	course	father	1	1	0	no	yes	yes	yes
other	course	mother	2	2	0	no	yes	yes	no
services	course	mother	1	1	3	no	yes	no	yes
other	course	mother	1	2	0	no	no	no	yes

6 rows | 11-20 of 38 columns

</td><td>

The function "contains" compares the string value with the substring passed in the pat (pattern) parameter.

student_score.loc[student_score['reason'].str.contains(pat="our")]

```
#Select rows based on a partial string match
student_score.loc[student_score['reason'].str.contains(pat="our")].head()
```

	school	sex	age	address	famsize	Mjob	Fjob	reason
0	GP	F	18	U	GT3	at_home	teacher	course
1	GP	F	17	U	GT3	at_home	other	course
12	GP	M	15	U	LE3	health	services	course
13	GP	M	15	U	GT3	teacher	other	course
18	GP	M	17	U	GT3	services	services	course

</td></tr>
<tr><td>Filter rows based on values equal/not equal to custom values</td><td>

student_score %>% filter(Mjob != "health")

```{r}
student_score %>% filter(Mjob!="health") %>% head()
```

	school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>
1	GP	F	18	U	GT3	A	4	4	at_home
2	GP	F	17	U	GT3	T	1	1	at_home
3	GP	F	15	U	LE3	T	1	1	at_home
4	GP	F	16	U	GT3	T	3	3	other
5	GP	M	16	U	LE3	T	4	3	services
6	GP	M	16	U	LE3	T	2	2	other

6 rows | 1-10 of 38 columns

</td><td>

student_score[~student_score.Mjob.isin(['health'])]

```
student_score[~student_score.Mjob.isin(['health'])].head()
```

cust_keys	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	Tot
1000	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	1
1001	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	1
1002	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	1
1004	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	1
1005	GP	M	16	U	LE3	T	4	3	services	other	...	4	2	1	2	5	10	15	15	15	3

5 rows × 34 columns

</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Statistic of a column based on groups formed in another column</td><td>While calculating the statistic of a numerical column, the missing values are to be handled appropriately. ‘na.rm=True’ has to be exclusively mentioned to omit the missing values while calculating mean (if not already dropped from the dataframe).
student_score %>% group_by(Mjob) %>% summarise(G1_mean=mean(G1,na.rm=T))
<pre>```{r}
#Mean of values in a column, based on grouped values on another column
student_score %>% group_by(Mjob) %>% summarise(G1_mean=mean(G1,na.rm=T))
```</pre>
R Console tbl_df 5 x 2<br>
<table>
<tr><th>Mjob<br><fctr></th><th>G1_mean<br><dbl></th></tr>
<tr><td>at_home</td><td>10.45763</td></tr>
<tr><td>health</td><td>12.20588</td></tr>
<tr><td>other</td><td>10.17730</td></tr>
<tr><td>services</td><td>11.38835</td></tr>
<tr><td>teacher</td><td>11.53448</td></tr>
</table>
5 rows</td><td>The missing values are automatically omitted while calculating the statistic of a numerical column, even if the null values are not dropped previously.<br><b><i>student_score.groupby(‘Mjob’)[‘G1’].mean()</i></b><br>
<pre>#Mean of values in a column, based on grouped values on another column
student_score.groupby('Mjob')['G1'].mean()</pre>
<pre>Mjob
at_home     10.457627
health      12.205882
other       10.177305
services    11.388350
teacher     11.534483
Name: G1, dtype: float64</pre></td></tr>
<tr><td>Count of categorical variables based on groups formed in another column</td><td><b><i>student_score %>% group_by(Mjob, famsize) %>% summarise(MJob_famsize_count=n())</i></b><br>
<pre>```{r}
#Count of values in a column, based on grouped values on another column
student_score %>% group_by(Mjob, famsize) %>% summarise(MJob_famsize_count=n())
```</pre>
R Console grouped_df 10 x 3
<table>
<tr><th>Mjob
<fctr></th><th>famsize
<fctr></th><th>MJob_famsize_count
<int></th></tr>
<tr><td>at_home</td><td>GT3</td><td>44</td></tr>
<tr><td>at_home</td><td>LE3</td><td>15</td></tr>
<tr><td>health</td><td>GT3</td><td>24</td></tr>
<tr><td>health</td><td>LE3</td><td>10</td></tr>
<tr><td>other</td><td>GT3</td><td>106</td></tr>
<tr><td>other</td><td>LE3</td><td>35</td></tr>
<tr><td>services</td><td>GT3</td><td>70</td></tr>
<tr><td>services</td><td>LE3</td><td>33</td></tr>
<tr><td>teacher</td><td>GT3</td><td>37</td></tr>
<tr><td>teacher</td><td>LE3</td><td>21</td></tr>
</table>
1-10 of 10 rows</td><td>student_score.groupby(‘Mjob’)[‘famsize’].value_counts()
<pre>#Count of values in a column, based on grouped values on another column
student_score.groupby('Mjob')['famsize'].value_counts()</pre>
<pre>Mjob   famsize
at_home   GT3   44
          LE3   15
health   GT3   24
          LE3   10
other   GT3   106
          LE3   35
services GT3   70
          LE3   33
teacher   GT3   37
          LE3   21
Name: famsize, dtype: int64</pre></td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Merge two dataframes – row wise</td><td>The rbind function merges the dataframes row wise. Hence, both dataframes should have equal number of columns.
rbind(student_score, student_score_por)
<pre>```{r}
student_score_por = read.csv("student-por.csv", header = T, sep = ";")
dim(student_score_por)
```</pre>
<pre>[1] 649  33</pre>
<pre>```{r}
#Merge two dataframes - row wise
rbind(student_score,student_score_por)
```</pre>
<table>
<tr><th>school
<fctr></th><th>sex
<fctr></th><th>age
<int></th><th>address
<fctr></th><th>famsize
<fctr></th><th>Pstatus
<fctr></th><th>Medu
<int></th><th>Fedu
<int></th><th>Mjob
<fctr></th><th>Fjob
<fctr></th></tr>
<tr><td>GP</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>at_home</td><td>teacher</td></tr>
<tr><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td></tr>
<tr><td>GP</td><td>F</td><td>15</td><td>U</td><td>LE3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td></tr>
<tr><td>GP</td><td>F</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>4</td><td>2</td><td>health</td><td>services</td></tr>
<tr><td>GP</td><td>F</td><td>16</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>3</td><td>other</td><td>other</td></tr>
<tr><td>GP</td><td>M</td><td>16</td><td>U</td><td>LE3</td><td>T</td><td>4</td><td>3</td><td>services</td><td>other</td></tr>
<tr><td>GP</td><td>M</td><td>16</td><td>U</td><td>LE3</td><td>T</td><td>2</td><td>2</td><td>other</td><td>other</td></tr>
<tr><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>other</td><td>teacher</td></tr>
<tr><td>GP</td><td>M</td><td>15</td><td>U</td><td>LE3</td><td>A</td><td>3</td><td>2</td><td>services</td><td>other</td></tr>
<tr><td>GP</td><td>M</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>4</td><td>other</td><td>other</td></tr>
</table>
1-10 of 1,044 rows | 1-10 of 33 columns Previous 1 2 3 4 5 6 ... 100 Next</td><td>The ‘axis = 0’ in the concat function below refers to row wise merge. It is not mandatory to have equal number of columns. The extra columns in the dataframe will be filled with “NA”.
pd.concat([student_score, stduent_score_por],axis=0)
<pre>student_score_por = pd.read_csv("student-por.csv", sep=";", header=0)
student_score_por.shape</pre>
<pre>(649, 33)</pre>
<pre>#Merge two dataframes - row wise
pd.concat([student_score,student_score_por],axis=0)</pre>
<table>
<tr><th></th><th>school</th><th>sex</th><th>age</th><th>address</th><th>famsize</th><th>Pstatus</th><th>Medu</th><th>Fedu</th><th>Mjob</th><th>Fjob</th><th>...</th><th>famrel</th><th>freetime</th><th>goout</th><th>Dalc</th><th>Walc</th><th>health</th><th>absences</th><th>G1</th><th>G2</th><th>G3</th></tr>
<tr><td>0</td><td>GP</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>at_home</td><td>teacher</td><td>...</td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td><td>6</td></tr>
<tr><td>1</td><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>5</td><td>3</td><td>3</td><td>1</td><td>1</td><td>3</td><td>4</td><td>5</td><td>5</td><td>6</td></tr>
<tr><td>2</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>LE3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>2</td><td>3</td><td>3</td><td>10</td><td>7</td><td>8</td><td>10</td></tr>
<tr><td>3</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>4</td><td>2</td><td>health</td><td>services</td><td>...</td><td>3</td><td>2</td><td>2</td><td>1</td><td>1</td><td>5</td><td>2</td><td>15</td><td>14</td><td>15</td></tr>
<tr><td>4</td><td>GP</td><td>F</td><td>16</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>3</td><td>other</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>6</td><td>10</td><td>10</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>644</td><td>MS</td><td>F</td><td>19</td><td>R</td><td>GT3</td><td>T</td><td>2</td><td>3</td><td>services</td><td>other</td><td>...</td><td>5</td><td>4</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>10</td><td>11</td><td>10</td></tr>
<tr><td>645</td><td>MS</td><td>F</td><td>18</td><td>U</td><td>LE3</td><td>T</td><td>3</td><td>1</td><td>teacher</td><td>services</td><td>...</td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>1</td><td>4</td><td>15</td><td>15</td><td>16</td></tr>
<tr><td>646</td><td>MS</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>other</td><td>other</td><td>...</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>5</td><td>6</td><td>11</td><td>12</td><td>9</td></tr>
<tr><td>647</td><td>MS</td><td>M</td><td>17</td><td>U</td><td>LE3</td><td>T</td><td>3</td><td>1</td><td>services</td><td>services</td><td>...</td><td>2</td><td>4</td><td>5</td><td>3</td><td>4</td><td>2</td><td>6</td><td>10</td><td>10</td><td>10</td></tr>
<tr><td>648</td><td>MS</td><td>M</td><td>18</td><td>R</td><td>LE3</td><td>T</td><td>3</td><td>2</td><td>services</td><td>other</td><td>...</td><td>4</td><td>4</td><td>1</td><td>3</td><td>4</td><td>5</td><td>4</td><td>10</td><td>11</td><td>11</td></tr>
</table>
1044 rows × 33 columns</td></tr>
<tr><td>Merge two dataframes – column wise</td><td>The cbind function can be used to merge the dataframes column wise. Hence, the number of rows in both the dataframes should be equal. Since the rows are unequal, cbind cannot be done in this example.</td><td>The ‘axis = 1’ in the concat function below refers to column wise merge. It is not mandatory to have equal number of rows. The extra rows in the dataframe will be filled with “NA”.
pd.concat([student_score, student_score_por], axis = 1)
<pre>#Merge two dataframes - column wise
pd.concat([student_score,student_score_por],axis=1)</pre>
<table>
<tr><th></th><th>school</th><th>sex</th><th>age</th><th>address</th><th>famsize</th><th>Pstatus</th><th>Medu</th><th>Fedu</th><th>Mjob</th><th>Fjob</th><th>...</th><th>famrel</th><th>freetime</th><th>goout</th><th>Dalc</th><th>Walc</th><th>health</th><th>absences</th><th>G1</th><th>G2</th><th>G3</th></tr>
<tr><td>0</td><td>GP</td><td>F</td><td>18.0</td><td>U</td><td>GT3</td><td>A</td><td>4.0</td><td>4.0</td><td>at_home</td><td>teacher</td><td>...</td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>3</td><td>4</td><td>0</td><td>11</td><td>11</td></tr>
<tr><td>1</td><td>GP</td><td>F</td><td>17.0</td><td>U</td><td>GT3</td><td>T</td><td>1.0</td><td>1.0</td><td>at_home</td><td>other</td><td>...</td><td>5</td><td>3</td><td>3</td><td>1</td><td>1</td><td>3</td><td>2</td><td>9</td><td>11</td><td>11</td></tr>
<tr><td>2</td><td>GP</td><td>F</td><td>15.0</td><td>U</td><td>LE3</td><td>T</td><td>1.0</td><td>1.0</td><td>at_home</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>2</td><td>3</td><td>3</td><td>6</td><td>12</td><td>13</td><td>12</td></tr>
<tr><td>3</td><td>GP</td><td>F</td><td>15.0</td><td>U</td><td>GT3</td><td>T</td><td>4.0</td><td>2.0</td><td>health</td><td>services</td><td>...</td><td>3</td><td>2</td><td>2</td><td>1</td><td>1</td><td>5</td><td>0</td><td>14</td><td>14</td><td>14</td></tr>
<tr><td>4</td><td>GP</td><td>F</td><td>16.0</td><td>U</td><td>GT3</td><td>T</td><td>3.0</td><td>3.0</td><td>other</td><td>other</td><td>...</td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>5</td><td>0</td><td>11</td><td>13</td><td>13</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>644</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>...</td><td>5</td><td>4</td><td>2</td><td>1</td><td>2</td><td>5</td><td>4</td><td>10</td><td>11</td><td>10</td></tr>
<tr><td>645</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>...</td><td>4</td><td>3</td><td>4</td><td>1</td><td>1</td><td>1</td><td>4</td><td>15</td><td>15</td><td>16</td></tr>
<tr><td>646</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>...</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>5</td><td>6</td><td>11</td><td>12</td><td>9</td></tr>
<tr><td>647</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>...</td><td>2</td><td>4</td><td>5</td><td>3</td><td>4</td><td>2</td><td>6</td><td>10</td><td>10</td><td>10</td></tr>
<tr><td>648</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>NaN</td><td>...</td><td>4</td><td>4</td><td>1</td><td>3</td><td>4</td><td>5</td><td>4</td><td>10</td><td>11</td><td>11</td></tr>
</table>
649 rows × 66 columns</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Join dataframes based on the index</td><td>

full_join(student_score, student_score_por)

Here “full_join” performs the same operation as rbind. Other similar functions are inner_join, left_join, right_join.

```{r}
#Join operation - to merge two dataframes
full_join(student_score, student_score_por)
```

R Console | data.frame 1044 x 33

school <fctr>	sex <fctr>	age <int>	address <fctr>	famsize <fctr>	Pstatus <fctr>	Medu <int>	Fedu <int>	Mjob <fctr>	Fjob <fctr>
GP	F	18	U	GT3	A	4	4	at_home	teacher
GP	F	17	U	GT3	T	1	1	at_home	other
GP	F	15	U	LE3	T	1	1	at_home	other
GP	F	15	U	GT3	T	4	2	health	services
GP	F	16	U	GT3	T	3	3	other	other
GP	M	16	U	LE3	T	4	3	services	other
GP	M	16	U	LE3	T	2	2	other	other
GP	F	17	U	GT3	A	4	4	other	teacher
GP	M	15	U	LE3	A	3	2	services	other
GP	M	15	U	GT3	T	3	4	other	other

1-10 of 1,044 rows | 1-10 of 33 columns Previous 1 2 3 4 5 6 ... 100 Next

</td><td>

The below function can be performed to join the dataframes, column wise. The number of rows need not be the same. The extra rows will be filled with “NA”. The column names in the dataframes ‘cannot’ be identical.

student_score.join(student_score_por, how=’outer’)

```
#Join operation - to merge two dataframes
student_score.join(student_score_por,how='outer')
```

	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18.0	U	GT3	A	4.0	4.0	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17.0	U	GT3	T	1.0	1.0	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15.0	U	LE3	T	1.0	1.0	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15.0	U	GT3	T	4.0	2.0	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16.0	U	GT3	T	3.0	3.0	other	other	...	4	3	2	1	2	5	0	11	13	13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	5	4	2	1	2	5	4	10	11	10
645	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	4	3	4	1	1	1	4	15	15	16
646	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1	1	1	1	1	5	6	11	12	9
647	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2	4	5	3	4	2	6	10	10	10
648	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	4	4	1	3	4	5	4	10	11	11

649 rows × 66 columns

</td></tr>
<tr><td>Merge two dataframes based on the values of the same column</td><td>

The “by” attribute in the merge function indicates, by which column the merging has to be performed. “all” parameter refers to the type of join. Here, it is outer join.

merge(student_score, student_score_por, by=”school”, all = TRUE)

```{r}
#Merge dataframes - Merge based on one column
merge(student_score, student_score_por, by="school", all = TRUE)
```

Walc.x <int>	health.x <int>	absences.x <int>	G1.x <int>	G2.x <int>	G3.x <int>	sex.y <fctr>	age.y <int>	address.y <fctr>	famsize.y <fctr>
1	3	6	5	6	6	F	18	U	GT3
1	3	6	5	6	6	F	17	U	GT3
1	3	6	5	6	6	F	15	U	LE3
1	3	6	5	6	6	F	15	U	GT3
1	3	6	5	6	6	F	16	U	GT3
1	3	6	5	6	6	M	16	U	LE3
1	3	6	5	6	6	M	16	U	LE3
1	3	6	5	6	6	F	17	U	GT3
1	3	6	5	6	6	M	15	U	LE3
1	3	6	5	6	6	M	15	U	GT3

1-10 of 158,023 rows | 28-37 of 65 columns Previous 1 2 3 4 5 6 ... 100 Next

</td><td>

The “on” attribute in the merge function by which column the merging has to be performed. “how” parameter refers to the type of join.

pd.merge(student_score, student_score_por, how=”outer”, on=”school”)

```
#Merge dataframes - Merge based on one column
pd.merge(student_score, student_score_por, how="outer", on="school")
```

	school	sex_x	age_x	address_x	famsize_x	Pstatus_x	Medu_x	Fedu_x	Mjob_x	Fjob_x	...	famrel_y	freetime_y	goout_y	Dalc_y	Walc_y	heal
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	
1	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	5	3	3	1	1	
2	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	2	2	3	
3	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	2	2	1	1	
4	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	2	1	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
158018	MS	M	19	U	LE3	T	1	1	other	at_home	...	5	4	2	1	2	
158019	MS	M	19	U	LE3	T	1	1	other	at_home	...	4	3	4	1	1	
158020	MS	M	19	U	LE3	T	1	1	other	at_home	...	1	1	1	1	1	
158021	MS	M	19	U	LE3	T	1	1	other	at_home	...	2	4	5	3	4	
158022	MS	M	19	U	LE3	T	1	1	other	at_home	...	4	4	1	3	4	

158023 rows × 65 columns

</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Apply the same function to all the values in a column</td><td>The lapply function takes in one column, applies the function that is passed and returns a list with the same number of values as in the original column. The function specified can be built-in or user-defined.
lapply(student_score$G1, percent)

```{r}
#Pass a column to a function
percent <- function(g){
  return ((g/20)*100)
}

G1_percent = lapply(student_score$G1,percent)
G1_percent %>% tail(5)
```

<pre>
[[1]]
[1] 45

[[2]]
[1] 70

[[3]]
[1] 50

[[4]]
[1] 55

[[5]]
[1] 40
</pre></td><td>The apply function takes in one series and applies the function that is passed and returns a series with the same number of values as the original series. The function specified can be built-in or user-defined.
student_score['G1'].apply(percent)

```
#Defining a function
def percent(g):
    return ((g/20)*100)
```

```
#Pass a series to a function
student_score['G1'].apply(percent)
```

<pre>
0   25.0
1   25.0
2   35.0
3   75.0
4   30.0
       ...
390   45.0
391   70.0
392   50.0
393   55.0
394   40.0
Name: G1, Length: 395, dtype: float64
</pre></td></tr>
<tr><td>Apply the same function to all the values in more than one columns</td><td>Lapply can take only one column at a time. To apply the function to more than one column, a simple “for” loop may be used. Similar functions are apply, sapply, tapply.</td><td>The ‘applymap’ function is used instead of ‘apply’ when the function has to be applied to more than one series.
student_score[['G1','G2','G3']].applymap(percent)

```
#Passing more than one series to a function
student_score[['G1', 'G2', 'G3']].applymap(percent)
```

	G1	G2	G3
0	25.0	30.0	30.0
1	25.0	25.0	30.0
2	35.0	40.0	50.0
3	75.0	70.0	75.0
4	30.0	50.0	50.0
...	...	...	...
390	45.0	45.0	45.0
391	70.0	80.0	80.0
392	50.0	40.0	35.0
393	55.0	60.0	50.0
394	40.0	45.0	45.0

395 rows × 3 columns</td></tr>
</table>

Function	R	Python
Custom Column names	***colnames(student_score) = cols***	***student_score.columns = cols***
Rename specific columns	***student_score %>% rename(school=col1)***	***student_score.rename(columns={'col1':'school'},inplace=True)***

R – Custom Column names

```r
#Change Column Names
cols = c("col1","col2","col3","col4","col5","col6","col7","col8","col9","col10","col11","col12","col13","col14","col15","col16","col17","col18","col19","col20","col21","col22","col23","col24","col25","col26","col27","col28","col29","col30","col31","col32","col33")
colnames(student_score)=cols
colnames(student_score)
```

```
 [1] "col1"  "col2"  "col3"  "col4"  "col5"  "col6"  "col7"  "col8"  "col9"  "col10" "col11" "col12" "col13"
[14] "col14" "col15" "col16" "col17" "col18" "col19" "col20" "col21" "col22" "col23" "col24" "col25" "col26"
[27] "col27" "col28" "col29" "col30" "col31" "col32" "col33"
```

Python – Custom Column names

```python
#Custom column names
cols = ["col1","col2","col3","col4","col5","col6","col7","col8","col9","col10","col11","col12","col13","col14","col15","col16","
student_score.columns = cols
student_score.columns
```

```
Index(['col1', 'col2', 'col3', 'col4', 'col5', 'col6', 'col7', 'col8', 'col9',
       'col10', 'col11', 'col12', 'col13', 'col14', 'col15', 'col16', 'col17',
       'col18', 'col19', 'col20', 'col21', 'col22', 'col23', 'col24', 'col25',
       'col26', 'col27', 'col28', 'col29', 'col30', 'col31', 'col32', 'col33',
       'col34'],
      dtype='object')
```

R – Rename specific columns

```r
#Rename specific columns
student_score %>% rename(school=col1)
```

school	col2	col3	col4	col5	col6	col7	col8	col9	col10
GP	F	18	U	GT3	A	4	4	at_home	teacher
GP	F	17	U	GT3	T	1	1	at_home	other
GP	F	15	U	LE3	T	1	1	at_home	other
GP	F	15	U	GT3	T	4	2	health	services
GP	F	16	U	GT3	T	3	3	other	other
GP	M	16	U	LE3	T	4	3	services	other
GP	M	16	U	LE3	T	2	2	other	other
GP	F	17	U	GT3	A	4	4	other	teacher
GP	M	15	U	LE3	A	3	2	services	other
GP	M	15	U	GT3	T	3	4	other	other

1-10 of 395 rows | 1-10 of 33 columns — Previous 1 2 3 4 5 6 ... 40 Next

Python – Rename specific columns

```python
#Rename specific column name
student_score.rename(columns={'col1':'school'},inplace=True)
student_score
```

	school	col2	col3	col4	col5	col6	col7	col8	col9	col10	...	col25	col26	col27	col28	col29	col30	col31	col32	col33	col34
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	1000
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	1001
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	1002
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1	1	5	2	15	14	15	1003
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	1004
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	4	4	5	4	11	9	9	9	1390
391	MS	M	17	U	LE3	T	3	1	services	services	...	4	5	3	4	2	3	14	16	16	1391
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	3	3	3	3	3	10	8	7	1392
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	1	3	4	5	0	11	12	10	1393
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	2	3	3	3	5	5	8	9	9	1394

395 rows × 34 columns

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/

<table>
<tr><th>Function</th><th>R</th><th>Python</th></tr>
<tr><td>Wide to long format</td><td>

scores_long <- melt(scores_wide, id.vars="Student", variable.name="Subject", value.name="Marks")

```{r}
scores_wide %>% head(5)
```

	Student <fctr>	G1 <int>	G2 <int>	G3 <int>
1	Student 1	5	6	6
2	Student 2	5	5	6
3	Student 3	7	8	10
4	Student 4	15	14	15
5	Student 5	6	10	10

5 rows

```{r}
#Wide to Long
scores_long <- melt(scores_wide, id.vars="Student", variable.name="Subject", value.name="Marks")
scores_long %>% head(5)
```

	Student <fctr>	Subject <fctr>	Marks <int>
1	Student 1	G1	5
2	Student 2	G1	5
3	Student 3	G1	7
4	Student 4	G1	15
5	Student 5	G1	6

</td><td>

pd.melt(scores_wide, id_vars=["Student"], var_name="Subject", value_name="Marks")

```
scores_wide.head()
```

	Student	G1	G2	G3
0	Student 1	5	6	6
1	Student 2	5	5	6
2	Student 3	7	8	10
3	Student 4	15	14	15
4	Student 5	6	10	10

```
#Wide to Long
scores_long=pd.melt(scores_wide, id_vars=["Student"], var_name="Subject", value_name="Marks")
scores_long.head()
```

	Student	Subject	Marks
0	Student 1	G1	5
1	Student 2	G1	5
2	Student 3	G1	7
3	Student 4	G1	15
4	Student 5	G1	6

</td></tr>
<tr><td>Long to wide format</td><td>

scores_wide_new <- spread(scores_long, "Subject", "Marks")

```{r}
#Long to wide
scores_wide_new <- spread(scores_long, "Subject", "Marks")
scores_wide_new %>% head(5)
```

	Student <fctr>	G1 <int>	G2 <int>	G3 <int>
1	Student 1	5	6	6
2	Student 10	14	15	15
3	Student 100	7	9	8
4	Student 101	7	7	5
5	Student 102	16	17	17

5 rows

</td><td>

scores_long.pivot(index="Student", columns = "Subject", values = "Marks")

```
scores_wide_new = scores_long.pivot(index="Student", columns = "Subject", values = "Marks")
scores_wide_new.head()
```

Subject	G1	G2	G3
Student			
Student 1	5	6	6
Student 10	14	15	15
Student 100	7	9	8
Student 101	7	7	5
Student 102	16	17	17

</td></tr>
</table>

Dhivya Karthic
https://www.linkedin.com/in/dhivyakarthic/