Data Wrangling with R

Bibek Sapkota

Import Data - Basics

task 1:Importing the library

```
library(readr)
```

Read Data

task 1:Reading the hsb2.csv data.

```
read_csv('hsb2.csv')
```

```
## Rows: 200 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): id, female, race, ses, schtyp, prog, read, write, math, science, s...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 200 x 11
##       id female  race   ses schtyp  prog  read write  math science socst
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>
##  1    70      0     4     1      1     1    57    52    41      47    57
##  2   121      1     4     2      1     3    68    59    53      63    61
##  3    86      0     4     3      1     1    44    33    54      58    31
##  4   141      0     4     3      1     3    63    44    47      53    56
##  5   172      0     4     2      1     2    47    52    57      53    61
##  6   113      0     4     2      1     2    44    52    51      63    61
##  7    50      0     3     2      1     1    50    59    42      53    61
##  8    11      0     1     2      1     2    34    46    45      39    36
##  9    84      0     4     2      1     1    63    57    54      58    51
## 10    48      0     3     2      1     2    57    55    52      50    51
## # i 190 more rows
```

task 2:specifying the column types.

```
spec_csv('hsb2.csv')
```

```
## cols(
##   id = col_double(),
##   female = col_double(),
##   race = col_double(),
##   ses = col_double(),
##   schtyp = col_double(),
##   prog = col_double(),
##   read = col_double(),
##   write = col_double(),
##   math = col_double(),
##   science = col_double(),
##   socst = col_double()
## )
```

task 3:Reading the hsb3.csv data.

```
read_csv('hsb3.csv')
```

```
## New names:
## Rows: 199 Columns: 11
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (11): 70, 0, 4, 1...4, 1...5, 1...6, 57...7, 52, 41, 47, 57...11
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `1` -> `1...4`
## * `1` -> `1...5`
## * `1` -> `1...6`
## * `57` -> `57...7`
## * `57` -> `57...11`
```

```
## # A tibble: 199 x 11
##     `70`   `0`   `4` `1...4` `1...5` `1...6` `57...7`  `52`  `41`  `47`
##    <dbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>    <dbl> <dbl> <dbl> <dbl>
##  1   121     1     4       2       1       3       68    59    53    63
##  2    86     0     4       3       1       1       44    33    54    58
##  3   141     0     4       3       1       3       63    44    47    53
##  4   172     0     4       2       1       2       47    52    57    53
##  5   113     0     4       2       1       2       44    52    51    63
##  6    50     0     3       2       1       1       50    59    42    53
##  7    11     0     1       2       1       2       34    46    45    39
##  8    84     0     4       2       1       1       63    57    54    58
##  9    48     0     3       2       1       2       57    55    52    50
## 10    75     0     4       2       1       3       60    46    51    53
## # i 189 more rows
## # i 1 more variable: `57...11` <dbl>
```

task 4:treating the first row of the file as data rather than column names in the resulting dataframe.

```
read_csv('hsb3.csv', col_names = FALSE)
```

```
## Rows: 200 Columns: 11
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 200 x 11
##       X1    X2    X3    X4    X5    X6    X7    X8    X9   X10   X11
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1    70     0     4     1     1     1    57    52    41    47    57
##  2   121     1     4     2     1     3    68    59    53    63    61
##  3    86     0     4     3     1     1    44    33    54    58    31
##  4   141     0     4     3     1     3    63    44    47    53    56
##  5   172     0     4     2     1     2    47    52    57    53    61
##  6   113     0     4     2     1     2    44    52    51    63    61
##  7    50     0     3     2     1     1    50    59    42    53    61
##  8    11     0     1     2     1     2    34    46    45    39    36
##  9    84     0     4     2     1     1    63    57    54    58    51
## 10    48     0     3     2     1     2    57    55    52    50    51
## # i 190 more rows
```

task 5:specifying custom column names using the vector cnames for the resulting dataframe columns.

```
cnames <- c("id", "gender", "race", "socio_economic_status", "school_type", "program", "read", "write",
read_csv('hsb3.csv', col_names = cnames)
```

```
## Rows: 200 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): id, gender, race, socio_economic_status, school_type, program, rea...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 200 x 11
##       id gender  race socio_economic_status school_type program  read write
##    <dbl>  <dbl> <dbl>                 <dbl>       <dbl>   <dbl> <dbl> <dbl>
##  1    70      0     4                     1           1       1    57    52
##  2   121      1     4                     2           1       3    68    59
##  3    86      0     4                     3           1       1    44    33
##  4   141      0     4                     3           1       3    63    44
##  5   172      0     4                     2           1       2    47    52
##  6   113      0     4                     2           1       2    44    52
##  7    50      0     3                     2           1       1    50    59
##  8    11      0     1                     2           1       2    34    46
##  9    84      0     4                     2           1       1    63    57
## 10    48      0     3                     2           1       2    57    55
## # i 190 more rows
## # i 3 more variables: math <dbl>, science <dbl>, socst <dbl>
```

Skip Lines

task 1:Reading the data without skipping any lines/rows and observe the result.

```
read_csv('hsb4.csv')
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 203 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): # A dataset containing demographic information and standardized
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 203 x 1
##    `# A dataset containing demographic information and standardized`
##    <chr>
##  1 # test scores of high school students.
##  2 # http://www.ats.ucla.edu/stat/spss/whatstat/whatstat.htm
##  3 id,female,race,ses,schtyp,prog,read,write,math,science,socst
##  4 70,0,4,1,1,1,57,52,41,47,57
##  5 121,1,4,2,1,3,68,59,53,63,61
##  6 86,0,4,3,1,1,44,33,54,58,31
##  7 141,0,4,3,1,3,63,44,47,53,56
##  8 172,0,4,2,1,2,47,52,57,53,61
##  9 113,0,4,2,1,2,44,52,51,63,61
## 10 50,0,3,2,1,1,50,59,42,53,61
## # i 193 more rows
```

task 2:Skiping the first 3 lines as they contain information about the data set which we do not need.

```
read_csv('hsb4.csv', skip = 3)
```

```
## Rows: 200 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): id, female, race, ses, schtyp, prog, read, write, math, science, s...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 200 x 11
##       id female  race   ses schtyp  prog  read write  math science socst
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>
##  1    70      0     4     1      1     1    57    52    41      47    57
##  2   121      1     4     2      1     3    68    59    53      63    61
##  3    86      0     4     3      1     1    44    33    54      58    31
```

```
##  4   141      0     4     3      1     3    63    44    47      53    56
##  5   172      0     4     2      1     2    47    52    57      53    61
##  6   113      0     4     2      1     2    44    52    51      63    61
##  7    50      0     3     2      1     1    50    59    42      53    61
##  8    11      0     1     2      1     2    34    46    45      39    36
##  9    84      0     4     2      1     1    63    57    54      58    51
## 10    48      0     3     2      1     2    57    55    52      50    51
## # i 190 more rows
```

Maximum Lines

task 1:Reading the first 120 rows from the hsb2 dataset.

```
read_csv('hsb2.csv', n_max = 120)
```

```
## Rows: 120 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (11): id, female, race, ses, schtyp, prog, read, write, math, science, s...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 120 x 11
##       id female  race   ses schtyp  prog  read write  math science socst
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>
##  1    70      0     4     1      1     1    57    52    41      47    57
##  2   121      1     4     2      1     3    68    59    53      63    61
##  3    86      0     4     3      1     1    44    33    54      58    31
##  4   141      0     4     3      1     3    63    44    47      53    56
##  5   172      0     4     2      1     2    47    52    57      53    61
##  6   113      0     4     2      1     2    44    52    51      63    61
##  7    50      0     3     2      1     1    50    59    42      53    61
##  8    11      0     1     2      1     2    34    46    45      39    36
##  9    84      0     4     2      1     1    63    57    54      58    51
## 10    48      0     3     2      1     2    57    55    52      50    51
## # i 110 more rows
```

Column Types

task 1:Specifying the data types for each column explicitly, where certain columns are defined as integers and others as factor variables with specified levels.

```
read_csv('hsb2.csv', col_types = list(
  col_integer(), col_factor(levels = c("0", "1")),
  col_factor(levels = c("1", "2", "3", "4")), col_factor(levels = c("1", "2", "3")),
  col_factor(levels = c("1", "2")), col_factor(levels = c("1", "2", "3")),
  col_integer(), col_integer(), col_integer(), col_integer(),
  col_integer())
)
```

```
## # A tibble: 200 x 11
##       id female race  ses   schtyp prog   read write  math science socst
##    <int> <fct>  <fct> <fct> <fct>  <fct> <int> <int> <int>   <int> <int>
##  1    70 0      4     1     1      1        57    52    41      47    57
##  2   121 1      4     2     1      3        68    59    53      63    61
##  3    86 0      4     3     1      1        44    33    54      58    31
##  4   141 0      4     3     1      3        63    44    47      53    56
##  5   172 0      4     2     1      2        47    52    57      53    61
##  6   113 0      4     2     1      2        44    52    51      63    61
##  7    50 0      3     2     1      1        50    59    42      53    61
##  8    11 0      1     2     1      2        34    46    45      39    36
##  9    84 0      4     2     1      1        63    57    54      58    51
## 10    48 0      3     2     1      2        57    55    52      50    51
## # i 190 more rows
```

task 2:Secifying column types only for selected columns (id as integer, prog as a factor with specified levels, and read as integer).

```
read_csv('hsb2.csv', col_types = cols_only(id = col_integer(),
  prog = col_factor(levels = c("1", "2", "3")), read = col_integer())
)
```

```
## # A tibble: 200 x 3
##       id prog   read
##    <int> <fct> <int>
##  1    70 1        57
##  2   121 3        68
##  3    86 1        44
##  4   141 3        63
##  5   172 2        47
##  6   113 2        44
##  7    50 1        50
##  8    11 2        34
##  9    84 1        63
## 10    48 2        57
## # i 190 more rows
```

Import Data - Advanced

task 1:Reading the library

```
library(readxl)
library(haven)
```

List Sheets

task 1:Seeing how many sheets are present in sample.xls file and their respective names using excel_sheets().

```
excel_sheets('sample.xls')
```

```
## [1] "ecom"
```

Read Sheet

task 1:Reading data from the ecom sheet of the sample.xls file using read_excel(), specifying the sheet number.

```
read_excel('sample.xls', sheet = 1)
```

```
## # A tibble: 7 x 5
##   channel        users new_users sessions bounce_rate
##   <chr>          <dbl>     <dbl>    <dbl> <chr>
## 1 Organic Search 43296     40238    50810 48.72%
## 2 Direct         12916     12311    16419 49.27%
## 3 Referral       10983      7636    18105 22.26%
## 4 Social         10346     10029    11101 61.92%
## 5 Display         5564      4790     7220 83.30%
## 6 Paid Search     2687      2205     3438 38.02%
## 7 Affiliates      1773      1585     2167 55.75%
```

task 2:Specifying the sheet name.

```
read_excel('sample.xls', sheet = 'ecom')
```

```
## # A tibble: 7 x 5
##   channel        users new_users sessions bounce_rate
##   <chr>          <dbl>     <dbl>    <dbl> <chr>
## 1 Organic Search 43296     40238    50810 48.72%
## 2 Direct         12916     12311    16419 49.27%
## 3 Referral       10983      7636    18105 22.26%
## 4 Social         10346     10029    11101 61.92%
## 5 Display         5564      4790     7220 83.30%
## 6 Paid Search     2687      2205     3438 38.02%
## 7 Affiliates      1773      1585     2167 55.75%
```

Read Specific Cells

task 1:Reading data from first 4 rows of columns B and C, we will specify the range as “B1:C4”

```
read_excel('sample.xls', sheet = 1, range = "B1:C4")
```

```
## # A tibble: 3 x 2
##   users new_users
##   <dbl>     <dbl>
## 1 43296     40238
## 2 12916     12311
## 3 10983      7636
```

task 2:Reading data from first 5 rows of columns A, B and C, we will specify the range as “A1:C5”

```
read_excel('sample.xls', sheet = 1, range = "A1:C5")
```

```
## # A tibble: 4 x 3
##   channel        users new_users
##   <chr>          <dbl>     <dbl>
## 1 Organic Search 43296     40238
## 2 Direct         12916     12311
## 3 Referral       10983      7636
## 4 Social         10346     10029
```

task 3:Reading the data from first 3 rows and 2 columns starting from A4.

```
read_excel('sample.xls', sheet = 1, col_names = FALSE,
  range = anchored("A4", dim = c(3, 2)))
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
```

```
## # A tibble: 3 x 2
##   ...1      ...2
##   <chr>    <dbl>
## 1 Referral 10983
## 2 Social   10346
## 3 Display   5564
```

task 4:Reading data from the first 6 rows and 4 columns.

```
read_excel('sample.xls', sheet = 1,
  range = cell_limits(c(1, 1), c(6, 4)))
```

```
## # A tibble: 5 x 4
##   channel        users new_users sessions
##   <chr>          <dbl>     <dbl>    <dbl>
## 1 Organic Search 43296     40238    50810
## 2 Direct         12916     12311    16419
## 3 Referral       10983      7636    18105
## 4 Social         10346     10029    11101
## 5 Display         5564      4790     7220
```

task 5:Reading data from all the rows from the second column onwards.

```
read_excel('sample.xls', sheet = 1,
  range = cell_limits(c(1, 2), c(NA, NA)))
```

```
## # A tibble: 7 x 4
##   users new_users sessions bounce_rate
##   <dbl>     <dbl>    <dbl> <chr>
## 1 43296     40238    50810 48.72%
## 2 12916     12311    16419 49.27%
## 3 10983      7636    18105 22.26%
## 4 10346     10029    11101 61.92%
## 5  5564      4790     7220 83.30%
## 6  2687      2205     3438 38.02%
## 7  1773      1585     2167 55.75%
```

task 6:Reading data from the first 4 rows of columns B and C.(Method 1)

```
read_excel('sample.xls', sheet = 1,
  range = "B1:C4")
```

```
## # A tibble: 3 x 2
##   users new_users
##   <dbl>     <dbl>
## 1 43296     40238
## 2 12916     12311
## 3 10983      7636
```

task 7:Reading data from the first 4 rows of columns B and C.(Method 2)

```
read_excel('sample.xls', sheet = 1,
  range = anchored("B1", dim = c(4, 2)))
```

```
## # A tibble: 3 x 2
##   users new_users
##   <dbl>     <dbl>
## 1 43296     40238
## 2 12916     12311
## 3 10983      7636
```

task 8:Reading data from the first 4 rows of columns B and C.(Method 3)

```
read_excel('sample.xls', sheet = 1,
  range = cell_limits(c(1, 2), c(4, 3)))
```

```
## # A tibble: 3 x 2
##   users new_users
##   <dbl>     <dbl>
## 1 43296     40238
## 2 12916     12311
## 3 10983      7636
```

Read Specific Rows

task 1:Reading the first 4 rows of data from the sample.xls file.

```
read_excel('sample.xls', sheet = 1, range = cell_rows(1:4))
```

```
## # A tibble: 3 x 5
##   channel        users new_users sessions bounce_rate
##   <chr>          <dbl>     <dbl>    <dbl> <chr>
## 1 Organic Search 43296     40238    50810 48.72%
## 2 Direct         12916     12311    16419 49.27%
## 3 Referral       10983      7636    18105 22.26%
```

Read Single Column

task 1:Reading the second column from the sample.xls file using cell_cols().

```
read_excel('sample.xls', sheet = 1, range = cell_cols(2))
```

```
## # A tibble: 7 x 1
##   users
##   <dbl>
## 1 43296
## 2 12916
## 3 10983
## 4 10346
## 5  5564
## 6  2687
## 7  1773
```

Read Multiple Columns

task 1:Reading the 2nd, 4th and 6th column from the sample.xls file.

```
read_excel('sample.xls', sheet = 1, range = cell_cols(c(2, 4, 6)))
```

```
## New names:
## * `` -> `...5`
```

```
## # A tibble: 7 x 5
##   users new_users sessions bounce_rate ...5
##   <dbl>     <dbl>    <dbl> <chr>       <lgl>
## 1 43296     40238    50810 48.72%      NA
## 2 12916     12311    16419 49.27%      NA
## 3 10983      7636    18105 22.26%      NA
## 4 10346     10029    11101 61.92%      NA
## 5  5564      4790     7220 83.30%      NA
## 6  2687      2205     3438 38.02%      NA
## 7  1773      1585     2167 55.75%      NA
```

task 2:Reading data from the 2nd column upto and including the 6th column.

```
read_excel('sample.xls', sheet = 1, range = cell_cols(c(2:6)))
```

```
## New names:
## * `` -> `...5`
```

```
## # A tibble: 7 x 5
##   users new_users sessions bounce_rate ...5
##   <dbl>     <dbl>    <dbl> <chr>       <lgl>
## 1 43296     40238    50810 48.72%      NA
## 2 12916     12311    16419 49.27%      NA
## 3 10983      7636    18105 22.26%      NA
## 4 10346     10029    11101 61.92%      NA
## 5  5564      4790     7220 83.30%      NA
## 6  2687      2205     3438 38.02%      NA
## 7  1773      1585     2167 55.75%      NA
```

Statistical Softwares

task 1:Reading data using read_stata.

```
read_stata('airline.dta')
```

```
## # A tibble: 32 x 6
##     year     y     w     r     l     k
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  1948  1.21 0.243 0.145  1.41 0.612
##  2  1949  1.35 0.260 0.218  1.38 0.559
##  3  1950  1.57 0.278 0.316  1.39 0.573
##  4  1951  1.95 0.297 0.394  1.55 0.564
##  5  1952  2.27 0.310 0.356  1.80 0.574
##  6  1953  2.73 0.322 0.359  1.93 0.711
##  7  1954  3.03 0.335 0.403  1.96 0.776
##  8  1955  3.56 0.350 0.396  2.12 0.827
##  9  1956  3.98 0.361 0.382  2.43 0.800
## 10  1957  4.42 0.379 0.305  2.71 0.921
## # i 22 more rows
```

task 2:Reading data using read_spss.

```
read_spss('employee.sav')
```

```
## # A tibble: 474 x 9
##       id gender     educ      jobcat  salary salbegin jobtime prevexp   minority
##    <dbl> <chr+lbl>  <dbl+lbl> <dbl+l> <dbl+> <dbl+lb> <dbl+l> <dbl+lbl> <dbl+lb>
##  1     1 m [Male]   15 [15]   3 [Man~ 57000  27000    98      144       0 [No]
##  2     2 m [Male]   16 [16]   1 [Cle~ 40200  18750    98       36       0 [No]
##  3     3 f [Female] 12 [12]   1 [Cle~ 21450  12000    98      381       0 [No]
##  4     4 f [Female]  8 [8]    1 [Cle~ 21900  13200    98      190       0 [No]
##  5     5 m [Male]   15 [15]   1 [Cle~ 45000  21000    98      138       0 [No]
##  6     6 m [Male]   15 [15]   1 [Cle~ 32100  13500    98       67       0 [No]
##  7     7 m [Male]   15 [15]   1 [Cle~ 36000  18750    98      114       0 [No]
##  8     8 f [Female] 12 [12]   1 [Cle~ 21900   9750    98        0 [mis~ 0 [No]
##  9     9 f [Female] 15 [15]   1 [Cle~ 27900  12750    98      115       0 [No]
## 10    10 f [Female] 12 [12]   1 [Cle~ 24000  13500    98      244       0 [No]
## # i 464 more rows
```

task 3:Reading data using read_sas.

```
read_sas('airline.sas7bdat')
```

```
## # A tibble: 32 x 6
##     YEAR     Y     W     R     L     K
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  1948  1.21 0.243 0.145  1.41 0.612
##  2  1949  1.35 0.260 0.218  1.38 0.559
##  3  1950  1.57 0.278 0.316  1.39 0.573
##  4  1951  1.95 0.297 0.394  1.55 0.564
##  5  1952  2.27 0.310 0.356  1.80 0.574
```

```
##  6  1953  2.73 0.322 0.359  1.93 0.711
##  7  1954  3.03 0.335 0.403  1.96 0.776
##  8  1955  3.56 0.350 0.396  2.12 0.827
##  9  1956  3.98 0.361 0.382  2.43 0.800
## 10  1957  4.42 0.379 0.305  2.71 0.921
## # i 22 more rows
```

dplyr Basics

task 1:Importing library

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
```

task 2: Reading the data.

```
ecom <-
  read_csv('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/web.csv',
    col_types = cols_only(device = col_factor(levels = c("laptop", "tablet", "mobile")),
      referrer = col_factor(levels = c("bing", "direct", "social", "yahoo", "google")),
      purchase = col_logical(), n_pages = col_double(), n_visit = col_double(),
      duration = col_double(), order_value = col_double(), order_items = col_double()
    )
  )

ecom
```

```
## # A tibble: 1,000 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 google   laptop      10       1      693 FALSE              0           0
##  2 yahoo    tablet       9       1      459 FALSE              0           0
##  3 direct   laptop       0       1      996 FALSE              0           0
##  4 bing     tablet       3      18      468 TRUE               6         434
##  5 yahoo    mobile       9       1      955 FALSE              0           0
##  6 yahoo    laptop       5       5      135 FALSE              0           0
##  7 yahoo    mobile      10       1       75 FALSE              0           0
##  8 direct   mobile      10       1      908 FALSE              0           0
##  9 bing     mobile       3      19      209 FALSE              0           0
## 10 google   mobile       6       1      208 FALSE              0           0
## # i 990 more rows
```

Average Order Value by Devices

task 1:Calculating the average Order Value by devices.

```
ecom %>%
  filter(purchase) %>%
  select(device, order_value) %>%
  group_by(device) %>%
  summarise_all(funs(revenue = sum, orders = n())) %>%
  mutate(
    aov = revenue / orders
  ) %>%
  select(device, aov) %>%
  arrange(aov)
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 3 x 2
##   device   aov
##   <fct>  <dbl>
## 1 tablet 1426.
## 2 mobile 1431.
## 3 laptop 1824.
```

Filter Rows

task 1:Filtering the device mobile and Displaying it.

```
filter(ecom, device == "mobile")
```

```
## # A tibble: 344 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 yahoo    mobile       9       1      955 FALSE              0           0
##  2 yahoo    mobile      10       1       75 FALSE              0           0
##  3 direct   mobile      10       1      908 FALSE              0           0
##  4 bing     mobile       3      19      209 FALSE              0           0
##  5 google   mobile       6       1      208 FALSE              0           0
##  6 direct   mobile       9      14      406 TRUE               3         651
##  7 yahoo    mobile       7       1       19 FALSE              7        2423
##  8 google   mobile       5       1      147 FALSE              0           0
##  9 bing     mobile       0       7      196 FALSE              4         237
## 10 google   mobile      10       1      338 FALSE              0           0
## # i 334 more rows
```

task 2:Filtering the device mobile and Displaying the purchased one only.

```
filter(ecom, device == "mobile", purchase)
```

```
## # A tibble: 36 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 direct   mobile       9      14      406 TRUE               3         651
##  2 bing     mobile       4      20      440 TRUE               3         184
##  3 bing     mobile       3      18      288 TRUE               6         764
##  4 social   mobile      10      11      242 TRUE               4         287
##  5 yahoo    mobile       6      14      322 TRUE               3        1443
##  6 google   mobile       1      18      252 TRUE               3        2449
##  7 social   mobile       7      16      352 TRUE              10        2824
##  8 direct   mobile       4      18      324 TRUE               3        1670
##  9 social   mobile       1      20      520 TRUE               5        1021
## 10 yahoo    mobile       0      13      351 TRUE              10         288
## # i 26 more rows
```

task 3:Filtering the device tablet and N-page which is less than 15.

```
filter(ecom, device == "tablet", purchase, n_pages < 15)
```

```
## # A tibble: 12 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 social   tablet       7      10      290 TRUE               9        1304
##  2 yahoo    tablet       2      14      364 TRUE               6        1667
##  3 google   tablet       7      12      324 TRUE               2        1358
##  4 direct   tablet       3      12      324 TRUE              10        1257
##  5 yahoo    tablet       0      13      390 TRUE               5        1748
##  6 social   tablet       2      12      300 TRUE               2        2754
##  7 direct   tablet       6      13      338 TRUE               5         683
##  8 yahoo    tablet       2      10      280 TRUE               4         293
##  9 social   tablet      10      10      290 TRUE               9          37
## 10 direct   tablet       3      10      260 TRUE               7         980
## 11 google   tablet       9      14      308 TRUE               7        2436
## 12 social   tablet      10      11      330 TRUE               1        2171
```

task 4:Filtering purchased only.

```
filter(ecom, purchase)
```

```
## # A tibble: 103 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 bing     tablet       3      18      468 TRUE               6         434
##  2 direct   mobile       9      14      406 TRUE               3         651
##  3 bing     tablet       5      16      368 TRUE               6        1049
##  4 social   tablet       7      10      290 TRUE               9        1304
##  5 direct   tablet       2      19      342 TRUE               5         622
##  6 social   tablet       9      20      420 TRUE               7        1613
```

```
##  7 bing     mobile       4      20      440 TRUE              3         184
##  8 yahoo    tablet       2      16      480 TRUE              9         286
##  9 bing     mobile       3      18      288 TRUE              6         764
## 10 yahoo    tablet       2      14      364 TRUE              6        1667
## # i 93 more rows
```

Select Columns

task 1:Selecting device and duration columns only.

```
select(ecom, device, duration)
```

```
## # A tibble: 1,000 x 2
##    device duration
##    <fct>     <dbl>
##  1 laptop      693
##  2 tablet      459
##  3 laptop      996
##  4 tablet      468
##  5 mobile      955
##  6 laptop      135
##  7 mobile       75
##  8 mobile      908
##  9 mobile      209
## 10 mobile      208
## # i 990 more rows
```

task 2:Selecting the columns from referrer to order_items.

```
select(ecom, referrer:order_items)
```

```
## # A tibble: 1,000 x 7
##    referrer device n_visit n_pages duration purchase order_items
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>
##  1 google   laptop      10       1      693 FALSE              0
##  2 yahoo    tablet       9       1      459 FALSE              0
##  3 direct   laptop       0       1      996 FALSE              0
##  4 bing     tablet       3      18      468 TRUE               6
##  5 yahoo    mobile       9       1      955 FALSE              0
##  6 yahoo    laptop       5       5      135 FALSE              0
##  7 yahoo    mobile      10       1       75 FALSE              0
##  8 direct   mobile      10       1      908 FALSE              0
##  9 bing     mobile       3      19      209 FALSE              0
## 10 google   mobile       6       1      208 FALSE              0
## # i 990 more rows
```

task 3:Removing the n_page and duration columns.

```
select(ecom, -n_pages, -duration)
```

```
## # A tibble: 1,000 x 6
##    referrer device n_visit purchase order_items order_value
##    <fct>    <fct>    <dbl> <lgl>          <dbl>       <dbl>
##  1 google   laptop      10 FALSE              0           0
##  2 yahoo    tablet       9 FALSE              0           0
##  3 direct   laptop       0 FALSE              0           0
##  4 bing     tablet       3 TRUE               6         434
##  5 yahoo    mobile       9 FALSE              0           0
##  6 yahoo    laptop       5 FALSE              0           0
##  7 yahoo    mobile      10 FALSE              0           0
##  8 direct   mobile      10 FALSE              0           0
##  9 bing     mobile       3 FALSE              0           0
## 10 google   mobile       6 FALSE              0           0
## # i 990 more rows
```

task 4:Selecting device and order_value.

```
select(ecom, device, order_value)
```

```
## # A tibble: 1,000 x 2
##    device order_value
##    <fct>        <dbl>
##  1 laptop           0
##  2 tablet           0
##  3 laptop           0
##  4 tablet         434
##  5 mobile           0
##  6 laptop           0
##  7 mobile           0
##  8 mobile           0
##  9 mobile           0
## 10 mobile           0
## # i 990 more rows
```

task 5:Filtering the putchased and then displaying its device and order_value.

```
ecom1 <- filter(ecom, purchase)

ecom2 <- select(ecom1, device, order_value)

ecom2
```

```
## # A tibble: 103 x 2
##    device order_value
##    <fct>        <dbl>
##  1 tablet         434
##  2 mobile         651
##  3 tablet        1049
##  4 tablet        1304
##  5 tablet         622
##  6 tablet        1613
##  7 mobile         184
##  8 tablet         286
```

```
##  9 mobile         764
## 10 tablet        1667
## # i 93 more rows
```

Grouping Data

task 1:spliting the referrer columns.

```
group_by(ecom, referrer)
```

```
## # A tibble: 1,000 x 8
## # Groups:   referrer [5]
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 google   laptop      10       1      693 FALSE              0           0
##  2 yahoo    tablet       9       1      459 FALSE              0           0
##  3 direct   laptop       0       1      996 FALSE              0           0
##  4 bing     tablet       3      18      468 TRUE               6         434
##  5 yahoo    mobile       9       1      955 FALSE              0           0
##  6 yahoo    laptop       5       5      135 FALSE              0           0
##  7 yahoo    mobile      10       1       75 FALSE              0           0
##  8 direct   mobile      10       1      908 FALSE              0           0
##  9 bing     mobile       3      19      209 FALSE              0           0
## 10 google   mobile       6       1      208 FALSE              0           0
## # i 990 more rows
```

task 2:split ecom2 by device type.

```
ecom3 <- group_by(ecom2, device)
ecom3
```

```
## # A tibble: 103 x 2
## # Groups:   device [3]
##    device order_value
##    <fct>        <dbl>
##  1 tablet         434
##  2 mobile         651
##  3 tablet        1049
##  4 tablet        1304
##  5 tablet         622
##  6 tablet        1613
##  7 mobile         184
##  8 tablet         286
##  9 mobile         764
## 10 tablet        1667
## # i 93 more rows
```

Summarise Data

task 1:Split data by referrer type.

```
step_1 <- group_by(ecom, referrer)
```

task 2:Compute average number of pages.

```
step_2 <- summarise(step_1, mean(n_pages))
step_2
```

```
## # A tibble: 5 x 2
##   referrer `mean(n_pages)`
##   <fct>              <dbl>
## 1 bing                6.13
## 2 direct              6.38
## 3 social              5.42
## 4 yahoo               5.99
## 5 google              5.73
```

task 3:Computing average number of pages.

```
step_2 <- summarise(step_1, mean(n_pages), median(n_pages))
step_2
```

```
## # A tibble: 5 x 3
##   referrer `mean(n_pages)` `median(n_pages)`
##   <fct>              <dbl>             <dbl>
## 1 bing                6.13                 1
## 2 direct              6.38                 1
## 3 social              5.42                 1
## 4 yahoo               5.99                 2
## 5 google              5.73                 1
```

task 4:Selecting relevant columns.

```
step_1 <- select(ecom, referrer, order_value)
```

task 5:spliting data by referrer type.

```
step_2 <- group_by(step_1, referrer)
```

task 6:computing average number of pages.

```
step_3 <- summarise_all(step_2, funs(mean))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
step_3
```

```
## # A tibble: 5 x 2
##   referrer order_value
##   <fct>          <dbl>
## 1 bing            316.
## 2 direct          441.
## 3 social          380.
## 4 yahoo           470.
## 5 google          328.
```

task 7:Selecting relevant columns.

```
step_1 <- select(ecom, referrer, order_value)
```

task 8:Spliting data by referrer type.

```
step_2 <- group_by(step_1, referrer)
```

task 9:Computing mean and median number of pages.

```
step_3 <- summarise_all(step_2, funs(mean, median))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
step_3
```

```
## # A tibble: 5 x 3
##   referrer  mean median
##   <fct>    <dbl>  <dbl>
## 1 bing      316.      0
## 2 direct    441.      0
## 3 social    380.      0
## 4 yahoo     470.      0
## 5 google    328.      0
```

task 10:Summarizing the revenue and orders.

```
ecom4 <- summarise(ecom3, revenue = sum(order_value),
          orders = n())
ecom4
```

```
## # A tibble: 3 x 3
##   device revenue orders
##   <fct>    <dbl>  <int>
## 1 laptop   56531     31
## 2 tablet   51321     36
## 3 mobile   51504     36
```

task 11:Summarizing the revenue and orders using funs.

```
ecom4 <- summarise_all(ecom3, funs(revenue = sum, orders = n()))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
ecom4
```

```
## # A tibble: 3 x 3
##   device revenue orders
##   <fct>    <dbl>  <int>
## 1 laptop   56531     31
## 2 tablet   51321     36
## 3 mobile   51504     36
```

Create Columns

task 1:Selecting duration and n_pages from ecom.

```
mutate_1 <- select(ecom, n_pages, duration)
mutate(mutate_1, avg_page_time = duration / n_pages)
```

```
## # A tibble: 1,000 x 3
##    n_pages duration avg_page_time
##      <dbl>    <dbl>         <dbl>
##  1       1      693           693
##  2       1      459           459
##  3       1      996           996
##  4      18      468            26
##  5       1      955           955
##  6       5      135            27
##  7       1       75            75
##  8       1      908           908
##  9      19      209            11
## 10       1      208           208
## # i 990 more rows
```

task 2:creating another column sqrt_avg_page_time i.e. square root of the average time on page using avg_page_time.

```
mutate(mutate_1,
       avg_page_time = duration / n_pages,
       sqrt_avg_page_time = sqrt(avg_page_time))
```

```
## # A tibble: 1,000 x 4
##    n_pages duration avg_page_time sqrt_avg_page_time
##      <dbl>    <dbl>         <dbl>              <dbl>
##  1       1      693           693               26.3
##  2       1      459           459               21.4
##  3       1      996           996               31.6
##  4      18      468            26                5.10
##  5       1      955           955               30.9
##  6       5      135            27                5.20
##  7       1       75            75                8.66
##  8       1      908           908               30.1
##  9      19      209            11                3.32
## 10       1      208           208               14.4
## # i 990 more rows
```

task 3:Creating new columns using mutate.

```
ecom5 <- mutate(ecom4, aov = revenue / orders)
ecom5
```

```
## # A tibble: 3 x 4
##   device revenue orders   aov
##   <fct>    <dbl>  <int> <dbl>
## 1 laptop   56531     31 1824.
## 2 tablet   51321     36 1426.
## 3 mobile   51504     36 1431.
```

task 3:Selecting device and aov columns.

```
ecom6 <- select(ecom5, device, aov)
ecom6
```

```
## # A tibble: 3 x 2
##   device   aov
##   <fct>  <dbl>
## 1 laptop 1824.
## 2 tablet 1426.
## 3 mobile 1431.
```

Arrange Data

task 1:Arranging data by n_page columns.

```r
arrange(ecom, n_pages)
```

```
## # A tibble: 1,000 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 google   laptop      10       1      693 FALSE              0           0
##  2 yahoo    tablet       9       1      459 FALSE              0           0
##  3 direct   laptop       0       1      996 FALSE              0           0
##  4 yahoo    mobile       9       1      955 FALSE              0           0
##  5 yahoo    mobile      10       1       75 FALSE              0           0
##  6 direct   mobile      10       1      908 FALSE              0           0
##  7 google   mobile       6       1      208 FALSE              0           0
##  8 direct   laptop       9       1      738 FALSE              0           0
##  9 yahoo    mobile       7       1       19 FALSE              7        2423
## 10 bing     laptop       1       1      995 FALSE              0           0
## # i 990 more rows
```

task 2:Arranging n_page columns in descending order.

```r
arrange(ecom , desc(n_pages))
```

```
## # A tibble: 1,000 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 social   tablet       9      20      420 TRUE               7        1613
##  2 bing     mobile       4      20      440 TRUE               3         184
##  3 yahoo    tablet       0      20      200 FALSE              0           0
##  4 direct   tablet       6      20      580 TRUE               5        1155
##  5 social   mobile       1      20      520 TRUE               5        1021
##  6 google   mobile       8      20      300 TRUE               7        2091
##  7 social   laptop       4      20      200 FALSE              0           0
##  8 yahoo    mobile       3      20      480 FALSE              0           0
##  9 social   laptop      10      20      280 TRUE               1        2011
## 10 yahoo    mobile       2      20      240 FALSE              0           0
## # i 990 more rows
```

task 3:Arranging data first by number of visits and then by number of pages in a descending order.

```r
arrange(ecom, n_visit, desc(n_pages))
```

```
## # A tibble: 1,000 x 8
##    referrer device n_visit n_pages duration purchase order_items order_value
##    <fct>    <fct>    <dbl>   <dbl>    <dbl> <lgl>          <dbl>       <dbl>
##  1 yahoo    tablet       0      20      200 FALSE              0           0
##  2 google   laptop       0      19      418 TRUE               2         996
##  3 bing     laptop       0      18      180 FALSE              0           0
##  4 yahoo    laptop       0      18      522 TRUE               8        1523
##  5 direct   tablet       0      18      252 FALSE              0           0
##  6 social   laptop       0      17      204 FALSE              0           0
##  7 bing     laptop       0      17      272 TRUE               9        1384
##  8 bing     mobile       0      16      272 FALSE              0           0
```

```
##  9 yahoo    mobile       0      15      255 FALSE              0           0
## 10 direct   laptop       0      15      255 FALSE              0           0
## # i 990 more rows
```

task 4:Arranging aov column in descending order.

```
arrange(ecom6, aov)
```

```
## # A tibble: 3 x 2
##   device   aov
##   <fct>  <dbl>
## 1 tablet 1426.
## 2 mobile 1431.
## 3 laptop 1824.
```

AOV by Devices

task 1:combine all the code from the above steps.

```
ecom1 <- filter(ecom, purchase)
ecom2 <- select(ecom1, device, order_value)
ecom3 <- group_by(ecom2, device)
ecom4 <- summarise_all(ecom3, funs(revenue = sum, orders = n()))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
ecom5 <- mutate(ecom4, aov = revenue / orders)
ecom6 <- select(ecom5, device, aov)
ecom7 <- arrange(ecom6, aov)
ecom7
```

```
## # A tibble: 3 x 2
##   device   aov
##   <fct>  <dbl>
## 1 tablet 1426.
## 2 mobile 1431.
## 3 laptop 1824.
```

task 2: without creating the intermediate data (ecom1 - ecom6)? .using the %>% operator to chain the steps and get rid of the intermediate data.

```
ecom %>%
  filter(purchase) %>%
  select(device, order_value) %>%
  group_by(device) %>%
  summarise_all(funs(revenue = sum, orders = n())) %>%
  mutate(
    aov = revenue / orders
  ) %>%
  select(device, aov) %>%
  arrange(aov)
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 3 x 2
##   device   aov
##   <fct>  <dbl>
## 1 tablet 1426.
## 2 mobile 1431.
## 3 laptop 1824.
```

Joining Tables using dplyr

task 1:Loading the packages.

```
library(dplyr)
library(readr)
options(tibble.width = Inf)
```

task 2:Reading the data of order.

```
order <- read_delim('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/order.csv', deli
```

```
## Rows: 300 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): order_date
## dbl (2): id, amount
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
order
```

```
## # A tibble: 300 x 3
##       id order_date amount
##    <dbl> <chr>       <dbl>
##  1   368 7/2/2016     365.
##  2   286 11/2/2016   2064.
##  3    28 2/22/2017    432.
##  4   309 3/5/2017     480.
##  5     2 12/28/2016   235.
##  6    31 12/30/2016  2745.
##  7   179 12/21/2016  2358.
##  8   484 11/24/2016  1031.
##  9   115 9/9/2016    1218.
## 10   340 5/6/2017    1184.
## # i 290 more rows
```

task 3::Reading the data of customer.

```
customer <- read_delim('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/customer.csv'
```

```
## Rows: 91 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (2): first_name, city
## dbl (1): id
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
customer
```

```
## # A tibble: 91 x 3
##       id first_name city
##    <dbl> <chr>      <chr>
##  1     1 Elbertine  California
##  2     2 Marcella   Colorado
##  3     3 Daria      Florida
##  4     4 Sherilyn   Distric...
##  5     5 Ketty      Texas
##  6     6 Jethro     California
##  7     7 Jeremiah   California
##  8     8 Constancia Texas
##  9     9 Muire      Idaho
## 10    10 Abigail    Texas
## # i 81 more rows
```

task 4:Joining data by id using inner join.

```
inner_join(customer, order, by = "id")
```

```
## # A tibble: 55 x 5
##       id first_name city       order_date amount
##    <dbl> <chr>      <chr>      <chr>       <dbl>
##  1     2 Marcella   Colorado   12/28/2016   235.
##  2     2 Marcella   Colorado   8/31/2016   1150.
##  3     5 Ketty      Texas      1/17/2017    346.
##  4     6 Jethro     California 1/27/2017   2317.
##  5     7 Jeremiah   California 6/21/2016    136.
##  6     7 Jeremiah   California 2/13/2017   1407.
##  7     7 Jeremiah   California 7/8/2016    1914.
##  8     8 Constancia Texas      11/5/2016   2461.
##  9     8 Constancia Texas      5/19/2017   2714.
## 10     9 Muire      Idaho      12/28/2016   187.
## # i 45 more rows
```

task 5:To get data for all those customers and their orders irrespective of whether a customer has placed orders or not let us join the order data with the customer data using left_join.

```
left_join(customer, order, by = "id")
```

```
## # A tibble: 104 x 5
##       id first_name city       order_date amount
##    <dbl> <chr>      <chr>      <chr>       <dbl>
##  1     1 Elbertine  California <NA>          NA
##  2     2 Marcella   Colorado   12/28/2016   235.
##  3     2 Marcella   Colorado   8/31/2016   1150.
##  4     3 Daria      Florida    <NA>          NA
##  5     4 Sherilyn   Distric... <NA>          NA
##  6     5 Ketty      Texas      1/17/2017    346.
##  7     6 Jethro     California 1/27/2017   2317.
##  8     7 Jeremiah   California 6/21/2016    136.
##  9     7 Jeremiah   California 2/13/2017   1407.
## 10     7 Jeremiah   California 7/8/2016    1914.
## # i 94 more rows
```

task 6:To get customer data for all orders, let us join the order data with the customer data using right_join.

```
right_join(customer, order, by = "id")
```

```
## # A tibble: 300 x 5
##       id first_name city       order_date amount
##    <dbl> <chr>      <chr>      <chr>       <dbl>
##  1     2 Marcella   Colorado   12/28/2016   235.
##  2     2 Marcella   Colorado   8/31/2016   1150.
##  3     5 Ketty      Texas      1/17/2017    346.
##  4     6 Jethro     California 1/27/2017   2317.
##  5     7 Jeremiah   California 6/21/2016    136.
##  6     7 Jeremiah   California 2/13/2017   1407.
##  7     7 Jeremiah   California 7/8/2016    1914.
##  8     8 Constancia Texas      11/5/2016   2461.
##  9     8 Constancia Texas      5/19/2017   2714.
## 10     9 Muire      Idaho      12/28/2016   187.
## # i 290 more rows
```

task 7:To get customer data for all orders where customer data exists, let us join the order data with the customer data using semi_join

```
semi_join(customer, order, by = "id")
```

```
## # A tibble: 42 x 3
##       id first_name city
##    <dbl> <chr>      <chr>
##  1     2 Marcella   Colorado
##  2     5 Ketty      Texas
##  3     6 Jethro     California
##  4     7 Jeremiah   California
##  5     8 Constancia Texas
##  6     9 Muire      Idaho
##  7    15 Valentijn  California
##  8    16 Monique    Missouri
##  9    20 Colette    Texas
## 10    28 Avrit      Texas
## # i 32 more rows
```

task 8:To get details of customers who have not placed orders, let us join the order data with the customer data using anti_join.

```
anti_join(customer, order, by = "id")
```

```
## # A tibble: 49 x 3
##       id first_name city
##    <dbl> <chr>      <chr>
##  1     1 Elbertine  California
##  2     3 Daria      Florida
##  3     4 Sherilyn   Distric...
##  4    10 Abigail    Texas
##  5    11 Wynne      Georgia
##  6    12 Pietra     Minnesota
##  7    13 Bram       Iowa
##  8    14 Rees       New York
##  9    17 Orazio     Louisiana
## 10    18 Mason      Texas
## # i 39 more rows
```

task 9:To get details of all customers and all orders, let us join the order data with the customer data using full_join.

```
full_join(customer, order, by = "id")
```

```
## # A tibble: 349 x 5
##       id first_name city       order_date amount
##    <dbl> <chr>      <chr>      <chr>       <dbl>
##  1     1 Elbertine  California <NA>          NA
##  2     2 Marcella   Colorado   12/28/2016   235.
##  3     2 Marcella   Colorado   8/31/2016   1150.
##  4     3 Daria      Florida    <NA>          NA
```

```
##  5     4 Sherilyn   Distric... <NA>          NA
##  6     5 Ketty      Texas      1/17/2017    346.
##  7     6 Jethro     California 1/27/2017   2317.
##  8     7 Jeremiah   California 6/21/2016    136.
##  9     7 Jeremiah   California 2/13/2017   1407.
## 10     7 Jeremiah   California 7/8/2016    1914.
## # i 339 more rows
```

dplyr Helpers

task 1:Loading the data.

```
ecom <-
  read_csv('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/web.csv',
    col_types = cols_only(device = col_factor(levels = c("laptop", "tablet", "mobile")),
      referrer = col_factor(levels = c("bing", "direct", "social", "yahoo", "google")),
      purchase = col_logical(), bouncers = col_logical(), duration = col_double(),
      n_visit = col_double(), n_pages = col_double()
    )
  )

ecom
```

```
## # A tibble: 1,000 x 7
##    referrer device bouncers n_visit n_pages duration purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
##  1 google   laptop TRUE          10       1      693 FALSE
##  2 yahoo    tablet TRUE           9       1      459 FALSE
##  3 direct   laptop TRUE           0       1      996 FALSE
##  4 bing     tablet FALSE          3      18      468 TRUE
##  5 yahoo    mobile TRUE           9       1      955 FALSE
##  6 yahoo    laptop FALSE          5       5      135 FALSE
##  7 yahoo    mobile TRUE          10       1       75 FALSE
##  8 direct   mobile TRUE          10       1      908 FALSE
##  9 bing     mobile FALSE          3      19      209 FALSE
## 10 google   mobile TRUE           6       1      208 FALSE
## # i 990 more rows
```

Data Sanitization

task 2:Using distinct to examine the values in the referrer column.

```
distinct(ecom, referrer)
```

```
## # A tibble: 5 x 1
##   referrer
##   <fct>
## 1 google
## 2 yahoo
## 3 direct
## 4 bing
## 5 social
```

task 3:Using distinct to examine the values in the device column.

```
distinct(ecom, device)
```

```
## # A tibble: 3 x 1
##   device
##   <fct>
## 1 laptop
## 2 tablet
## 3 mobile
```

Rename Columns

task 1:Renaming duration columns to time_on_site.

```
rename(ecom, time_on_site = duration)
```

```
## # A tibble: 1,000 x 7
##    referrer device bouncers n_visit n_pages time_on_site purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>        <dbl> <lgl>
##  1 google   laptop TRUE          10       1          693 FALSE
##  2 yahoo    tablet TRUE           9       1          459 FALSE
##  3 direct   laptop TRUE           0       1          996 FALSE
##  4 bing     tablet FALSE          3      18          468 TRUE
##  5 yahoo    mobile TRUE           9       1          955 FALSE
##  6 yahoo    laptop FALSE          5       5          135 FALSE
##  7 yahoo    mobile TRUE          10       1           75 FALSE
##  8 direct   mobile TRUE          10       1          908 FALSE
##  9 bing     mobile FALSE          3      19          209 FALSE
## 10 google   mobile TRUE           6       1          208 FALSE
## # i 990 more rows
```

Data Tabulation

task 1:Coung how many time a value is repeated in referrer.

```
ecom %>%
  group_by(referrer) %>%
  tally()
```

```
## # A tibble: 5 x 2
##   referrer     n
##   <fct>    <int>
## 1 bing       194
## 2 direct     191
## 3 social     200
## 4 yahoo      207
## 5 google     208
```

task 2:Knowing the number of bouncers driven by the different sources of traffic.

```
ecom %>%
  group_by(referrer, bouncers) %>%
  tally()
```

```
## # A tibble: 10 x 3
## # Groups:   referrer [5]
##    referrer bouncers     n
##    <fct>    <lgl>    <int>
##  1 bing     FALSE      104
##  2 bing     TRUE        90
##  3 direct   FALSE       98
##  4 direct   TRUE        93
##  5 social   FALSE       93
##  6 social   TRUE       107
##  7 yahoo    FALSE      110
##  8 yahoo    TRUE        97
##  9 google   FALSE      101
## 10 google   TRUE       107
```

task 3:looking how many conversions happen across different devices.

```
ecom %>%
  group_by(device, purchase) %>%
  tally() %>%
  filter(purchase)
```

```
## # A tibble: 3 x 3
## # Groups:   device [3]
##   device purchase     n
##   <fct>  <lgl>    <int>
## 1 laptop TRUE        31
## 2 tablet TRUE        36
## 3 mobile TRUE        36
```

task 4:Extracting the above information is by using count.

```
ecom %>%
  count(referrer, purchase) %>%
  filter(purchase)
```

```
## # A tibble: 5 x 3
##   referrer purchase     n
##   <fct>    <lgl>    <int>
## 1 bing     TRUE        17
## 2 direct   TRUE        25
## 3 social   TRUE        20
## 4 yahoo    TRUE        22
## 5 google   TRUE        19
```

Sampling Data

task 1:sampling a specific number of observations.

```
sample_n(ecom, 700)
```

```
## # A tibble: 700 x 7
##    referrer device bouncers n_visit n_pages duration purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
##  1 bing     laptop FALSE          8       6      180 FALSE
##  2 social   laptop TRUE           1       1       35 FALSE
##  3 bing     tablet TRUE           1       1      406 FALSE
##  4 google   laptop TRUE           9       1      869 FALSE
##  5 google   mobile FALSE          9       6       96 FALSE
##  6 bing     mobile TRUE           3       1      119 FALSE
##  7 google   tablet FALSE          1      11      220 FALSE
##  8 bing     mobile FALSE         10      14      140 FALSE
##  9 google   tablet TRUE           8       1      249 FALSE
## 10 bing     mobile TRUE           6       1      604 FALSE
## # i 690 more rows
```

task 2:groups the 'ecom' dataframe by the 'referrer' column and then randomly samples 100 rows from each group.

```
ecom %>%
  group_by(referrer) %>%
  sample_n(100)
```

```
## # A tibble: 500 x 7
## # Groups:   referrer [5]
##    referrer device bouncers n_visit n_pages duration purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
##  1 bing     laptop FALSE          5      18      234 FALSE
##  2 bing     mobile TRUE           6       1      966 FALSE
##  3 bing     tablet TRUE           0       1      845 FALSE
##  4 bing     laptop TRUE          10       1      210 FALSE
##  5 bing     laptop TRUE          10       1      947 FALSE
##  6 bing     mobile FALSE          3      18      288 TRUE
##  7 bing     tablet TRUE           7       1      423 FALSE
##  8 bing     tablet FALSE          4      15      435 TRUE
##  9 bing     tablet TRUE           9       1      281 FALSE
## 10 bing     laptop TRUE           1       1      169 FALSE
## # i 490 more rows
```

task 3:sample_frac() allows a specific percentage of observations.

```
sample_frac(ecom, size = 0.7)
```

```
## # A tibble: 700 x 7
##    referrer device bouncers n_visit n_pages duration purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
##  1 google   tablet FALSE          1       5       75 FALSE
```

```
##  2 bing     mobile FALSE          3      19      209 FALSE
##  3 yahoo    tablet FALSE          7      10      240 FALSE
##  4 social   mobile FALSE          2       2       36 FALSE
##  5 social   tablet FALSE          8       6      156 FALSE
##  6 yahoo    laptop TRUE           8       1      924 FALSE
##  7 bing     tablet TRUE           3       1      134 FALSE
##  8 bing     laptop FALSE         10      16      304 FALSE
##  9 social   tablet FALSE          6       6      108 FALSE
## 10 direct   mobile TRUE           1       1      860 FALSE
## # i 690 more rows
```

Sample Data

task 1:Extracting the t column from ecom using column name.

```
ecom_mini <- sample_n(ecom, size = 10)
pull(ecom_mini, device)
```

```
##  [1] mobile mobile mobile tablet laptop tablet laptop tablet laptop mobile
## Levels: laptop tablet mobile
```

task 2:Extracting the first column from ecom using column position instead of name.

```
pull(ecom_mini, 1)
```

```
##  [1] bing   bing   social yahoo  social google google social yahoo  google
## Levels: bing direct social yahoo google
```

task 3:Extracting data from the last column.

```
pull(ecom_mini, -1)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

task 4:Extracting data starting from the 5th row and upto the 15th row.

```
slice(ecom, 5:15)
```

```
## # A tibble: 11 x 7
##    referrer device bouncers n_visit n_pages duration purchase
##    <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
##  1 yahoo    mobile TRUE           9       1      955 FALSE
##  2 yahoo    laptop FALSE          5       5      135 FALSE
##  3 yahoo    mobile TRUE          10       1       75 FALSE
##  4 direct   mobile TRUE          10       1      908 FALSE
##  5 bing     mobile FALSE          3      19      209 FALSE
##  6 google   mobile TRUE           6       1      208 FALSE
##  7 direct   laptop TRUE           9       1      738 FALSE
##  8 direct   tablet FALSE          6      12      132 FALSE
##  9 direct   mobile FALSE          9      14      406 TRUE
## 10 yahoo    tablet FALSE          5       8       80 FALSE
## 11 yahoo    mobile FALSE          7       1       19 FALSE
```

task 5:Useing n() inside slice() to extract the last row.

```
slice(ecom, n())
```

```
## # A tibble: 1 x 7
##   referrer device bouncers n_visit n_pages duration purchase
##   <fct>    <fct>  <lgl>      <dbl>   <dbl>    <dbl> <lgl>
## 1 google   mobile TRUE           9       1      269 FALSE
```

Case Between

task 1:Checking how many visits browsed pages between 5 and 15.

```
ecom_sample <- sample_n(ecom, 30)

ecom_sample %>%
  pull(n_pages) %>%
  between(5, 15)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [13]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [25] FALSE FALSE  TRUE  TRUE  TRUE FALSE
```

Case When

task 1:creating a new column repeat_visit from n_visit (the number of previous visits).

```
ecom %>%
  mutate(
    repeat_visit = case_when(
      n_visit > 0 ~ TRUE,
      TRUE ~ FALSE
    )
  ) %>%
  select(n_visit, repeat_visit)
```

```
## # A tibble: 1,000 x 2
##    n_visit repeat_visit
##      <dbl> <lgl>
##  1      10 TRUE
##  2       9 TRUE
##  3       0 FALSE
##  4       3 TRUE
##  5       9 TRUE
##  6       5 TRUE
##  7      10 TRUE
##  8      10 TRUE
##  9       3 TRUE
## 10       6 TRUE
## # i 990 more rows
```

Pipe Operator

task 1:Using the following R packages.

```
library(magrittr)
library(readr)
library(dplyr)
library(stringr)
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:magrittr':
##
##     set_names
```

task 2: Loading the data .

```
ecom <-
  read_csv('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/web.csv',
    col_types = cols_only(
      referrer = col_factor(levels = c("bing", "direct", "social", "yahoo", "google")),
      n_pages = col_double(), duration = col_double(), purchase = col_logical()
    )
  )

ecom
```

```
## # A tibble: 1,000 x 4
##    referrer n_pages duration purchase
##    <fct>      <dbl>    <dbl> <lgl>
##  1 google         1      693 FALSE
##  2 yahoo          1      459 FALSE
##  3 direct         1      996 FALSE
##  4 bing          18      468 TRUE
##  5 yahoo          1      955 FALSE
##  6 yahoo          5      135 FALSE
##  7 yahoo          1       75 FALSE
##  8 direct         1      908 FALSE
##  9 bing          19      209 FALSE
## 10 google         1      208 FALSE
## # i 990 more rows
```

task 3:Taking only 1st 10 rows.

```
ecom_mini <- sample_n(ecom, size = 10)
ecom_mini
```

```
## # A tibble: 10 x 4
##    referrer n_pages duration purchase
```

```
##    <fct>      <dbl>    <dbl> <lgl>
##  1 social        17      459 FALSE
##  2 bing           9      261 FALSE
##  3 social        17      306 TRUE
##  4 google         1      369 FALSE
##  5 bing           1      490 FALSE
##  6 google        13      338 FALSE
##  7 bing           1      845 FALSE
##  8 social        20      420 TRUE
##  9 social         9      225 FALSE
## 10 direct        13      390 TRUE
```

task 4:Loading 1st 10 data.

```
head(ecom, 10)
```

```
## # A tibble: 10 x 4
##    referrer n_pages duration purchase
##    <fct>      <dbl>    <dbl> <lgl>
##  1 google         1      693 FALSE
##  2 yahoo          1      459 FALSE
##  3 direct         1      996 FALSE
##  4 bing          18      468 TRUE
##  5 yahoo          1      955 FALSE
##  6 yahoo          5      135 FALSE
##  7 yahoo          1       75 FALSE
##  8 direct         1      908 FALSE
##  9 bing          19      209 FALSE
## 10 google         1      208 FALSE
```

task 5:Loading 1st 10 rows of data using %>%.

```
ecom %>% head(10)
```

```
## # A tibble: 10 x 4
##    referrer n_pages duration purchase
##    <fct>      <dbl>    <dbl> <lgl>
##  1 google         1      693 FALSE
##  2 yahoo          1      459 FALSE
##  3 direct         1      996 FALSE
##  4 bing          18      468 TRUE
##  5 yahoo          1      955 FALSE
##  6 yahoo          5      135 FALSE
##  7 yahoo          1       75 FALSE
##  8 direct         1      908 FALSE
##  9 bing          19      209 FALSE
## 10 google         1      208 FALSE
```

Square Root

task 1: square root of n_pages column from the data set.

```r
y <- sqrt(ecom_mini$n_pages)
```

task 2:select n_pages variable and assign it to y and compute square root of y and assign it to y.

```r
y <-
    ecom_mini %$%
    n_pages

y %<>% sqrt
```

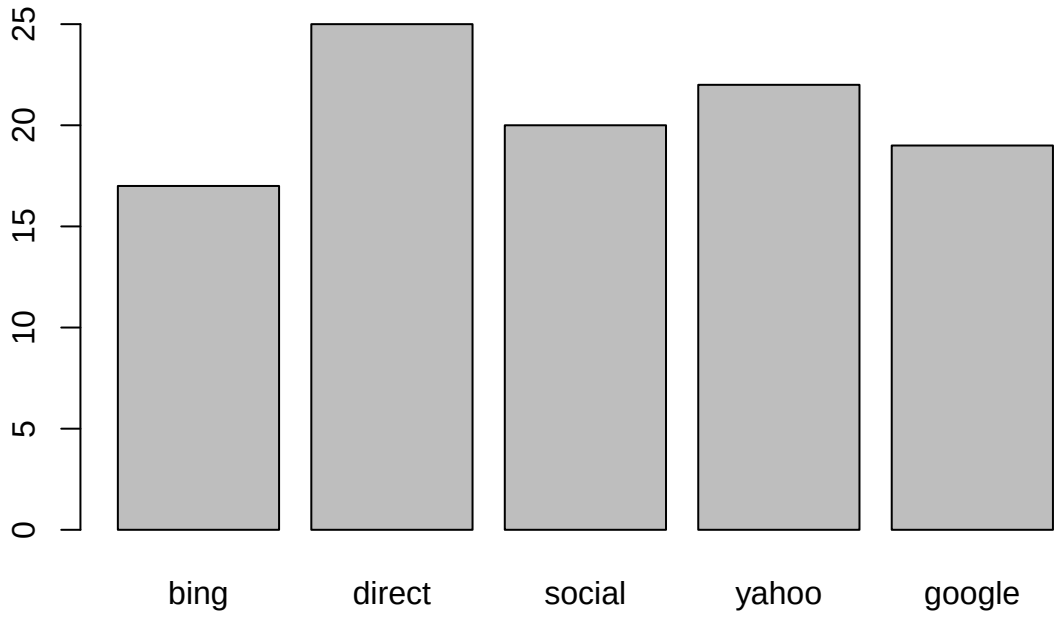
task 3:Another way to compute the square root of y.

```r
y <-
  ecom_mini %$%
  n_pages %>%
  sqrt()
```

Visualization

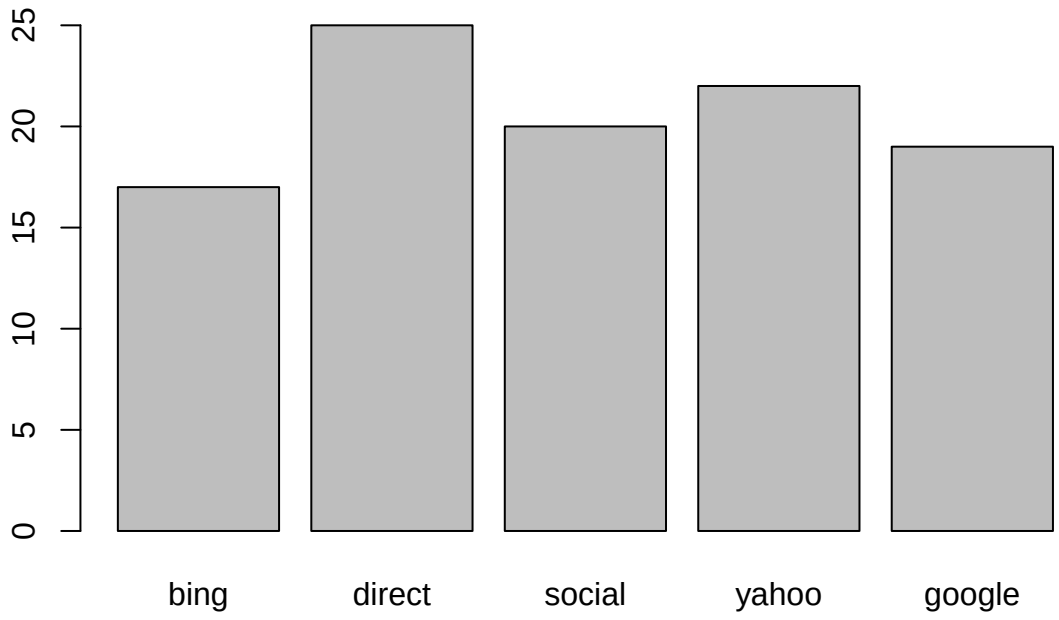
task 1:creating bar plot of referrer.

```r
barplot(table(subset(ecom, purchase)$referrer))
```



task 2:creating bar plot using pipe operator.

```
ecom %>%
  subset(purchase) %>%
  extract('referrer') %>%
  table() %>%
  barplot()
```



Correlation

task 1:Calculating correlation.

```
ecom1 <- subset(ecom, purchase)
cor(ecom1$n_pages, ecom1$duration)
```

```
## [1] 0.4290905
```

task 2:Calculating correlation with pipe.

```
ecom %>%
  subset(purchase) %$%
  cor(n_pages, duration)
```

```
## [1] 0.4290905
```

Regression

task 1:Calculating Regression.

```
summary(lm(duration ~ n_pages, data = ecom))
```

```
##
## Call:
## lm(formula = duration ~ n_pages, data = ecom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -386.45 -213.03  -38.93  179.31  602.55
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  404.803     11.323  35.750  < 2e-16 ***
## n_pages       -8.355      1.296  -6.449 1.76e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 263.3 on 998 degrees of freedom
## Multiple R-squared:   0.04,  Adjusted R-squared:  0.03904
## F-statistic: 41.58 on 1 and 998 DF,  p-value: 1.756e-10
```

task 2:Calculating Regression using pipe operator.

```
ecom %$%
  lm(duration ~ n_pages) %>%
  summary()
```

```
##
## Call:
## lm(formula = duration ~ n_pages)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -386.45 -213.03  -38.93  179.31  602.55
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  404.803     11.323  35.750  < 2e-16 ***
## n_pages       -8.355      1.296  -6.449 1.76e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 263.3 on 998 degrees of freedom
## Multiple R-squared:   0.04,  Adjusted R-squared:  0.03904
## F-statistic: 41.58 on 1 and 998 DF,  p-value: 1.756e-10
```

String Manipulation

task 1:Extracting the first name (jovial) from the below email id and convert it to upper case.

```
email <- 'jovialcann@anymail.com'

# without pipe
str_to_upper(str_sub(str_split(email, '@')[[1]][1], start = 1, end = 6))
```

```
## [1] "JOVIAL"
```

```
# with pipe
email %>%
  str_split(pattern = '@') %>%
  extract2(1) %>%
  extract(1) %>%
  str_sub(start = 1, end = 6) %>%
  str_to_upper()
```

```
## [1] "JOVIAL"
```

task 2:Using another method that uses map_chr() from the purrr package.

```
email %>%
  str_split(pattern = '@') %>%
  map_chr(1) %>%
  str_sub(start = 1, end = 6) %>%
  str_to_upper()
```

```
## [1] "JOVIAL"
```

Data Extraction

task 1:Displaying n_pages.

```
ecom_mini['n_pages']
```

```
## # A tibble: 10 x 1
##    n_pages
##      <dbl>
##  1      17
##  2       9
##  3      17
##  4       1
##  5       1
##  6      13
##  7       1
##  8      20
##  9       9
## 10      13
```

task 2:Extracting columns using their name.

```
extract(ecom_mini, 'n_pages')
```

```
## # A tibble: 10 x 1
##    n_pages
##      <dbl>
##  1      17
##  2       9
##  3      17
##  4       1
##  5       1
##  6      13
##  7       1
##  8      20
##  9       9
## 10      13
```

task 3:Displaying using their index position.

```
ecom_mini[2]
```

```
## # A tibble: 10 x 1
##    n_pages
##      <dbl>
##  1      17
##  2       9
##  3      17
##  4       1
##  5       1
##  6      13
##  7       1
##  8      20
##  9       9
## 10      13
```

task 4:Extracting columns using their index position.

```
extract(ecom_mini, 2)
```

```
## # A tibble: 10 x 1
##    n_pages
##      <dbl>
##  1      17
##  2       9
##  3      17
##  4       1
##  5       1
##  6      13
##  7       1
##  8      20
##  9       9
## 10      13
```

task 5:$ will also return a atomic vector.

```
ecom_mini$n_pages
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

task 6:Using use_series() in place of $.

```
use_series(ecom_mini, 'n_pages')
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

task 7:converting ecom_mini into a list using as.list() .

```
ecom_list <- as.list(ecom_mini)
```

task 8:Extracting elements of a list using [[ ]].

```
ecom_list[['n_pages']]
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

task 9:Extracting elements of a list using extract2().

```
extract2(ecom_list, 'n_pages')
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

task 10:Extracting elements using index position.

```
ecom_list[[1]]
```

```
##  [1] social bing   social google bing   google bing   social social direct
## Levels: bing direct social yahoo google
```

```
extract2(ecom_list, 1)
```

```
##  [1] social bing   social google bing   google bing   social social direct
## Levels: bing direct social yahoo google
```

task 11:Extracting the elements of a list using use_series() .

```
ecom_list$n_pages
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

```r
use_series(ecom_list, n_pages)
```

```
##  [1] 17  9 17  1  1 13  1 20  9 13
```

Arithmetic Operations

task 1:Adding the data using different methods.

```r
1:10 + 1
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11
```

```r
add(1:10, 1)
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11
```

```r
`+`(1:10, 1)
```

```
##  [1]  2  3  4  5  6  7  8  9 10 11
```

task 2:Multiplying the data using different methods.

```r
1:10 * 3
```

```
##  [1]  3  6  9 12 15 18 21 24 27 30
```

```r
multiply_by(1:10, 3)
```

```
##  [1]  3  6  9 12 15 18 21 24 27 30
```

```r
`*`(1:10, 3)
```

```
##  [1]  3  6  9 12 15 18 21 24 27 30
```

task 3:Dividing the data using different methods.

```r
1:10 / 2
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```r
divide_by(1:10, 2)
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```r
`/`(1:10, 2)
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

task 4:power the data using different methods.

```r
1:10 ^ 2
```

```
##   [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
```

```r
raise_to_power(1:10, 2)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```r
`^`(1:10, 2)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Logical Operators

task 1:Using greater than in different methods.

```r
1:10 > 5
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
is_greater_than(1:10, 5)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
`>`(1:10, 5)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

task 3:Using weakly greater than in different methods.

```r
1:10 >= 5
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
is_weakly_greater_than(1:10, 5)
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
`>=`(1:10, 5)
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

tibbles

task 1:Loading the library.

```r
library(tibble)
library(dplyr)
```

task 2:Creating tibbles

```r
tibble(x = letters,
       y = 1:26,
       z = sample(100, 26))
```

```
## # A tibble: 26 x 3
##    x         y     z
##    <chr> <int> <int>
##  1 a         1    45
##  2 b         2    66
##  3 c         3    50
##  4 d         4     9
##  5 e         5    24
##  6 f         6    73
##  7 g         7    41
##  8 h         8    33
##  9 i         9    46
## 10 j        10   100
## # i 16 more rows
```

tibble features

task 1:never changes input's types.

```r
tibble(x = letters,
       y = 1:26,
       z = sample(100, 26))
```

```
## # A tibble: 26 x 3
##    x         y     z
##    <chr> <int> <int>
##  1 a         1    36
```

```
##  2 b         2    83
##  3 c         3     1
##  4 d         4     6
##  5 e         5    30
##  6 f         6    59
##  7 g         7    98
##  8 h         8    93
##  9 i         9    13
## 10 j        10    84
## # i 16 more rows
```

task 2:never adjusts variable names.

```
names(data.frame(`order value` = 10))
```

```
## [1] "order.value"
```

```
names(tibble(`order value` = 10))
```

```
## [1] "order value"
```

task 3:never prints all rows.

```
x <- 1:100
y <- letters[1]
z <- sample(c(TRUE, FALSE), 100, replace = TRUE)
tibble(x, y, z)
```

```
## # A tibble: 100 x 3
##        x y     z
##    <int> <chr> <lgl>
##  1     1 a     TRUE
##  2     2 a     TRUE
##  3     3 a     TRUE
##  4     4 a     FALSE
##  5     5 a     TRUE
##  6     6 a     TRUE
##  7     7 a     TRUE
##  8     8 a     TRUE
##  9     9 a     TRUE
## 10    10 a     TRUE
## # i 90 more rows
```

task 4:Never recycles vector of length greater than 1

```
x <- 1:100
y <- rep(letters, length.out = 100)
z <- sample(c(TRUE, FALSE), 100, replace = TRUE)
tibble(x, y, z)
```

```
## # A tibble: 100 x 3
##        x y     z
##    <int> <chr> <lgl>
##  1     1 a     TRUE
##  2     2 b     TRUE
##  3     3 c     TRUE
##  4     4 d     FALSE
##  5     5 e     TRUE
##  6     6 f     TRUE
##  7     7 g     TRUE
##  8     8 h     TRUE
##  9     9 i     TRUE
## 10    10 j     TRUE
## # i 90 more rows
```

task 5:Testing if an object is a tibble using is_tibble().

```
is_tibble(mtcars)
```

```
## [1] FALSE
```

```
is_tibble(as_tibble(mtcars))
```

```
## [1] TRUE
```

task 7:Creating tibbles is using tribble()

```
tribble(
  ~x, ~y, ~z,
  #--/--/----
  1, TRUE, 'a',
  2, FALSE, 'b'
)
```

```
## # A tibble: 2 x 3
##       x y     z
##   <dbl> <lgl> <chr>
## 1     1 TRUE  a
## 2     2 FALSE b
```

task 8:Names of the columns in tibbles

```
tibble(
  ` ` = 'space',
  `2` = 'integer',
  `:)` = 'smiley'
)
```

```
## # A tibble: 1 x 3
##   ‘ ‘   ‘2‘     ‘:)‘
##   <chr> <chr>   <chr>
## 1 space integer smiley
```

task 9:Adding data related to Safari browser to the web traffic data using add_row().

```
browsers <- enframe(c(chrome = 40, firefox = 20, edge = 30))
browsers
```

```
## # A tibble: 3 x 2
##   name    value
##   <chr>   <dbl>
## 1 chrome     40
## 2 firefox    20
## 3 edge       30
```

```
add_row(browsers, name = 'safari', value = 10)
```

```
## # A tibble: 4 x 2
##   name    value
##   <chr>   <dbl>
## 1 chrome     40
## 2 firefox    20
## 3 edge       30
## 4 safari     10
```

task 10:Changing the safari from last to the second.

```
add_row(browsers, name = 'safari', value = 10, .before = 2)
```

```
## # A tibble: 4 x 2
##   name    value
##   <chr>   <dbl>
## 1 chrome     40
## 2 safari     10
## 3 firefox    20
## 4 edge       30
```

task 11:Adding the new columns using add_columns()

```
browsers <- enframe(c(chrome = 40, firefox = 20, edge = 30, safari = 10))
add_column(browsers, visits = c(4000, 2000, 3000, 1000))
```

```
## # A tibble: 4 x 3
##   name    value visits
##   <chr>   <dbl>  <dbl>
## 1 chrome     40   4000
## 2 firefox    20   2000
## 3 edge       30   3000
## 4 safari     10   1000
```

task 12:Checking whether a data set has rownames, using has_rownames().

```
has_rownames(mtcars)
```

```
## [1] TRUE
```

task 13:Removing Rownames.

```
remove_rownames(mtcars)
```

```
##     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## 7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## 8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## 9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## 10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## 11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## 12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## 13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## 14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## 15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## 16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## 17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## 18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## 19 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## 20 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## 21 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## 22 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## 23 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## 24 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## 25 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## 26 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## 27 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## 28 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## 29 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## 30 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## 31 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## 32 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

task 14:Rownames to Column.

```
head(rownames_to_column(mtcars))
```

```
##              rowname  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1          Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2      Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3         Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4     Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5  Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6            Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

task 15:Converting the first column in the data set to rownames, use column_to_rownames():

```
mtcars_tbl <- rownames_to_column(mtcars)
column_to_rownames(mtcars_tbl)
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona      21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger   15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin        15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28         13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird   19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9          27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa       30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

task 16:Using glimpse() to get an overview of the data.

```
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
```

```
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

task 17:Checking if a tibble has a specific column using has_name().

```
has_name(mtcars, 'cyl')
```

```
## [1] TRUE
```

```
has_name(mtcars, 'gears')
```

```
## [1] FALSE
```

Hacking Strings

task 1:Loading the library.

```
library(stringr)
library(tibble)
library(magrittr)
library(purrr)
library(dplyr)
library(readr)
```

task 2:Importing the data.

```
mockstring <- read_csv('https://raw.githubusercontent.com/rsquaredacademy/datasets/master/mock_strings.
```

```
## Rows: 1000 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): image_url, domain, imageurl, email, filename, phone, address, url,...
## dbl  (1): id
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
mockstring
```

```
## # A tibble: 1,000 x 12
##       id
##    <dbl>
##  1     1
##  2     2
##  3     3
##  4     4
##  5     5
##  6     6
```

```
##  7     7
##  8     8
##  9     9
## 10    10
##    image_url
##    <chr>
##  1 https://robohash.org/providentassumendaexplicabo.jpg?size=50x50&set=set1
##  2 https://robohash.org/etillumvoluptate.jpg?size=50x50&set=set1
##  3 https://robohash.org/nonoptiovoluptatibus.jpg?size=50x50&set=set1
##  4 https://robohash.org/voluptatumauthic.jpg?size=50x50&set=set1
##  5 https://robohash.org/placeaterrorqui.jpg?size=50x50&set=set1
##  6 https://robohash.org/temporeutea.jpg?size=50x50&set=set1
##  7 https://robohash.org/maximesaepequi.bmp?size=50x50&set=set1
##  8 https://robohash.org/nemoautesse.png?size=50x50&set=set1
##  9 https://robohash.org/odiorerumaut.png?size=50x50&set=set1
## 10 https://robohash.org/omnismolestiaearchitecto.png?size=50x50&set=set1
##    domain           imageurl
##    <chr>            <chr>
##  1 addtoany.com     http://dummyimage.com/130x183.jpg/dddddd/000000
##  2 gmpg.org         http://dummyimage.com/106x217.bmp/dddddd/000000
##  3 samsung.com      http://dummyimage.com/146x127.bmp/cc0000/ffffff
##  4 spotify.com      http://dummyimage.com/181x194.png/5fa2dd/ffffff
##  5 wunderground.com http://dummyimage.com/220x123.jpg/ff4444/ffffff
##  6 alexa.com        http://dummyimage.com/118x176.bmp/dddddd/000000
##  7 google.it        http://dummyimage.com/185x202.jpg/ff4444/ffffff
##  8 ed.gov           http://dummyimage.com/223x163.jpg/ff4444/ffffff
##  9 jigsy.com        http://dummyimage.com/145x113.jpg/5fa2dd/ffffff
## 10 jugem.jp         http://dummyimage.com/238x214.png/cc0000/ffffff
##    email                         filename              phone
##    <chr>                         <chr>                 <chr>
##  1 mnewburn0@fastcompany.com     PedeMalesuada.xls     66-(777)902-6181
##  2 mdankersley1@digg.com         LobortisVel.mp3       351-(422)736-6807
##  3 hgirhard2@altervista.org      CongueDiamId.pdf      33-(371)684-5114
##  4 pmcmenamy3@sciencedirect.com EleifendQuam.avi       86-(410)823-6712
##  5 drisbrough4@bandcamp.com      PurusPhasellus.mp3    223-(518)814-6361
##  6 cphlippi5@surveymonkey.com    ElementumInHac.avi    420-(760)354-8671
##  7 kdodswell6@un.org             Mattis.doc            1-(712)615-2879
##  8 vhourihane7@ovh.net           PurusEu.tiff          62-(437)705-1118
##  9 rdike8@timesonline.co.uk      JustoEtiamPretium.xls 1-(683)965-1323
## 10 tdudbridge9@clickbank.net     Ante.tiff             30-(553)559-7448
##    address
##    <chr>
##  1 8 Anhalt Crossing
##  2 697 East Avenue
##  3 89 Dottie Circle
##  4 98135 Blue Bill Park Drive
##  5 7814 Pennsylvania Street
##  6 4897 Little Fleur Drive
##  7 53541 Morrow Center
##  8 4819 Hermina Parkway
##  9 68096 Monument Park
## 10 9595 Spaight Avenue
##    url
##    <chr>
```

```
##  1 https://engadget.com/nascetur/ridiculus/mus/vivamus/vestibulum.jsp?eu=est&ti~
##  2 http://delicious.com/phasellus/in/felis/donec.json?interdum=risus&mauris=dap~
##  3 https://w3.org/sed/augue/aliquam/erat/volutpat.json?dictumst=mi&morbi=sit&ve~
##  4 http://indiatimes.com/pede/lobortis/ligula/sit/amet.jpg?quam=nullam&sollicit~
##  5 https://tumblr.com/id/mauris/vulputate/elementum.png?tincidunt=maecenas&eget~
##  6 https://unblog.fr/est/quam/pharetra.jpg?amet=phasellus&erat=sit&nulla=amet&t~
##  7 http://vinaora.com/posuere.jpg?convallis=in&nulla=faucibus&neque=orci&libero~
##  8 https://globo.com/accumsan.png?elementum=eu&pellentesque=mi&quisque=nulla&po~
##  9 https://xing.com/elementum/eu/interdum/eu/tincidunt.html?sit=proin&amet=eu&s~
## 10 https://bigcartel.com/tortor/quis/turpis/sed/ante/vivamus.html?in=lorem&elei~
##    full_name          currency passwords
##    <chr>              <chr>    <chr>
##  1 Mufi Ruit          ¥34.37   VybPYpEXUjJh6nQk
##  2 Leese Furmagier    $67.37   mxET3n6dz42X8YUv
##  3 Blakelee Wilshire  €33,85   Z9f4WeNVQ28FwKML
##  4 Terencio McIllrick €42,89   Ndbm8nwCps6jUze3
##  5 Debee McErlaine    €13,19   U3Lj9xJw8NHzB5Sg
##  6 Fran Painten       ¥87.35   KEhVAC3QNvjWDFJ7
##  7 Frasco Bowich      $34.89   jydGPCW7fa2bZpU4
##  8 Car Ponten         ¥41.66   pytVHesNZjAL8WKc
##  9 Tades Checcucci    €70,80   Rsw4EQGk9tKTnzDp
## 10 Wilton Kemmey      €62,76   KvrNGQ7yL3pfsaZA
## # i 990 more rows
```

task 3:use a smaller data (i.e. 1st 10 rows)

```
mockdata <- slice(mockstring, 1:10)
mockdata
```

```
## # A tibble: 10 x 12
##       id
##    <dbl>
##  1     1
##  2     2
##  3     3
##  4     4
##  5     5
##  6     6
##  7     7
##  8     8
##  9     9
## 10    10
##    image_url
##    <chr>
##  1 https://robohash.org/providentassumendaexplicabo.jpg?size=50x50&set=set1
##  2 https://robohash.org/etillumvoluptate.jpg?size=50x50&set=set1
##  3 https://robohash.org/nonoptiovoluptatibus.jpg?size=50x50&set=set1
##  4 https://robohash.org/voluptatumauthic.jpg?size=50x50&set=set1
##  5 https://robohash.org/placeaterrorqui.jpg?size=50x50&set=set1
##  6 https://robohash.org/temporeutea.jpg?size=50x50&set=set1
##  7 https://robohash.org/maximesaepequi.bmp?size=50x50&set=set1
##  8 https://robohash.org/nemoautesse.png?size=50x50&set=set1
##  9 https://robohash.org/odiorerumaut.png?size=50x50&set=set1
## 10 https://robohash.org/omnismolestiaearchitecto.png?size=50x50&set=set1
```

```
##    domain           imageurl
##    <chr>            <chr>
##  1 addtoany.com     http://dummyimage.com/130x183.jpg/dddddd/000000
##  2 gmpg.org         http://dummyimage.com/106x217.bmp/dddddd/000000
##  3 samsung.com      http://dummyimage.com/146x127.bmp/cc0000/ffffff
##  4 spotify.com      http://dummyimage.com/181x194.png/5fa2dd/ffffff
##  5 wunderground.com http://dummyimage.com/220x123.jpg/ff4444/ffffff
##  6 alexa.com        http://dummyimage.com/118x176.bmp/dddddd/000000
##  7 google.it        http://dummyimage.com/185x202.jpg/ff4444/ffffff
##  8 ed.gov           http://dummyimage.com/223x163.jpg/ff4444/ffffff
##  9 jigsy.com        http://dummyimage.com/145x113.jpg/5fa2dd/ffffff
## 10 jugem.jp         http://dummyimage.com/238x214.png/cc0000/ffffff
##    email                        filename              phone
##    <chr>                        <chr>                 <chr>
##  1 mnewburn0@fastcompany.com    PedeMalesuada.xls     66-(777)902-6181
##  2 mdankersley1@digg.com        LobortisVel.mp3       351-(422)736-6807
##  3 hgirhard2@altervista.org     CongueDiamId.pdf      33-(371)684-5114
##  4 pmcmenamy3@sciencedirect.com EleifendQuam.avi      86-(410)823-6712
##  5 drisbrough4@bandcamp.com     PurusPhasellus.mp3    223-(518)814-6361
##  6 cphlippi5@surveymonkey.com   ElementumInHac.avi    420-(760)354-8671
##  7 kdodswell6@un.org            Mattis.doc            1-(712)615-2879
##  8 vhourihane7@ovh.net          PurusEu.tiff          62-(437)705-1118
##  9 rdike8@timesonline.co.uk     JustoEtiamPretium.xls 1-(683)965-1323
## 10 tdudbridge9@clickbank.net    Ante.tiff             30-(553)559-7448
##    address
##    <chr>
##  1 8 Anhalt Crossing
##  2 697 East Avenue
##  3 89 Dottie Circle
##  4 98135 Blue Bill Park Drive
##  5 7814 Pennsylvania Street
##  6 4897 Little Fleur Drive
##  7 53541 Morrow Center
##  8 4819 Hermina Parkway
##  9 68096 Monument Park
## 10 9595 Spaight Avenue
##    url
##    <chr>
##  1 https://engadget.com/nascetur/ridiculus/mus/vivamus/vestibulum.jsp?eu=est&ti~
##  2 http://delicious.com/phasellus/in/felis/donec.json?interdum=risus&mauris=dap~
##  3 https://w3.org/sed/augue/aliquam/erat/volutpat.json?dictumst=mi&morbi=sit&ve~
##  4 http://indiatimes.com/pede/lobortis/ligula/sit/amet.jpg?quam=nullam&sollicit~
##  5 https://tumblr.com/id/mauris/vulputate/elementum.png?tincidunt=maecenas&eget~
##  6 https://unblog.fr/est/quam/pharetra.jpg?amet=phasellus&erat=sit&nulla=amet&t~
##  7 http://vinaora.com/posuere.jpg?convallis=in&nulla=faucibus&neque=orci&libero~
##  8 https://globo.com/accumsan.png?elementum=eu&pellentesque=mi&quisque=nulla&po~
##  9 https://xing.com/elementum/eu/interdum/eu/tincidunt.html?sit=proin&amet=eu&s~
## 10 https://bigcartel.com/tortor/quis/turpis/sed/ante/vivamus.html?in=lorem&elei~
##    full_name          currency passwords
##    <chr>              <chr>    <chr>
##  1 Mufi Ruit          ¥34.37   VybPYpEXUjJh6nQk
##  2 Leese Furmagier    $67.37   mxET3n6dz42X8YUv
##  3 Blakelee Wilshire  €33,85   Z9f4WeNVQ28FwKML
##  4 Terencio McIllrick €42,89   Ndbm8nwCps6jUze3
```

```
##  5 Debee McErlaine    €13,19   U3Lj9xJw8NHzB5Sg
##  6 Fran Painten       ¥87.35   KEhVAC3QNvjWDFJ7
##  7 Frasco Bowich      $34.89   jydGPCW7fa2bZpU4
##  8 Car Ponten         ¥41.66   pytVHesNZjAL8WKc
##  9 Tades Checcucci    €70,80   Rsw4EQGk9tKTnzDp
## 10 Wilton Kemmey      €62,76   KvrNGQ7yL3pfsaZA
```

task 4:Using str_detect() to detect @ .

```
str_detect(mockdata$email, pattern = "@")
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

task 5:Using str_count() to count the number of times @ appears in the email ids.

```
str_count(mockdata$email, pattern = "@")
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

task 6:Using str_c() to concatenate strings, adding the string email id: before each email id in the data set.

```
str_c("email id:", mockdata$email)
```

```
##  [1] "email id:mnewburn0@fastcompany.com"
##  [2] "email id:mdankersley1@digg.com"
##  [3] "email id:hgirhard2@altervista.org"
##  [4] "email id:pmcmenamy3@sciencedirect.com"
##  [5] "email id:drisbrough4@bandcamp.com"
##  [6] "email id:cphlippi5@surveymonkey.com"
##  [7] "email id:kdodswell6@un.org"
##  [8] "email id:vhourihane7@ovh.net"
##  [9] "email id:rdike8@timesonline.co.uk"
## [10] "email id:tdudbridge9@clickbank.net"
```

task 7:spliting domain name and extension from the domain column.

```
str_split(mockdata$domain, pattern = "\\.")
```

```
## [[1]]
## [1] "addtoany" "com"
##
## [[2]]
## [1] "gmpg" "org"
##
## [[3]]
## [1] "samsung" "com"
##
## [[4]]
## [1] "spotify" "com"
##
```

```
## [[5]]
## [1] "wunderground" "com"
##
## [[6]]
## [1] "alexa" "com"
##
## [[7]]
## [1] "google" "it"
##
## [[8]]
## [1] "ed"  "gov"
##
## [[9]]
## [1] "jigsy" "com"
##
## [[10]]
## [1] "jugem" "jp"
```

task 8:It truncates each email address in the 'email' column of the 'mockdata' dataframe to a maximum width of 10 characters.

```
str_trunc(mockdata$email, width = 10)
```

```
##  [1] "mnewbur..." "mdanker..." "hgirhar..." "pmcmena..." "drisbro..."
##  [6] "cphlipp..." "kdodswe..." "vhourih..." "rdike8@..." "tdudbri..."
```

```
str_trunc(mockdata$email, width = 10, side = "left")
```

```
##  [1] "...any.com" "...igg.com" "...sta.org" "...ect.com" "...amp.com"
##  [6] "...key.com" "...@un.org" "...ovh.net" "...e.co.uk" "...ank.net"
```

```
str_trunc(mockdata$email, width = 10, side = "center")
```

```
##  [1] "mnew...com" "mdan...com" "hgir...org" "pmcm...com" "dris...com"
##  [6] "cphl...com" "kdod...org" "vhou...net" "rdik....uk" "tdud...net"
```

task 9:quickly sort the emails in both ascending and descending orders.

```
str_sort(mockdata$email)
```

```
##  [1] "cphlippi5@surveymonkey.com"   "drisbrough4@bandcamp.com"
##  [3] "hgirhard2@altervista.org"     "kdodswell6@un.org"
##  [5] "mdankersley1@digg.com"        "mnewburn0@fastcompany.com"
##  [7] "pmcmenamy3@sciencedirect.com" "rdike8@timesonline.co.uk"
##  [9] "tdudbridge9@clickbank.net"    "vhourihane7@ovh.net"
```

```
str_sort(mockdata$email, decreasing = TRUE)
```

```
##  [1] "vhourihane7@ovh.net"          "tdudbridge9@clickbank.net"
##  [3] "rdike8@timesonline.co.uk"     "pmcmenamy3@sciencedirect.com"
##  [5] "mnewburn0@fastcompany.com"    "mdankersley1@digg.com"
##  [7] "kdodswell6@un.org"            "hgirhard2@altervista.org"
##  [9] "drisbrough4@bandcamp.com"     "cphlippi5@surveymonkey.com"
```

task 10:Making full_name in uppercase.

```
str_to_upper(mockdata$full_name)
```

```
##  [1] "MUFI RUIT"          "LEESE FURMAGIER"    "BLAKELEE WILSHIRE"
##  [4] "TERENCIO MCILLRICK" "DEBEE MCERLAINE"    "FRAN PAINTEN"
##  [7] "FRASCO BOWICH"      "CAR PONTEN"         "TADES CHECCUCCI"
## [10] "WILTON KEMMEY"
```

task 11:Making full_name in lowercase.

```
str_to_lower(mockdata$full_name)
```

```
##  [1] "mufi ruit"          "leese furmagier"    "blakelee wilshire"
##  [4] "terencio mcillrick" "debee mcerlaine"    "fran painten"
##  [7] "frasco bowich"      "car ponten"         "tades checcucci"
## [10] "wilton kemmey"
```

task 12:Replacing word street with ST.

```
str_replace(mockdata$address, "Street", "ST")
```

```
##  [1] "8 Anhalt Crossing"          "697 East Avenue"
##  [3] "89 Dottie Circle"           "98135 Blue Bill Park Drive"
##  [5] "7814 Pennsylvania ST"       "4897 Little Fleur Drive"
##  [7] "53541 Morrow Center"        "4819 Hermina Parkway"
##  [9] "68096 Monument Park"        "9595 Spaight Avenue"
```

task 13:Replacing word Road with RD.

```
str_replace(mockdata$address, "Road", "RD")
```

```
##  [1] "8 Anhalt Crossing"          "697 East Avenue"
##  [3] "89 Dottie Circle"           "98135 Blue Bill Park Drive"
##  [5] "7814 Pennsylvania Street"   "4897 Little Fleur Drive"
##  [7] "53541 Morrow Center"        "4819 Hermina Parkway"
##  [9] "68096 Monument Park"        "9595 Spaight Avenue"
```

task 14:Extracting parts of the string that match a particular pattern(org) using str_extract().

```
str_extract(mockdata$email, pattern = "org")
```

```
##  [1] NA    NA    "org" NA    NA    NA    "org" NA    NA    NA
```

task 15:Using str_match() to see if the pattern is present in the string.

```
str_match(mockdata$email, pattern = "org")
```

```
##       [,1]
##  [1,] NA
##  [2,] NA
##  [3,] "org"
##  [4,] NA
##  [5,] NA
##  [6,] NA
##  [7,] "org"
##  [8,] NA
##  [9,] NA
## [10,] NA
```

task 16:Using str_which() to identify the index of the strings that match our pattern.

```
str_which(mockdata$email, pattern = "org")
```

```
## [1] 3 7
```

task 17: knowing the position of @ in the email ids,using str_locate().

```
str_locate(mockdata$email, pattern = "@")
```

```
##       start end
##  [1,]    10  10
##  [2,]    13  13
##  [3,]    10  10
##  [4,]    11  11
##  [5,]    12  12
##  [6,]    10  10
##  [7,]    11  11
##  [8,]    12  12
##  [9,]     7   7
## [10,]    12  12
```

task 18:Using str_length to ensure that the length of the strings in the password column.

```
str_length(mockdata$passwords)
```

```
##  [1] 16 16 16 16 16 16 16 16 16 16
```

task 19:Extracting the currency type from the currency column.

```
str_sub(mockdata$currency, start = 1, end = 1)
```

```
##  [1] "¥" "$" "€" "€" "€" "¥" "$" "¥" "€" "€"
```

task 20:using it to extract the first name from the full_name column.

```
word(mockdata$full_name, 1)
```

```
##  [1] "Mufi"     "Leese"    "Blakelee" "Terencio" "Debee"    "Fran"
##  [7] "Frasco"   "Car"      "Tades"    "Wilton"
```

task 21:using it to extract the last name from the full_name column.

```
word(mockdata$full_name, 2)
```

```
##  [1] "Ruit"      "Furmagier" "Wilshire"  "McIllrick" "McErlaine" "Painten"
##  [7] "Bowich"    "Ponten"    "Checcucci" "Kemmey"
```

Extract domain name from email ids

task 1:Displaying emails before we extract the domain names.

```
emails <-
  mockstring %>%
  pull(email) %>%
  head()

emails
```

```
## [1] "mnewburn0@fastcompany.com"    "mdankersley1@digg.com"
## [3] "hgirhard2@altervista.org"     "pmcmenamy3@sciencedirect.com"
## [5] "drisbrough4@bandcamp.com"     "cphlippi5@surveymonkey.com"
```

task 2:Spliting the email using using the pattern @.

```
 str_split(emails, pattern = '@')
```

```
## [[1]]
## [1] "mnewburn0"       "fastcompany.com"
##
## [[2]]
## [1] "mdankersley1" "digg.com"
##
## [[3]]
## [1] "hgirhard2"      "altervista.org"
##
## [[4]]
## [1] "pmcmenamy3"        "sciencedirect.com"
##
## [[5]]
## [1] "drisbrough4"  "bandcamp.com"
##
## [[6]]
## [1] "cphlippi5"        "surveymonkey.com"
```

task 3:Extract the second element from the list.

```
emails %>%
  str_split(pattern = '@') %>%
  map_chr(2)
```

```
## [1] "fastcompany.com"   "digg.com"          "altervista.org"
## [4] "sciencedirect.com" "bandcamp.com"      "surveymonkey.com"
```

task 4:Spliting the above using pattern \..

```
emails %>%
  str_split(pattern = '@') %>%
  map_chr(2) %>%
  str_split(pattern = '\\.')
```

```
## [[1]]
## [1] "fastcompany" "com"
##
## [[2]]
## [1] "digg" "com"
##
## [[3]]
## [1] "altervista" "org"
##
## [[4]]
## [1] "sciencedirect" "com"
##
## [[5]]
## [1] "bandcamp" "com"
##
## [[6]]
## [1] "surveymonkey" "com"
```

task 5:Extracting the first element from the list.

```
emails %>%
  str_split(pattern = '@') %>%
  map_chr(2) %>%
  str_split(pattern = '\\.') %>%
  map_chr(extract(1))
```

```
## [1] "fastcompany"   "digg"          "altervista"    "sciencedirect"
## [5] "bandcamp"      "surveymonkey"
```

task 6:Extracting Domain Extension.

```
emails %>%
  str_split(pattern = '@') %>%
  map_chr(2) %>%
  str_split(pattern = '\\.', simplify = TRUE) %>%
  extract(, 2)
```

```
## [1] "com" "com" "org" "com" "com" "com"
```

Extract image type from URL

task 1: Displaying the URL of the image.

```
img <-
  mockstring %>%
  pull(imageurl) %>%
  head()

img
```

```
## [1] "http://dummyimage.com/130x183.jpg/dddddd/000000"
## [2] "http://dummyimage.com/106x217.bmp/dddddd/000000"
## [3] "http://dummyimage.com/146x127.bmp/cc0000/ffffff"
## [4] "http://dummyimage.com/181x194.png/5fa2dd/ffffff"
## [5] "http://dummyimage.com/220x123.jpg/ff4444/ffffff"
## [6] "http://dummyimage.com/118x176.bmp/dddddd/000000"
```

task 2:Spliting imageurl using pattern \.

```
str_split(img, pattern = '\\.')
```

```
## [[1]]
## [1] "http://dummyimage" "com/130x183"       "jpg/dddddd/000000"
##
## [[2]]
## [1] "http://dummyimage" "com/106x217"       "bmp/dddddd/000000"
##
## [[3]]
## [1] "http://dummyimage" "com/146x127"       "bmp/cc0000/ffffff"
##
## [[4]]
## [1] "http://dummyimage" "com/181x194"       "png/5fa2dd/ffffff"
##
## [[5]]
## [1] "http://dummyimage" "com/220x123"       "jpg/ff4444/ffffff"
##
## [[6]]
## [1] "http://dummyimage" "com/118x176"       "bmp/dddddd/000000"
```

task 3:Extracting the third value from each element of the resulting list.

```
img %>%
  str_split(pattern = '\\.') %>%
  map_chr(extract(3))
```

```
## [1] "jpg/dddddd/000000" "bmp/dddddd/000000" "bmp/cc0000/ffffff"
## [4] "png/5fa2dd/ffffff" "jpg/ff4444/ffffff" "bmp/dddddd/000000"
```

task 4:subset the first 3 characters of the string using the index position.

```
img %>%
  str_split(pattern = '\\.') %>%
  map_chr(extract(3)) %>%
  str_sub(start = 1, end = 3)
```

```
## [1] "jpg" "bmp" "bmp" "png" "jpg" "bmp"
```

task 5:spliting the string using pattern / and extract the first value from the elements of the resulting list

```
img %>%
  str_split(pattern = '\\.') %>%
  map_chr(extract(3)) %>%
  str_split(pattern = '/') %>%
  map_chr(extract(1))
```

```
## [1] "jpg" "bmp" "bmp" "png" "jpg" "bmp"
```

Extract Image Dimesion from URL

task 1:Locating numbers between 0 and 9 using str_locate.

```
str_locate(img, pattern = "[0-9]")
```

```
##      start end
## [1,]    23  23
## [2,]    23  23
## [3,]    23  23
## [4,]    23  23
## [5,]    23  23
## [6,]    23  23
```

task 2:Extracting the part of the url that contains the image dimension using str_sub().

```
str_sub(img, start = 23)
```

```
## [1] "130x183.jpg/dddddd/000000" "106x217.bmp/dddddd/000000"
## [3] "146x127.bmp/cc0000/ffffff" "181x194.png/5fa2dd/ffffff"
## [5] "220x123.jpg/ff4444/ffffff" "118x176.bmp/dddddd/000000"
```

task 3: Spliting the string using the pattern \..

```
img %>%
  str_sub(start = 23) %>%
  str_split(pattern = '\\.')
```

```
## [[1]]
## [1] "130x183"           "jpg/dddddd/000000"
##
## [[2]]
```

```
## [1] "106x217"           "bmp/dddddd/000000"
##
## [[3]]
## [1] "146x127"           "bmp/cc0000/ffffff"
##
## [[4]]
## [1] "181x194"           "png/5fa2dd/ffffff"
##
## [[5]]
## [1] "220x123"           "jpg/ff4444/ffffff"
##
## [[6]]
## [1] "118x176"           "bmp/dddddd/000000"
```

task 4:Extracting the first element.

```
img %>%
  str_sub(start = 23) %>%
  str_split(pattern = '\\.') %>%
  map_chr(extract(1))
```

```
## [1] "130x183" "106x217" "146x127" "181x194" "220x123" "118x176"
```

Extract HTTP Protocol from URL

task 1:Displaying url

```
url1 <-
  mockstring %>%
  pull(url) %>%
  first()

url1
```

```
## [1] "https://engadget.com/nascetur/ridiculus/mus/vivamus/vestibulum.jsp?eu=est&tincidunt=risus&in=au
```

task 2: Spliting the url using the pattern ://.

```
str_split(url1, pattern = '://')
```

```
## [[1]]
## [1] "https"
## [2] "engadget.com/nascetur/ridiculus/mus/vivamus/vestibulum.jsp?eu=est&tincidunt=risus&in=auctor&leo
```

task 3:Extracting the first element.

```
url1 %>%
  str_split(pattern = '://') %>%
  map_chr(extract(1))
```

```
## [1] "https"
```

Extract file type

task 1:Displaying the urls

```
urls <-
  mockstring %>%
  use_series(url) %>%
  extract(1:3)

urls
```

```
## [1] "https://engadget.com/nascetur/ridiculus/mus/vivamus/vestibulum.jsp?eu=est&tincidunt=risus&in=au
## [2] "http://delicious.com/phasellus/in/felis/donec.json?interdum=risus&mauris=dapibus&non=augue&ligu
## [3] "https://w3.org/sed/augue/aliquam/erat/volutpat.json?dictumst=mi&morbi=sit&vestibulum=amet&velit=
```

task 2:Checking if there are only 2 dots in the URL

```
urls %>%
  str_locate_all(pattern = '\\.') %>%
  map_int(nrow) %>%
  is_greater_than(2) %>%
  sum()
```

```
## [1] 0
```

task 3:Checking if there is only 1 question mark in the URL

```
urls %>%
  str_locate_all(pattern = "[?]") %>%
  map_int(nrow) %>%
  is_greater_than(1) %>%
  sum()
```

```
## [1] 0
```

task 4: Detecting the staritng position of file type.

```
d <-
  urls %>%
  str_locate_all(pattern = '\\.') %>%
  map_int(extract(2)) %>%
  add(1)

d
```

```
## [1] 64 47 48
```

task 5:Detecting the ending position of file type.

```r
q <-
  urls %>%
  str_locate_all(pattern = "[?]") %>%
  map_int(extract(1)) %>%
  subtract(1)

q
```

```
## [1] 66 50 51
```

task 6:Specifying the index position for extracting file type

```r
str_sub(urls, start = d, end = q)
```

```
## [1] "jsp"  "json" "json"
```