dim(df)
#Displays the type and a preview of all columns as a row so that it's very easy to
take in.library(dplyr)
glimpse(df)

```
       ID          Gender               Grade        Horoscope           Subject             IntExt
 Min.   :  1   Length:185         Min.   :3.000   Length:185         Length:185         Length:185
 1st Qu.: 47   Class :character   1st Qu.:5.000   Class :character   Class :character   Class :character
 Median : 93   Mode  :character   Median :6.000   Mode  :character   Mode  :character   Mode  :character
 Mean   : 93                      Mean   :5.735
 3rd Qu.:139                      3rd Qu.:7.000
 Max.   :185                      Max.   :8.000

   OptPest            ScreenTime        Sleep          PhysActive     HrsHomework     SpendTime1
 Length:185         Min.   : 0.000   Min.   : 2.000   Min.   : 0.00   Min.   : 0.00   Length:185
 Class :character   1st Qu.: 1.000   1st Qu.: 8.000   1st Qu.: 6.00   1st Qu.: 1.00   Class :character
 Mode  :character   Median : 3.000   Median : 9.000   Median : 9.00   Median : 3.00   Mode  :character
                    Mean   : 2.997   Mean   : 8.641   Mean   :11.52   Mean   : 4.17
                    3rd Qu.: 4.000   3rd Qu.:10.000   3rd Qu.:12.00   3rd Qu.: 6.00
                    Max.   :18.000   Max.   :12.000   Max.   :82.00   Max.   :35.00
                                                      NA's   :1
  SpendTime2           Self1              Self2             Career           Superpower
 Length:185         Length:185         Length:185         Length:185         Length:185
 Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

In contrast to other programming languages like C and java in R, the variables are not declared as some data type. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable. There are many types of R-objects. The frequently used ones are –

R Objects:-

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

Creating and Manipulating Objects:-

Vectors:-

R programming, the very basic data types are the R-objects called vectors which hold elements of different classes as shown above. Please note in R the number of classes is not confined to only the above six types. For example, we can use many atomic vectors and create an array whose class will become array.

When you want to create vector with more than one element, you should use c() function which means to combine the elements into a vector.

```
# Create a vector.
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))
```

When we execute the above code, it produces the following result −

```
[1] "red"    "green"  "yellow"
[1] "character"
```

R has five basic or “atomic” classes of objects:

• character

• numeric (real numbers)

• integer

• complex

• logical (True/False)

Lists:

A list allows you to store a variety of objects.

```
> mylist <- list(x, y, z, gender, mydata)
> mylist
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] 1 3 5 7 9

[[3]]
[1] 1 2 5 4 7

[[4]]
[1] "m" "f" "m" "m" "f"

[[5]]
  x y z gender
1 1 1 1      m
2 2 3 2      f
3 3 5 5      m
4 4 7 4      m
5 5 9 7      f
```

You can use subscripts to select the specific component of the list.

```
> mylist[[3]]
[1] 1 2 5 4 7
```

> x <- list(1:3, TRUE, "Hello", list(1:2, 5))

Here x has 4 elements: a numeric vector, a logical, a string and another list.

We can select an entry of x with double square brackets:

> x[[3]]

[1] "Hello"

To get a sub-list, use single brackets:

> x[c(1,3)]

[[1]]

[1] 1 2 3

[[2]]

[1] "Hello"

Notice the difference between x[[3]] and x[3].

We can also name some or all of the entries in our list, by supplying argument names to list().

Matrices:-

Matrices are much used in statistics, and so play an important role in R. To create a matrix use the function matrix(), specifying elements by column first:

> matrix(1:12, nrow=3, ncol=4)

[,1] [,2] [,3] [,4]

[1,] 1 4 7 10

[2,] 2 5 8 11

[3,] 3 6 9 12

This is called column-major order. Of course, we need only give one of the dimensions:

> matrix(1:12, nrow=3)

unless we want vector recycling to help us:

> matrix(1:3, nrow=3, ncol=4)

[,1] [,2] [,3] [,4]

[1,] 1 1 1 1

[2,] 2 2 2 2

[3,] 3 3 3 3

Sometimes it’s useful to specify the elements by row first

> matrix(1:12, nrow=3, byrow=TRUE)

There are special functions for constructing certain matrices:

> diag(3)

[,1] [,2] [,3]

[1,] 1 0 0

[2,] 0 1 0

[3,] 0 0 1

> diag(1:3)

[,1] [,2] [,3]

[1,] 1 0 0

[2,] 0 2 0

[3,] 0 0 3

> 1:5 %o% 1:5

[,1] [,2] [,3] [,4] [,5]

[1,] 1 2 3 4 5

[2,] 2 4 6 8 10

[3,] 3 6 9 12 15

[4,] 4 8 12 16 20

[5,] 5 10 15 20 25

The last operator performs an outer product, so it creates a matrix with (i, j)-th entry xiyj . The function outer() generalizes this to any function f on two arguments, to create a matrix with entries f(xi , yj). (More on functions later.)

> outer(1:3, 1:4, "+")

[,1] [,2] [,3] [,4]

[1,] 2 3 4 5

[2,] 3 4 5 6

[3,] 4 5 6 7

Array:

If we have a data set consisting of more than two pieces of categorical informationabout each subject, then a matrix is not sufficient. The generalization of matrices to higher

dimensions is the array. Arrays are defined much like matrices, with a call to the array() command. Here is a 2 × 3 × 3 array:

> arr = array(1:18, dim=c(2,3,3))

> arr

, , 1

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

, , 2

[,1] [,2] [,3]

[1,] 7 9 11

[2,] 8 10 12

, , 3

[,1] [,2] [,3]

[1,] 13 15 17

[2,] 14 16 18

Each 2-dimensional slice defined by the last co-ordinate of the array is shown as a 2 × 3 matrix. Note that we no longer specify the number of rows and columns separately, but use a single vector dim whose length is the number of dimensions. You can recover this vector with the dim() function.

> dim(arr)

[1] 2 3 3

Note that a 2-dimensional array is identical to a matrix. Arrays can be

subsetted and modified in exactly the same way as a matrix, only using the appropriate number of co-ordinates:

```
> arr[1,2,3]
[1] 15
> arr[,2,]
```

[,1] [,2] [,3]

[1,] 3 9 15

[2,] 4 10 16

> arr[1,1,] = c(0,-1,-2) # change some values

> arr[,,1,drop=FALSE]

, , 1

[,1] [,2] [,3]

[1,] 0 3 5

[2,] 2 4 6

Factors:-

R has a special data structure to store categorical variables. It tells R that a variable is nominal or ordinal by making it a factor.

Simplest form of the factor function :

```
> gender <- c(1,2,1,2,1,2,1,2)
> gender <- factor(gender)
```

Ideal form of the factor function :

```
> gender <- c(1,2,1,2,1,2,1,2)
>
> gender <- factor(gender,
+ levels = c(1,2),
+ labels = c("male","female"))

> table(gender)
gender
  male female
     4      4
```

The factor function has three parameters:
1. Vector Name
2. Values (Optional)
3. Value labels (Optional)

Convert a column "x" to factor

data$x = as.factor(data$x)

Data Frames:-

A data frame is a table or a two-dimensional array-like structure in which each columncontains values of one variable and each row contains one set of values from each column.

- Data frames are tabular data objects.
- A Data frame is a list of vector of equal length.
- Data frame in R is used for storing data tables.

Characteristics of a data frame:

1. The column names should be non-empty.
2. The row names should be unique.
3. The data stored in a data frame can be of numeric, factor or character type.

Create Data Frame

```
# Create the data frame.
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
   salary = c(623.3,515.2,611.0,729.0,843.25),
      start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-
11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)# Print the data frame.
print(emp.data)
```

When we execute the above code, it produces the following result –

emp_id	emp_name	salary	start_date
1	Rick	623.30	2012-01-01
2	Dan	515.20	2013-09-23
3	Michelle	611.00	2014-11-15
4	Ryan	729.00	2014-05-11
5	Gary	843.25	2015-03-27

```
# Create the data frame.
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name =
   c("Rick","Dan","Michelle","Ryan","Gary"),salary =
   c(623.3,515.2,611.0,729.0,843.25),

   start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)
# Print the summary.
print(summary(emp.data))
```

Get the Structure of the Data Frame

The structure of the data frame can be seen by using str() function.

```
# Create the data
frame.emp.data <-
data.frame(
   emp_id = c (1:5),
   emp_name =
   c("Rick","Dan","Michelle","Ryan","Gary"),salary
   = c(623.3,515.2,611.0,729.0,843.25),

   start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)
```

When we execute the above code, it produces the following result –

```
'data.frame':   5 obs. of  4 variables:
 $ emp_id    : int  1 2 3 4 5
 $ emp_name  : chr  "Rick" "Dan" "Michelle" "Ryan" ...
 $ salary    : num  623 515 611 729 843
 $ start_date: Date, format: "2012-01-01" "2013-09-23" "2014-11-15" "2014-05-11" ...
```

Summary of Data in Data Frame

The statistical summary and nature of the data can be obtained by applying summary()function.

When we execute the above code, it produces the following result −

```
 Median :3   Mode  :character   Median :623.3   Median :2014-05-11
 Mean   :3                      Mean   :664.4   Mean   :2014-01-14
 3rd Qu.:4                      3rd Qu.:729.0   3rd Qu.:2014-11-15
 Max.   :5                      Max.   :843.2   Max.   :2015-03-27
```

Expand Data Frame

A data frame can be expanded by adding columns and rows.0

1.Add Column:-Just add the column vector using a new column name.

2.Add Row:-To add more rows permanently to an existing data frame, we need to bring in the new rowsin the same structure as the existing data frame and use the rbind() function.

```
# Add the "dept" coulmn.

emp.data$dept <- c("IT","Operations","IT","HR","Finance")

v <- emp.data

print(v)
```

When we execute the above code, it produces the following result −

	emp_id	emp_name	salary	start_date	dept
1	1	Rick	623.30	2012-01-01	IT
2	2	Dan	515.20	2013-09-23	Operations
3	3	Michelle	611.00	2014-11-15	IT
4	4	Ryan	729.00	2014-05-11	HR
5	5	Gary	843.25	2015-03-27	Finance

.