Assignment1.R

RKC

2024-10-30

```
#### 1. Create the vectors: ####
#A
# (1, 2, 3, . . . , 19, 20)

v = c(1:20); v
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
# b.
# (20, 19, . . . , 2, 1)

v = c(20:1); v
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
# c.
# (1, 2, 3, . . . , 19, 20, 19, 18, . . . , 2, 1)
v = c(1:20,19:1 ); v
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 19 18 17 16 15
## [26] 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
# d.
# (4, 6, 3) and assign it to the name tmp.
tmp = c(4,6,3); tmp
```

```
## [1] 4 6 3
```

```
# e.
# (4, 6, 3, 4, 6, 3, . . . , 4, 6, 3) where there are 10 occurrences of 4.
rep(tmp, 10)
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3
```

```
# f.
# (4, 6, 3, 4, 6, 3, . . . , 4, 6, 3, 4) where there are 11
#     occurrences of 4, 10 occurrences of 6 and 10 occurrences of 3.
?rep
```

```
## starting httpd help server ... done
```

```r
rep(tmp, c(11,10,10))
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 3 3 3 3 3 3 3 3 3 3
```

```r
# g.
# (4, 4, . . . , 4, 6, 6, . . . , 6, 3, 3, . . . , 3) where there are 10
# occurrences of 4, 20 occurrences of 6 and 30 occurrences of 3.
rep(tmp, c(10,20,30))
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 3 3 3 3 3 3 3 3
## [39] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```r
##### 2. Create a following matrix in R ####
v = c(0:4)
z = matrix(c(v, v+1,v+2,v+3, v+4), 5,5); z
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    2    3    4
## [2,]    1    2    3    4    5
## [3,]    2    3    4    5    6
## [4,]    3    4    5    6    7
## [5,]    4    5    6    7    8
```

```r
#### 3. Write a R program to take input from the user (name and age) and  ####
# display the values.

name = readline("Enter Name as input --> "); name
```

```
## Enter Name as input -->
```

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

```r
# age = scan(nmax=1); age

##### 4. Write a R program to create a Dataframes which contain  ####
# details of 5 employees and display summary of the data.

emp1 = c("Pranay B Shah", "Siemens Mobility", 'Rolling Stocks', 'Bid and Project Management')
emp2 = c("Pranay B Shah", "Tetra Pak India", 'World Class Manufacturing factory - EMEA/APAC', 'Industria
emp3 = c("Pranay B Shah", "General Electric", 'Wind Turbines', 'Associate Engineer')
emp4 = c("Pranay B Shah", "Johnson Controls",'Building Design',  'SCABA BMS Design Engg')
emp5 = c("Pranay B Shah", "Go Digital technologies", 'US Energy & Power', 'Ass. Data Engineer')

offers <- data.frame(rbind(emp1, emp2, emp3, emp4, emp5))
rownames(offers) <- c("Offer1", "Offer2", "Offer3", "Offer4", "Offer5")
colnames(offers) <- c("Name", "Company", "Domain", "Role"); View(offers)

#### 5. Create two different 2 by 2 matrices named A and B.  ####
# A should contain the values 1 - 4 and B the values 5-8.
# Try out the following commands and by looking at the results
# see if you can figure out what is going on.
# •
# A
A = c(1,2,3,4)
m1 = matrix(A, 2,2); m1
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```r
B = c(5,6,7,8)
m2 = matrix(B, 2,2); m2
```

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```r
# •
# A * B
mul = A * B; mul
```

```
## [1]  5 12 21 32
```

```r
# •
# A / B
A / B
```

```
## [1] 0.2000000 0.3333333 0.4285714 0.5000000
```

```r
# •
# C = A %x% B

C = A %x% B; C
```

```
##  [1]  5  6  7  8 10 12 14 16 15 18 21 24 20 24 28 32
```

```r
# •
# D = A + B
D = A+B; D
```

```
## [1]  6  8 10 12
```

```r
# •
# E = A - B
E = A-B; E
```

```
## [1] -4 -4 -4 -4
```

```
# •
# A == B
A == B
```

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

```
#### 6. Create a 4*3 Matrix containing 12 numbers ####
# • What is the length and the mode of the matrix
m = matrix(1:12, 4, 3); m
```

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

```
# • Extract all values from matrix that are larger than 6.
m[m>6]
```

```
## [1]  7  8  9 10 11 12
```

```
# • Shift places of column 1 and 3
m[,c(3,2,1)]
```

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

```
# • Add a vector with three zeros as a fifth row to the matrix
m = rbind(m, c(0,0,0)); m
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
## [5,]    0    0    0
```

```
# • Replace all values the first two columns in your matrix with "NA
m[,1:2] = NA; m
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA    9
## [2,]   NA   NA   10
## [3,]   NA   NA   11
## [4,]   NA   NA   12
## [5,]   NA   NA    0
```

```
# • Replace all values in the matrix with 0 and convert it to a vector
m[,] = 0 ;m; v2 = c(m); v2
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
## [4,]    0    0    0
## [5,]    0    0    0
```

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

```
#### 7. Data frame ####
# • Write a R program to create a data frame from four given vectors.
SrNo = c(1:5); name = c('Pranay', 'Hemant', 'Bhupendra', 'Pratik', 'Yash');
age =c(1:5); v = data.frame(SrNo, name, age); v
```

```
##   SrNo      name age
## 1    1    Pranay   1
## 2    2    Hemant   2
## 3    3 Bhupendra   3
## 4    4    Pratik   4
## 5    5      Yash   5
```

```
# • Write a R program to get the structure of a given data frame
str(v)
```

```
## ’data.frame’:    5 obs. of  3 variables:
##  $ SrNo: int  1 2 3 4 5
##  $ name: chr  "Pranay" "Hemant" "Bhupendra" "Pratik" ...
##  $ age : int  1 2 3 4 5
```

```
# • Write a R program to get the statistical summary and nature of
# the data of a given data frame.
summary(v)
```

```
##       SrNo       name                age   
##  Min.   :1   Length:5           Min.   :1  
##  1st Qu.:2   Class :character   1st Qu.:2  
##  Median :3   Mode  :character   Median :3  
##  Mean   :3                      Mean   :3  
##  3rd Qu.:4                      3rd Qu.:4  
##  Max.   :5                      Max.   :5  
```

```
# • Write a R program to extract specific column from a data frame
# using column name
v$a; v$b
```

```
## [1] 1 2 3 4 5
```

```
## NULL
```

```r
# • Write a R program to extract first two rows from a given data frame.
v[1:2, ]
```

```
##   SrNo   name age
## 1    1 Pranay   1
## 2    2 Hemant   2
```

```r
# • Write a R program to extract 3rd and 5th rows with 1st and
# 3rd columns from a given data frame
v[c(3,5), c(1,3)]
```

```
##   SrNo age
## 3    3   3
## 5    5   5
```

```r
# • Write a R program to add a new column in a given data frame.
# v %>% mutate(
#   Newcolumn = age + 22
# ); v

v$City = c('Kalyan', 'Mumbai Central', 'Airoli', 'Andheri', 'Ville Parla'); v
```

```
##   SrNo      name age           City
## 1    1    Pranay   1         Kalyan
## 2    2    Hemant   2 Mumbai Central
## 3    3 Bhupendra   3         Airoli
## 4    4    Pratik   4        Andheri
## 5    5      Yash   5    Ville Parla
```

```r
# • Write a R program to add new row(s) to an existing data frame.
v$Engineering = c('Electrical', 'Electronics', 'Civil', 'Civil', 'Electronics'); v
```

```
##   SrNo      name age           City Engineering
## 1    1    Pranay   1         Kalyan  Electrical
## 2    2    Hemant   2 Mumbai Central Electronics
## 3    3 Bhupendra   3         Airoli       Civil
## 4    4    Pratik   4        Andheri       Civil
## 5    5      Yash   5    Ville Parla Electronics
```

```r
# • Write a R program to drop column(s) by name from a given data frame.
v$age = NULL; v
```

```
##   SrNo      name           City Engineering
## 1    1    Pranay         Kalyan  Electrical
## 2    2    Hemant Mumbai Central Electronics
## 3    3 Bhupendra         Airoli       Civil
## 4    4    Pratik        Andheri       Civil
## 5    5      Yash    Ville Parla Electronics
```

```
# • Write a R program to drop row(s) by number from a given data frame
v = v[-5, ] ; v
```

```
##   SrNo      name           City Engineering
## 1    1    Pranay         Kalyan  Electrical
## 2    2    Hemant Mumbai Central Electronics
## 3    3 Bhupendra         Airoli       Civil
## 4    4    Pratik        Andheri       Civil
```

```
# • Write a R program to sort a given data frame by multiple column(s).
?sort

v[order(v$name, v$Engineering), ]
```

```
##   SrNo      name           City Engineering
## 3    3 Bhupendra         Airoli       Civil
## 2    2    Hemant Mumbai Central Electronics
## 1    1    Pranay         Kalyan  Electrical
## 4    4    Pratik        Andheri       Civil
```

```
# • Write a R program to create inner, outer, left,
# right join(merge) from given two data frames.
df1 = data.frame(ID = c(1,2,3), Name = c("Pranay", 'Hemant', 'Ash'), Age = c(23,24,22)); df1
```

```
##   ID   Name Age
## 1  1 Pranay  23
## 2  2 Hemant  24
## 3  3    Ash  22
```

```
df2 = data.frame(ID = c(1,2,4), Role = c("Data Engineering", 'Data Analyst',
                                         'Marketing'),
                 Division = c('Finance', 'Marketing', 'Marketing')); df2
```

```
##   ID             Role  Division
## 1  1 Data Engineering   Finance
## 2  2     Data Analyst Marketing
## 3  4        Marketing Marketing
```

```
df3 = merge(df1, df2, by = "ID"); df3
```

```
##   ID   Name Age             Role  Division
## 1  1 Pranay  23 Data Engineering   Finance
## 2  2 Hemant  24     Data Analyst Marketing
```

```
# • Write a R program to replace NA values with 3 in a given data frame.
df = data.frame(
  A = c(1, NA, 3, NA),
  B = c(NA, 5, NA, 7),
  C = c(9, NA, 11, NA)
); df
```

```
##    A  B  C
## 1  1 NA  9
## 2 NA  5 NA
## 3  3 NA 11
## 4 NA  7 NA
```

```
df[is.na(df)] = 3; df
```

```
##   A B  C
## 1 1 3  9
## 2 3 5  3
## 3 3 3 11
## 4 3 7  3
```

```
# • Write a R program to change a column name of a given data frame.
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
```

```
df3 %>% rename(IdentityNumber = ID)
```

```
##   IdentityNumber   Name Age             Role  Division
## 1              1 Pranay  23 Data Engineering   Finance
## 2              2 Hemant  24     Data Analyst Marketing
```

```
# • Write a R program to change more than one column name of a given data frame.
df3 %>% rename(IdentityNumber = ID, FullName = Name)
```

```
##   IdentityNumber FullName Age             Role  Division
## 1              1   Pranay  23 Data Engineering   Finance
## 2              2   Hemant  24     Data Analyst Marketing
```

```
# • Write a R program to select some random rows from a given data frame.
df1[sample(nrow(df1), 2), ]
```

```
##   ID   Name Age
## 1  1 Pranay  23
## 3  3    Ash  22
```

```r
# • Write a R program to reorder an given data frame by column name
v[,c("SrNo", "City", "Engineering", 'name')]
```

```
##   SrNo           City Engineering      name
## 1    1         Kalyan  Electrical    Pranay
## 2    2 Mumbai Central Electronics    Hemant
## 3    3         Airoli       Civil Bhupendra
## 4    4        Andheri       Civil    Pratik
```

```r
# • Write a R program to compare two data frames to find the row(s) in first
# data frame that are not present in second data frame.
df1 = data.frame(
  A = c(1, 2, 3, 4),
  B = c("Pranay", "Hemant", "Onkar", "Pratik")
); df1
```

```
##   A      B
## 1 1 Pranay
## 2 2 Hemant
## 3 3  Onkar
## 4 4 Pratik
```

```r
df2 = data.frame(
  A = c(2, 3, 5),
  B = c("Hemant", "Onkar", "Bhupendra")
); df2
```

```
##   A         B
## 1 2    Hemant
## 2 3     Onkar
## 3 5 Bhupendra
```

```r
?intersect
library(dplyr)
setdiff(df1, df2)
```

```
##   A      B
## 1 1 Pranay
## 2 4 Pratik
```

```r
# • Write a R program to find elements which are present in two given data frame.
intersect(df1, df2)
```

```
##   A      B
## 1 2 Hemant
## 2 3  Onkar
```

```r
# • Write a R program to find elements come only once that are common to both given data frames.
intersect(df1, df2)
```

```
##   A      B
## 1 2 Hemant
## 2 3  Onkar
```

```
# • Create a dataframe then export it in .csv, .txt, .xlsx file.
library(writexl)
getwd()
```

```
## [1] "A:/CDAC_SM_VITA/5_R_Programming"
```

```
setwd("A:/CDAC_SM_VITA/5_R_Programming")
write.csv(df, "7_1_assign.csv")
write.table(df, "7_2_assign.txt")
write_xlsx(df, "7_3_assign.xlsx")
# • Write a R program to count the number of NA values in a data frame column.
df = data.frame(
  A = c(1, NA, 3, NA),
  B = c(NA, 5, NA, 7),
  C = c(9, NA, 11, NA)
); df
```

```
##    A  B  C
## 1  1 NA  9
## 2 NA  5 NA
## 3  3 NA 11
## 4 NA  7 NA
```

```
sapply(df, function(col) sum(is.na(col)))
```

```
## A B C
## 2 2 2
```

```
# • Write a R program to call the (built-in) dataset airquality.
# Remove the variables 'Solar.R' and 'Wind' and display the data frame
df = airquality

names(df)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```
df[,c(-2,-3)]
```

```
##     Ozone Temp Month Day
## 1      41   67     5   1
## 2      36   72     5   2
## 3      12   74     5   3
## 4      18   62     5   4
## 5      NA   56     5   5
## 6      28   66     5   6
## 7      23   65     5   7
## 8        19   59     5   8
## 9         8   61     5   9
## 10       NA   69     5  10
## 11        7   74     5  11
## 12       16   69     5  12
## 13       11   66     5  13
## 14       14   68     5  14
## 15       18   58     5  15
## 16       14   64     5  16
## 17       34   66     5  17
## 18        6   57     5  18
## 19       30   68     5  19
## 20       11   62     5  20
## 21        1   59     5  21
## 22       11   73     5  22
## 23        4   61     5  23
## 24       32   61     5  24
## 25       NA   57     5  25
## 26       NA   58     5  26
## 27       NA   57     5  27
## 28       23   67     5  28
## 29       45   81     5  29
## 30      115   79     5  30
## 31       37   76     5  31
## 32       NA   78     6   1
## 33       NA   74     6   2
## 34       NA   67     6   3
## 35       NA   84     6   4
## 36       NA   85     6   5
## 37       NA   79     6   6
## 38       29   82     6   7
## 39       NA   87     6   8
## 40       71   90     6   9
## 41       39   87     6  10
## 42       NA   93     6  11
## 43       NA   92     6  12
## 44       23   82     6  13
## 45       NA   80     6  14
## 46       NA   79     6  15
## 47       21   77     6  16
## 48       37   72     6  17
## 49       20   65     6  18
## 50       12   73     6  19
## 51       13   76     6  20
## 52       NA   77     6  21
## 53       NA   76     6  22
## 54       NA   76     6  23
## 55       NA   76     6  24
## 56       NA   75     6  25
## 57       NA   78     6  26
## 58       NA   73     6  27
## 59       NA   80     6  28
## 60       NA   77     6  29
## 61       NA   83     6  30
## 62    135   84     7    1
## 63     49   85     7    2
## 64     32   81     7    3
## 65     NA   84     7    4
## 66     64   83     7    5
## 67     40   83     7    6
## 68     77   88     7    7
## 69     97   92     7    8
## 70     97   92     7    9
## 71     85   89     7   10
## 72     NA   82     7   11
## 73     10   73     7   12
## 74     27   81     7   13
## 75     NA   91     7   14
## 76      7   80     7   15
## 77     48   81     7   16
## 78     35   82     7   17
## 79     61   84     7   18
## 80     79   87     7   19
## 81     63   85     7   20
## 82     16   74     7   21
## 83     NA   81     7   22
## 84     NA   82     7   23
## 85     80   86     7   24
## 86    108   85     7   25
## 87     20   82     7   26
## 88     52   86     7   27
## 89     82   88     7   28
## 90     50   86     7   29
## 91     64   83     7   30
## 92     59   81     7   31
## 93     39   81     8    1
## 94      9   81     8    2
## 95     16   82     8    3
## 96     78   86     8    4
## 97     35   85     8    5
## 98     66   87     8    6
## 99    122   89     8    7
## 100    89   90     8    8
## 101   110   90     8    9
## 102    NA   92     8   10
## 103    NA   86     8   11
## 104    44   86     8   12
## 105    28   82     8   13
## 106    65   80     8   14
## 107    NA   79     8   15
## 108    22   77     8   16
## 109    59   79     8   17
## 110    23   76     8   18
## 111    31   78     8   19
## 112    44   78     8   20
## 113    21   77     8   21
## 114     9   72     8   22
## 115    NA   75     8   23
## 116    45   79     8  24
## 117   168   81     8  25
## 118    73   86     8  26
## 119    NA   88     8  27
## 120    76   97     8  28
## 121   118   94     8  29
## 122    84   96     8  30
## 123    85   94     8  31
## 124    96   91     9   1
## 125    78   92     9   2
## 126    73   93     9   3
## 127    91   93     9   4
## 128    47   87     9   5
## 129    32   84     9   6
## 130    20   80     9   7
## 131    23   78     9   8
## 132    21   75     9   9
## 133    24   73     9  10
## 134    44   81     9  11
## 135    21   76     9  12
## 136    28   77     9  13
## 137     9   71     9  14
## 138    13   71     9  15
## 139    46   78     9  16
## 140    18   67     9  17
## 141    13   76     9  18
## 142    24   68     9  19
## 143    16   82     9  20
## 144    13   64     9  21
## 145    23   71     9  22
## 146    36   81     9  23
## 147     7   69     9  24
## 148    14   63     9  25
## 149    30   70     9  26
## 150    NA   77     9  27
## 151    14   75     9  28
## 152    18   76     9  29
## 153    20   68     9  30
```

```
#### 8. Create two vectors, vec1 and vec2, with at least 5 elements each and ####
# Perform element-wise addition, subtraction, multiplication, and division of
# vec1 and vec2
v1 = c(50:55); v1
```

```
## [1] 50 51 52 53 54 55
```

```
v2 = c(70:75); v2
```

```
## [1] 70 71 72 73 74 75
```

```
add = v1 + v2; add
```

```
## [1] 120 122 124 126 128 130
```

```r
sub = v1 - v2; sub;
```

```
## [1] -20 -20 -20 -20 -20 -20
```

```r
mul = v1 * v2; mul
```

```
## [1] 3500 3621 3744 3869 3996 4125
```

```r
div = v1 / v2; div
```

```
## [1] 0.7142857 0.7183099 0.7222222 0.7260274 0.7297297 0.7333333
```

```r
### 9. Create a vector named numbers with 10 random integers between 1 and 100 ###
random = c(sample(c(1:100), 10)); random
```

```
##  [1] 29 32 99 61 81 20 37 83 79 50
```

```r
#### 10. Create a vector named grades containing random scores between 0 and 100 ####
# for a class of 10 students.
library(randomNames)
Grade = c(sample(c(0:100), 10))
Names = randomNames(10);
df = data.frame(Names, Grade); df
```

```
##                       Names Grade
## 1          Sherer, Mamadou    34
## 2             Dwyer, Roger    24
## 3           Lucero, Michel    21
## 4  Fuoco-Martinez, Blanca     5
## 5           Gully, Matthew    54
## 6           Llamas, Julian    22
## 7         el-Rauf, Sitaara    73
## 8         Johnson, Sequoya    90
## 9              Gozeh, Khoa    16
## 10     al-Soliman, Aaisha    35
```

```r
?randomNames

# • Find the highest and lowest grades in the grades vector.
max(Grade); min(Grade)
```

```
## [1] 90
```

```
## [1] 5
```

```r
# • Create a new vector pass_fail based on the condition that any grade
# below 60 is a fail (0) and above or equal to 60 is a pass (1)
df$pass_fail = ifelse(df$Grade >= 60, 1,0); df
```

```
##                      Names Grade pass_fail
## 1          Sherer, Mamadou    34         0
## 2             Dwyer, Roger    24         0
## 3           Lucero, Michel    21         0
## 4   Fuoco-Martinez, Blanca     5         0
## 5           Gully, Matthew    54         0
## 6           Llamas, Julian    22         0
## 7         el-Rauf, Sitaara    73         1
## 8         Johnson, Sequoya    90         1
## 9              Gozeh, Khoa    16         0
## 10      al-Soliman, Aaisha    35         0
```

```
#### 11. Create a vector named original_vec with at least 8 elements.####
original_vec = c(20:27); original_vec
```

```
## [1] 20 21 22 23 24 25 26 27
```

```
# • Extract the 3rd through 6th elements of original_vec and store them in a
# new vector called subset_vec.
subset_vec = c(original_vec[3:6]); subset_vec
```

```
## [1] 22 23 24 25
```

```
# • Append two more elements to original_vec.
c(subset_vec, c(1:2))
```

```
## [1] 22 23 24 25  1  2
```

```
#• Calculate the mean of original_vec.
mean(subset_vec)
```

```
## [1] 23.5
```

```
# 12. Create a vector named ages with 10 random ages between 20 and 60.
age = sample(20:60, 10);age
```

```
##  [1] 49 22 27 24 59 44 56 32 50 58
```

```
# • Find the maximum and minimum ages in the ages vector.
max(age); min(age)
```

```
## [1] 59
```

```
## [1] 22
```

```
# • Create a new vector seniors with ages above 50
senior = age[age > 50]; senior
```

```
## [1] 59 56 58
```

```r
#### 13. Create a vector named original_vec with at least 10 elements. ####
original_vec = sample(c(10:100), 10); original_vec
```

```
##  [1] 77 74 11 51 98 52 78 30 36 35
```

```r
# • Extract the first, third, and fifth elements of original_vec
# and store them in a new vector called subset_vec.
original_vec[c(1,3,5)]
```

```
## [1] 77 11 98
```

```r
# • Sort original_vec in descending order.
sort(original_vec, decreasing = TRUE)
```

```
##  [1] 98 78 77 74 52 51 36 35 30 11
```

```r
#### 14. Create a random 4x4 matrix named random_mat. ####

random_mat = matrix(sample(1:100, 16), 4,4); random_mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   92   10   29   31
## [2,]    5   84   19   71
## [3,]   20   23   32   15
## [4,]   21   60   95   97
```

```r
# • Write a function row_mean that takes a matrix as input and
# returns a vector containing the mean of each row.
row_mean = function(matrices){
  for (i in 1:4) {
    print(mean(random_mat[i, ]))
  }
}
# • Use the row_mean function to find the row means of random_mat
row_mean(random_mat)
```

```
## [1] 40.5
## [1] 44.75
## [1] 22.5
## [1] 68.25
```

```r
#### 15. Create a 5x5 matrix named student_grades with random grades between 0 and 100. ####

student_grades = matrix(sample(0:100, 25), 5,5); student_grades
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    8    3   91   95   40
## [2,]   86   83   76   33   50
## [3,]   21   80   28   23    9
## [4,]   75   62   71   25   39
## [5,]   66   19   29   34   64
```

```
# • Find the highest grade in the matrix along with its row and column index.
# index(mean(student_grades))

which(student_grades == max(student_grades), arr.ind = TRUE)
```

```
##      row col
## [1,]   1   4
```

```
# • Create a new matrix pass_fail based on the condition that any grade
# below 60 is a fail (0) and above or equal to 60 is a pass (1)

pass_fail = matrix(ifelse(student_grades>60,1,0), 5, 5); pass_fail
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    1    1    0
## [2,]    1    1    1    0    0
## [3,]    0    1    0    0    0
## [4,]    1    1    1    0    0
## [5,]    1    0    0    0    1
```

```
# 16. Create a 3x3 matrix named mat1 with elements 1 to 9.
mat1 = matrix(c(1:9), 3, 3); mat1
```

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

```
# • Define a 2x4 matrix mat2 with all elements set to 0.
mat2 = matrix(c(0), 2,4); mat2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
```

```
# • What is the difference between cbind() and rbind() functions when
# creating matrices?
  # cbind()
  # Combines objects by columns.
  # cbind() is used when we want to add additional columns to an existing data frame or matrix.

  # rbind()
  #Combines objects by rows.
  # rbind() is used when we want to add additional rows to an existing data frame or matrix.

# 17. Create a data frame named students with columns: Name, Age, Grade,
# and Gender, containing information for at least 5 students.

students = data.frame(Name =c('Pranay', 'Hemant', 'Bhupendra', 'Pratik', 'Yash'),
                      Age = sample(20:29, 5),
                      Grade = sample(80:100, 5),
                      Gender = c('M')); students
```

```
##         Name Age Grade Gender
## 1     Pranay  22    91      M
## 2     Hemant  28    87      M
## 3 Bhupendra  26    92      M
## 4     Pratik  24    94      M
## 5       Yash  20    89      M
```

```
?randomNames
# • Display the first 3 rows of the students data frame.
students[1:3,]
```

```
##         Name Age Grade Gender
## 1     Pranay  22    91      M
## 2     Hemant  28    87      M
## 3 Bhupendra  26    92      M
```

```
# • Calculate the average age of the students.
mean(students$Age)
```

```
## [1] 24
```

```
# 18. Extract the Grade column from the students data frame.
students$Grade
```

```
## [1] 91 87 92 94 89
```

```
# • Select the rows where the Grade is greater than or equal to 85.
students[(Grade >= 85), ]
```

```
##    Name Age Grade Gender
## NA <NA>  NA    NA   <NA>
```

```
# • Create a new data frame called female_students containing only the female students.
female_students = students[students$Gender == 'M',]; female_students
```

```
##         Name Age Grade Gender
## 1     Pranay  22    91      M
## 2     Hemant  28    87      M
## 3 Bhupendra  26    92      M
## 4     Pratik  24    94      M
## 5       Yash  20    89      M
```

```
# Since I dont have any female in above df, I've filtered using 'M'


#### 19. Create a dataframe named my_data with
# columns: Name, Age, City, and Salary containing information for at least 5 individuals.

my_data = data.frame(Name =c('Pranay', 'Hemant', 'Bhupendra', 'Pratik', 'Yash'),
                     Age = sample(20:29, 5),
                     City = c('Kalyan', 'Mumbai Central', 'Airoli', 'Andheri', 'Ville Parla'),
                     'Salary(K in Rs)' = sample(70:120 ,5),
                     Experience = sample(1:10, 5)
); my_data
```

```
##        Name Age           City Salary.K.in.Rs. Experience
## 1    Pranay  20         Kalyan             107         10
## 2    Hemant  27 Mumbai Central             119          4
## 3 Bhupendra  25         Airoli              75          6
## 4    Pratik  26        Andheri              84          2
## 5      Yash  22    Ville Parla              85          8
```

```
# • Display the first 5 rows of my_data.
my_data %>% head(5)
```

```
##        Name Age           City Salary.K.in.Rs. Experience
## 1    Pranay  20         Kalyan             107         10
## 2    Hemant  27 Mumbai Central             119          4
## 3 Bhupendra  25         Airoli              75          6
## 4    Pratik  26        Andheri              84          2
## 5      Yash  22    Ville Parla              85          8
```

```
# • Calculate the average salary in my_data.
mean(my_data$Salary.K.in.Rs.)
```

```
## [1] 94
```

```
# • Extract the Age column from my_data.
my_data$Age
```

```
## [1] 20 27 25 26 22
```

```
# • Select the rows where the Age is greater than 30.
subset(my_data, Age > 30)
```

```
## [1] Name            Age             City            Salary.K.in.Rs.
## [5] Experience
## <0 rows> (or 0-length row.names)
```

```
# • Create a new dataframe named high_earners containing only
# individuals with a salary above $50,000.
my_data1 = data.frame(high_earners = subset(my_data, Salary.K.in.Rs. > 50)); my_data1
```

```
##   high_earners.Name high_earners.Age high_earners.City
## 1            Pranay               20            Kalyan
## 2            Hemant               27    Mumbai Central
## 3         Bhupendra               25            Airoli
## 4            Pratik               26           Andheri
## 5              Yash               22       Ville Parla
##   high_earners.Salary.K.in.Rs. high_earners.Experience
## 1                          107                      10
## 2                          119                       4
## 3                           75                       6
## 4                           84                       2
## 5                           85                       8
```

```
# • Add a new column named Education to my_data, indicating the highest level
# of education for each individual.
my_data$Education = c('Electrical Engg', 'Electronics Engg', 'Civil Engg', 'Civil Engg', 'Electronics E
```

```
##        Name Age           City Salary.K.in.Rs. Experience        Education
## 1    Pranay  20         Kalyan             107         10  Electrical Engg
## 2    Hemant  27 Mumbai Central             119          4 Electronics Engg
## 3 Bhupendra  25         Airoli              75          6       Civil Engg
## 4    Pratik  26        Andheri              84          2       Civil Engg
## 5      Yash  22    Ville Parla              85          8 Electronics Engg
```

```
# • Rename the column City to Residence.
rename(my_data, Residence = City)
```

```
##        Name Age      Residence Salary.K.in.Rs. Experience        Education
## 1    Pranay  20         Kalyan             107         10  Electrical Engg
## 2    Hemant  27 Mumbai Central             119          4 Electronics Engg
## 3 Bhupendra  25         Airoli              75          6       Civil Engg
## 4    Pratik  26        Andheri              84          2       Civil Engg
## 5      Yash  22    Ville Parla              85          8 Electronics Engg
```

```
# • Remove the Salary column from my_data.
my_data[-4]
```

```
##        Name Age           City Experience        Education
## 1    Pranay  20         Kalyan         10  Electrical Engg
## 2    Hemant  27 Mumbai Central          4 Electronics Engg
## 3 Bhupendra  25         Airoli          6       Civil Engg
## 4    Pratik  26        Andheri          2       Civil Engg
## 5      Yash  22    Ville Parla          8 Electronics Engg
```

```
# • Find the maximum and minimum ages in the dataframe.
max(my_data$Age); min(my_data$Age)
```

```
## [1] 27
```

```
## [1] 20
```

```
# • Calculate the mean salary of individuals with more than 5 years of experience.
my_data[(my_data$Experience > 5),]
```

```
##        Name Age        City Salary.K.in.Rs. Experience        Education
## 1    Pranay  20      Kalyan             107         10  Electrical Engg
## 3 Bhupendra  25      Airoli              75          6       Civil Engg
## 5      Yash  22 Ville Parla              85          8 Electronics Engg
```

```
mean(my_data$Salary.K.in.Rs.[(my_data$Experience > 5)])
```

```
## [1] 89
```

```
# • Determine the number of individuals from each city.
library(dplyr)
my_data  %>% group_by(my_data$City) %>% summarise(Count = n())
```

```
## # A tibble: 5 x 2
##   ‘my_data$City‘ Count
##   <chr>          <int>
## 1 Airoli             1
## 2 Andheri            1
## 3 Kalyan             1
## 4 Mumbai Central     1
## 5 Ville Parla        1
```

```
# 20. Load the inbuilt iris dataset and display its first 6 rows.
df = iris
df %>% head(6)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# • Filter the dataset to include only rows where Sepal.Width is greater than 3.
df['Sepal.Width' > 3, ]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
## 19           5.7         3.8          1.7         0.3     setosa
## 20           5.1         3.8          1.5         0.3     setosa
## 21           5.4         3.4          1.7         0.2     setosa
## 22           5.1         3.7          1.5         0.4     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 24           5.1         3.3          1.7         0.5     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 26           5.0         3.0          1.6         0.2     setosa
## 27           5.0         3.4          1.6         0.4     setosa
## 28           5.2         3.5          1.5         0.2     setosa
## 29           5.2         3.4          1.4         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
## 32           5.4         3.4          1.5         0.4     setosa
## 33           5.2         4.1          1.5         0.1     setosa
## 34           5.5         4.2          1.4         0.2     setosa
## 35           4.9         3.1          1.5         0.2     setosa
## 36           5.0         3.2          1.2         0.2     setosa
## 37           5.5         3.5          1.3         0.2     setosa
## 38           4.9         3.6          1.4         0.1     setosa
## 39           4.4         3.0          1.3         0.2     setosa
## 40           5.1         3.4          1.5         0.2     setosa
## 41           5.0         3.5          1.3         0.3     setosa
## 42           4.5         2.3          1.3         0.3     setosa
## 43           4.4         3.2          1.3         0.2     setosa
## 44           5.0         3.5          1.6         0.6     setosa
## 45           5.1         3.8          1.9         0.4     setosa
## 46           4.8         3.0          1.4         0.3     setosa
## 47           5.1         3.8          1.6         0.2     setosa
## 48           4.6         3.2          1.4         0.2     setosa
## 49           5.3         3.7          1.5         0.2     setosa
## 50           5.0         3.3          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
## 52           6.4         3.2          4.5         1.5 versicolor
## 53           6.9         3.1          4.9         1.5 versicolor
## 54           5.5         2.3          4.0         1.3 versicolor
## 55           6.5         2.8          4.6         1.5 versicolor
## 56           5.7         2.8          4.5         1.3 versicolor
## 57           6.3         3.3          4.7         1.6 versicolor
## 58           4.9         2.4          3.3         1.0 versicolor
## 59           6.6         2.9          4.6         1.3 versicolor
## 60           5.2         2.7          3.9         1.4 versicolor
## 61           5.0         2.0          3.5         1.0 versicolor
## 62           5.9         3.0          4.2         1.5 versicolor
## 63           6.0         2.2          4.0         1.0 versicolor
## 64           6.1         2.9          4.7         1.4 versicolor
## 65           5.6         2.9          3.6         1.3 versicolor
## 66           6.7         3.1          4.4         1.4 versicolor
## 67           5.6         3.0          4.5         1.5 versicolor
## 68           5.8         2.7          4.1         1.0 versicolor
## 69           6.2         2.2          4.5         1.5 versicolor
## 70           5.6         2.5          3.9         1.1 versicolor
## 71           5.9         3.2          4.8         1.8 versicolor
## 72           6.1         2.8          4.0         1.3 versicolor
## 73           6.3         2.5          4.9         1.5 versicolor
## 74           6.1         2.8          4.7         1.2 versicolor
## 75           6.4         2.9          4.3         1.3 versicolor
## 76           6.6         3.0          4.4         1.4 versicolor
## 77           6.8         2.8          4.8         1.4 versicolor
## 78           6.7         3.0          5.0         1.7 versicolor
## 79           6.0         2.9          4.5         1.5 versicolor
## 80           5.7         2.6          3.5         1.0 versicolor
## 81           5.5         2.4          3.8         1.1 versicolor
## 82           5.5         2.4          3.7         1.0 versicolor
## 83           5.8         2.7          3.9         1.2 versicolor
## 84           6.0         2.7          5.1         1.6 versicolor
## 85           5.4         3.0          4.5         1.5 versicolor
## 86           6.0         3.4          4.5         1.6 versicolor
## 87           6.7         3.1          4.7         1.5 versicolor
## 88           6.3         2.3          4.4         1.3 versicolor
## 89           5.6         3.0          4.1         1.3 versicolor
## 90           5.5         2.5          4.0         1.3 versicolor
## 91           5.5         2.6          4.4         1.2 versicolor
## 92           6.1         3.0          4.6         1.4 versicolor
## 93           5.8         2.6          4.0         1.2 versicolor
## 94           5.0         2.3          3.3         1.0 versicolor
## 95           5.6         2.7          4.2         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
## 97           5.7         2.9          4.2         1.3 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 99           5.1         2.5          3.0         1.1 versicolor
## 100          5.7         2.8          4.1         1.3 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

```
# • Calculate the mean Petal.Length for each species.
df %>% group_by(Species)  %>% summarise(mean_petal_length = mean(Petal.Length))
```

```
## # A tibble: 3 x 2
##   Species    mean_petal_length
##   <fct>                  <dbl>
## 1 setosa                  1.46
## 2 versicolor              4.26
## 3 virginica               5.55
```

```
# 21 Load the mtcars dataset and display its first 5 rows.
df = mtcars


# • Create a new column named Miles_per_Gallon by converting mpg to
# kilometers per liter (1 mile = 1.60934 kilometers).

df %>% mutate(Miles_per_Gallon = (mpg * 1.60934) / 3.78541)
```

```
##                      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
##                     Miles_per_Gallon
## Mazda RX4                   8.928000
## Mazda RX4 Wag               8.928000
## Datsun 710                  9.693257
## Hornet 4 Drive              9.098057
## Hornet Sportabout           7.950171
## Valiant                     7.695086
## Duster 360                  6.079543
## Merc 240D                  10.373486
## Merc 230                    9.693257
## Merc 280                    8.162743
## Merc 280C                   7.567543
## Merc 450SE                  6.972343
## Merc 450SL                  7.354971
## Merc 450SLC                 6.462171
## Cadillac Fleetwood          4.421486
## Lincoln Continental         4.421486
## Chrysler Imperial           6.249600
## Fiat 128                   13.774628
## Honda Civic                12.924343
## Toyota Corolla             14.412343
## Toyota Corona               9.140571
## Dodge Challenger            6.589714
## AMC Javelin                 6.462171
## Camaro Z28                  5.654400
## Pontiac Firebird            8.162743
## Fiat X1-9                  11.606400
## Porsche 914-2              11.053714
## Lotus Europa               12.924343
## Ford Pantera L              6.717257
## Ferrari Dino                8.375314
## Maserati Bora               6.377143
## Volvo 142E                  9.098057
```

```r
# • Find the car with the highest horsepower.
df %>% filter(hp == max(hp))
```

```
##                mpg cyl disp  hp drat   wt qsec vs am gear carb
## Maserati Bora  15   8  301 335 3.54 3.57 14.6  0  1    5    8
```

```r
names(df)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```r
# 22 Load the ChickWeight dataset and display the first few rows.

df = ChickWeight

# • How many rows and columns does the dataset have?
dim(df)
```

```
## [1] 578   4
```

```r
#   • What are the unique values in the Diet column?
unique(df$Diet)
```

```
## [1] 1 2 3 4
## Levels: 1 2 3 4
```

```r
#   • Calculate the average weight of all chicks in the dataset.
mean(df$weight)
```

```
## [1] 121.8183
```

```r
# • Find the maximum and minimum weight of chicks.
max(df$weight); min(df$weight)
```

```
## [1] 373
```

```
## [1] 35
```

```r
# • Calculate the total number of observations for each Diet type.
df %>%
  group_by(Diet) %>%
  summarise(total_observations = n())
```

```
## # A tibble: 4 x 2
##   Diet  total_observations
##   <fct>              <int>
## 1 1                    220
## 2 2                    120
## 3 3                    120
## 4 4                    118
```

```
# • Create a new dataframe high_weight containing chicks with weight greater than 100.
df %>% filter(weight > 100)
```

```
##     weight Time Chick Diet
## 1      106   12     1    1
## 2      125   14     1    1
## 3      149   16     1    1
## 4      171   18     1    1
## 5      199   20     1    1
## 6      205   21     1    1
## 7      103   10     2    1
## 8      122   12     2    1
## 9      138   14     2    1
## 10     162   16     2    1
## 11     187   18     2    1
## 12     209   20     2    1
## 13     215   21     2    1
## 14     115   12     3    1
## 15     138   14     3    1
## 16     163   16     3    1
## 17     187   18     3    1
## 18     198   20     3    1
## 19     202   21     3    1
## 20     102   12     4    1
## 21     108   14     4    1
## 22     136   16     4    1
## 23     154   18     4    1
## 24     160   20     4    1
## 25     157   21     4    1
## 26     106   10     5    1
## 27     141   12     5    1
## 28     164   14     5    1
## 29     197   16     5    1
## 30     199   18     5    1
## 31     220   20     5    1
## 32     223   21     5    1
## 33     124   10     6    1
## 34     141   12     6    1
## 35     148   14     6    1
## 36     155   16     6    1
## 37     160   18     6    1
## 38     160   20     6    1
## 39     157   21     6    1
## 40     112   10     7    1
## 41     146   12     7    1
## 42     174   14     7    1
## 43     218   16     7    1
## 44     250   18     7    1
## 45     288   20     7    1
## 46     305   21     7    1
## 47     110   12     8    1
## 48     116   14     8    1
## 49     126   16     8    1
## 50     134   18     8    1
## 51     125   20     8    1
## 52     101   16    10    1
## 53     112   18    10    1
## 54     120   20    10    1
## 55     124   21    10    1
## 56     112    8    11    1
## 57     139   10    11    1
## 58     168   12    11    1
## 59     177   14    11    1
## 60     182   16    11    1
## 61     184   18    11    1
## 62     181   20    11    1
## 63     175   21    11    1
## 64     119   12    12    1
## 65     135   14    12    1
## 66     162   16    12    1
## 67     185   18    12    1
## 68     195   20    12    1
## 69     205   21    12    1
## 70     101    8    14    1
## 71     128   10    14    1
## 72     164   12    14    1
## 73     192   14    14    1
## 74     227   16    14    1
## 75     248   18    14    1
## 76     259   20    14    1
## 77     266   21    14    1
## 78     103   14    17    1
## 79     113   16    17    1
## 80     123   18    17    1
## 81     133   20    17    1
## 82     142   21    17    1
## 83     106   16    19    1
## 84     120   18    19    1
## 85     144   20    19    1
## 86     157   21    19    1
## 87     107   18    20    1
## 88     115   20    20    1
## 89     117   21    20    1
## 90     125    8    21    2
## 91     163   10    21    2
## 92     217   12    21    2
## 93     240   14    21    2
## 94     275   16    21    2
## 95     307   18    21    2
## 96     318   20    21    2
## 97     331  21  21  2
## 98     108  12  22  2
## 99     111  14  22  2
## 100    131  16  22  2
## 101    148  18  22  2
## 102    164  20  22  2
## 103    167  21  22  2
## 104    103  10  23  2
## 105    127  12  23  2
## 106    135  14  23  2
## 107    145  16  23  2
## 108    163  18  23  2
## 109    170  20  23  2
## 110    175  21  23  2
## 111    102   8  25  2
## 112    124  10  25  2
## 113    146  12  25  2
## 114    164  14  25  2
## 115    197  16  25  2
## 116    231  18  25  2
## 117    259  20  25  2
## 118    265  21  25  2
## 119    114  10  26  2
## 120    136  12  26  2
## 121    147  14  26  2
## 122    169  16  26  2
## 123    205  18  26  2
## 124    236  20  26  2
## 125    251  21  26  2
## 126    115  12  27  2
## 127    123  14  27  2
## 128    144  16  27  2
## 129    163  18  27  2
## 130    185  20  27  2
## 131    192  21  27  2
## 132    114  10  28  2
## 133    145  12  28  2
## 134    156  14  28  2
## 135    184  16  28  2
## 136    207  18  28  2
## 137    212  20  28  2
## 138    233  21  28  2
## 139    106  10  29  2
## 140    134  12  29  2
## 141    150  14  29  2
## 142    187  16  29  2
## 143    230  18  29  2
## 144    279  20  29  2
## 145    309  21  29  2
## 146    115  12  30  2
## 147    122  14  30  2
## 148    143  16  30  2
## 149    151  18  30  2
## 150    157  20  30  2
## 151    150   21    30    2
## 152    102   10    31    3
## 153    123   12    31    3
## 154    138   14    31    3
## 155    170   16    31    3
## 156    204   18    31    3
## 157    235   20    31    3
## 158    256   21    31    3
## 159    107    8    32    3
## 160    129   10    32    3
## 161    159   12    32    3
## 162    179   14    32    3
## 163    221   16    32    3
## 164    263   18    32    3
## 165    291   20    32    3
## 166    305   21    32    3
## 167    111   10    33    3
## 168    137   12    33    3
## 169    144   14    33    3
## 170    151   16    33    3
## 171    146   18    33    3
## 172    156   20    33    3
## 173    147   21    33    3
## 174    107    8    34    3
## 175    134   10    34    3
## 176    164   12    34    3
## 177    186   14    34    3
## 178    235   16    34    3
## 179    294   18    34    3
## 180    327   20    34    3
## 181    341   21    34    3
## 182    123    8    35    3
## 183    158   10    35    3
## 184    201   12    35    3
## 185    238   14    35    3
## 186    287   16    35    3
## 187    332   18    35    3
## 188    361   20    35    3
## 189    373   21    35    3
## 190    116   10    36    3
## 191    145   12    36    3
## 192    166   14    36    3
## 193    198   16    36    3
## 194    227   18    36    3
## 195    225   20    36    3
## 196    220   21    36    3
## 197    103   12    37    3
## 198    112   14    37    3
## 199    135   16    37    3
## 200    157   18    37    3
## 201    169   20    37    3
## 202    178   21    37    3
## 203    109   10    38    3
## 204    128   12    38    3
## 205    154   14    38    3
## 206    192   16    38    3
## 207    232   18    38    3
## 208    280   20    38    3
## 209    290   21    38    3
## 210    109   10    39    3
## 211    130   12    39    3
## 212    146   14    39    3
## 213    170   16    39    3
## 214    214   18    39    3
## 215    250   20    39    3
## 216    272   21    39    3
## 217    101    8    40    3
## 218    120   10    40    3
## 219    154   12    40    3
## 220    182   14    40    3
## 221    215   16    40    3
## 222    262   18    40    3
## 223    295   20    40    3
## 224    321   21    40    3
## 225    103    8    41    4
## 226    124   10    41    4
## 227    155   12    41    4
## 228    153   14    41    4
## 229    175   16    41    4
## 230    184   18    41    4
## 231    199   20    41    4
## 232    204   21    41    4
## 233    103    8    42    4
## 234    126   10    42    4
## 235    160   12    42    4
## 236    174   14    42    4
## 237    204   16    42    4
## 238    234   18    42    4
## 239    269   20    42    4
## 240    281   21    42    4
## 241    131    8    43    4
## 242    157   10    43    4
## 243    184   12    43    4
## 244    188   14    43    4
## 245    197   16    43    4
## 246    198   18    43    4
## 247    199   20    43    4
## 248    200   21    43    4
## 249    103    8    44    4
## 250    118   10    44    4
## 251    127   12    44    4
## 252    138   14    44    4
## 253    145   16    44    4
## 254    146   18    44    4
## 255    117   10    45    4
## 256    135   12    45    4
## 257    141   14    45    4
## 258    147   16    45    4
## 259    174   18    45    4
## 260    197   20    45    4
## 261    196   21    45    4
## 262    101    8    46    4
## 263    120   10    46    4
## 264    144   12    46    4
## 265    156   14    46    4
## 266    173   16    46    4
## 267    210   18    46    4
## 268    231   20    46    4
## 269    238   21    46    4
## 270    123   10    47    4
## 271    148   12    47    4
## 272    157   14    47    4
## 273    168   16    47    4
## 274    185   18    47    4
## 275    210   20    47    4
## 276    205   21    47    4
## 277    104    8    48    4
## 278    125   10    48    4
## 279    154   12    48    4
## 280    170   14    48    4
## 281    222   16    48    4
## 282    261   18    48    4
## 283    303   20    48    4
## 284    322   21    48    4
## 285    108    8    49    4
## 286    128   10    49    4
## 287    152   12    49    4
## 288    166   14    49    4
## 289    184   16    49    4
## 290    203   18    49    4
## 291    233   20    49    4
## 292    237   21    49    4
## 293    105    8    50    4
## 294    122   10    50    4
## 295    155   12    50    4
## 296    175   14    50    4
## 297    205   16    50    4
## 298    234   18    50    4
## 299    264   20    50    4
## 300    264   21    50    4
```

```
# • Extract the rows where Diet is equal to 1 and Time is greater than 10.
df %>% filter(Diet == 1 & Time > 10)
```

```
##     weight Time Chick Diet
## 1      106   12     1    1
## 2      125   14     1    1
## 3      149   16     1    1
## 4      171   18     1    1
## 5      199   20     1    1
## 6      205   21     1    1
## 7      122   12     2    1
## 8    138   14    2   1
## 9    162   16    2   1
## 10   187   18    2   1
## 11   209   20    2   1
## 12   215   21    2   1
## 13   115   12    3   1
## 14   138   14    3   1
## 15   163   16    3   1
## 16   187   18    3   1
## 17   198   20    3   1
## 18   202   21    3   1
## 19   102   12    4   1
## 20   108   14    4   1
## 21   136   16    4   1
## 22   154   18    4   1
## 23   160   20    4   1
## 24   157   21    4   1
## 25   141   12    5   1
## 26   164   14    5   1
## 27   197   16    5   1
## 28   199   18    5   1
## 29   220   20    5   1
## 30   223   21    5   1
## 31   141   12    6   1
## 32   148   14    6   1
## 33   155   16    6   1
## 34   160   18    6   1
## 35   160   20    6   1
## 36   157   21    6   1
## 37   146   12    7   1
## 38   174   14    7   1
## 39   218   16    7   1
## 40   250   18    7   1
## 41   288   20    7   1
## 42   305   21    7   1
## 43   110   12    8   1
## 44   116   14    8   1
## 45   126   16    8   1
## 46   134   18    8   1
## 47   125   20    8   1
## 48    90   12    9   1
## 49    92   14    9   1
## 50    93   16    9   1
## 51   100   18    9   1
## 52   100   20    9   1
## 53    98   21    9   1
## 54    89   12   10   1
## 55    96   14   10   1
## 56   101   16   10   1
## 57   112   18   10   1
## 58   120   20   10   1
## 59   124   21   10   1
## 60   168   12   11   1
## 61   177   14   11   1
## 62      182   16     11     1
## 63      184   18     11     1
## 64      181   20     11     1
## 65      175   21     11     1
## 66      119   12     12     1
## 67      135   14     12     1
## 68      162   16     12     1
## 69      185   18     12     1
## 70      195   20     12     1
## 71      205   21     12     1
## 72       71   12     13     1
## 73       70   14     13     1
## 74       71   16     13     1
## 75       81   18     13     1
## 76       91   20     13     1
## 77       96   21     13     1
## 78      164   12     14     1
## 79      192   14     14     1
## 80      227   16     14     1
## 81      248   18     14     1
## 82      259   20     14     1
## 83      266   21     14     1
## 84       67   12     15     1
## 85       68   14     15     1
## 86       54   12     16     1
## 87       98   12     17     1
## 88      103   14     17     1
## 89      113   16     17     1
## 90      123   18     17     1
## 91      133   20     17     1
## 92      142   21     17     1
## 93       82   12     19     1
## 94       88   14     19     1
## 95      106   16     19     1
## 96      120   18     19     1
## 97      144   20     19     1
## 98      157   21     19     1
## 99       77   12     20     1
## 100      89   14     20     1
## 101      98   16     20     1
## 102     107   18     20     1
## 103     115   20     20     1
## 104     117   21     20     1
```

```
# • Find the average weight of chicks for each Diet type and Time point.
df %>% group_by(Diet, Time) %>% summarise(average_weight = mean(weight))
```

```
## 'summarise()' has grouped output by 'Diet'. You can override using the
## '.groups' argument.
```

```
## # A tibble: 48 x 3
## # Groups:   Diet [4]
##    Diet   Time average_weight
##    <fct> <dbl>          <dbl>
##  1 1         0           41.4
##  2 1         2           47.2
##  3 1         4           56.5
##  4 1         6           66.8
##  5 1         8           79.7
##  6 1        10           93.1
##  7 1        12          109.
##  8 1        14          123.
##  9 1        16          145.
## 10 1        18          159.
## # i 38 more rows
```

```
# • Calculate the average weight for each combination of Diet and Time.
df %>% group_by(Diet, Time) %>% summarise(average_weight = mean(weight))
```

```
## `summarise()` has grouped output by 'Diet'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 48 x 3
## # Groups:   Diet [4]
##    Diet   Time average_weight
##    <fct> <dbl>          <dbl>
##  1 1         0           41.4
##  2 1         2           47.2
##  3 1         4           56.5
##  4 1         6           66.8
##  5 1         8           79.7
##  6 1        10           93.1
##  7 1        12          109.
##  8 1        14          123.
##  9 1        16          145.
## 10 1        18          159.
## # i 38 more rows
```

```
# • Find the chick with the highest weight in each Diet group.
df %>% group_by(Diet) %>% filter(weight == max(weight))
```

```
## # A tibble: 4 x 4
## # Groups:   Diet [4]
##   weight  Time Chick Diet
##    <dbl> <dbl> <ord> <fct>
## 1    305    21 7     1
## 2    331    21 21    2
## 3    373    21 35    3
## 4    322    21 48    4
```

```
# • Determine the total weight gain for each chick.
df %>% group_by(Chick) %>% summarise(total_gain = max(weight) - min(weight))
```

```
## # A tibble: 50 x 2
##    Chick total_gain
##    <ord>       <dbl>
##  1 18              4
##  2 16             16
##  3 15             27
##  4 13             55
##  5 9              58
##  6 20             76
##  7 10             83
##  8 8              92
##  9 17            100
## 10 19            114
## # i 40 more rows
```