Data Analytics

__/__/____

★ **Data Analysis :-**

- Data Analysis is a process of obtaining raw data and converting it into information useful for decision-making by users.

★ **Data Analytics :-**

- It is the science of examining raw data with the purpose of drawing conclusions about that information.

★ **Applications of ~~Dat~~ Analytics :-**

- In commercial industries, to enable organizations to make more informed business decicions and by scientists
- By researches, to verify or disaprove scientific models, theories and hypotheses.

★ **Analysis vs Analytics :-**

- Data analytics is a broader term and includes data analysis as necessary subcomponent.
- Analytics defines the science behind the analysis.
- The science means understanding the cognitive proces an analyst uses to understand problems and explore data in meaningful ways.
- Analytics also include data extract, transform, and load specific tools, techniques, and methods; and how to succesfully communicate results.

* Data Science :-

- Dealing with unstructured and structured data, Data Science is a field that comprises of everything that related to data cleansing, preparation, and analysis.
- Involves in creation of new algorithms.

* Data Vs ~~Algorithms~~ Information :-

- Data are the facts or details from which information is derived.
- Individual pieces of data are rarely useful alone.
- For data to become information, data needs to be put into context.

Data	Information
• Data means raw facts gathered about someone or something, which is bare and random.	• Facts, concerning a particular event or subject, which are refined by processing is called information.
• It is just text and numbers.	• It is refined data.
• Records and observations.	• Analysis
• Unorganized	• Organized.
• May or may not be useful.	• Always
• No	• Yes
• Does not depend on information	• Without data, information cannot be procured.

__/__/__

* Characteristics of data :-

1. Accuracy and Precision
2. Legitimacy and Validity
3. Reliability and Consistency
4. Timeliness and Relevance
5. Completeness and Comprehensiveness
6. Availability and Accessibility
7. Granularity and Uniqueness

1. Accuracy and Precision :- This characteristic refers to the exactness of the data.

2. Legitimacy and Validity :- Requirements governing data set the boundaries of this's characteristic.

3. Reliability and Consistency :- Regardless of what source collected the data or where it resides, it cannot contradict a value residing in a different source or collected by a different system.

4. Timeliness and Relevance :- Data collected or too late could misinterpret a situation and drive inaccurate decisions.

5. Completeness and Comprehensiveness :-
Incomplete data is as dangerous as inaccurate data.

6. Availability and Accessibility :- This presumes that the data exists and is available for access to be granted.

7. Granularity and Uniqueness :- The level of detail at which data is collected is important, because confusion and inaccurate decisions can otherwise occur.

★ Steps involved in Data Analysis :-

1. Data Collection :- Identify good sources.
2. Data Cleaning :- Remove noise.
3. Data Analysis :- Apply algorithms
4. Interpretation :- Explain results and take actions.

★ Data Munging :-

- Required for improve the quality of gathered data.
- Involves cleaning and transformation of messy data available with us.
- Also reffered as Data Wrangling.

★ Reasons for Noise in data :-

→ Lots of potentially incorrect data.
- Faulty instruments
- Human or computer error.
- Transmission error.

eg. Occupation = " " (missing data)
Salary = "-10" (an error)

→ **Binning :-**

- first sort data and partition into (equal-frequency) bins
- then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.

→ **Regression :-**

- Smooth by fitting the data into regression functions.

→ **Clustering :-**

- detect and remove outliers.

→ **Combined computer and human inspection :-**

- detect suspicious values and check by human (e.g.; deal with possible outliers)

* **Data Transformation :-**

- Transformation is mapping the Data to a new space.

eg :-

- Fourier Transform
- Wavelet Transform

* **Data Sampling :-**

- Obtaining a small sample S to represent the whole data set N.

* Types of Sampling :-

→ Simple Random Sampling :-

- There is an equal probability of selecting any particular item.

→ Stratified Sampling :-

- Partition the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data).
- Used in conjunction with skewed data.

* Types of Sampling :-

Stratified Sampling

* Sampling without replacement :-

- Once an object is selected, it is removed from the population.

* Sampling with replacement :-

- A selected object is not removed from the population

__/__/____

* Data Cleaning

Problem Solving

Data	% Fat
:	:
:	:

Q) Mean, Median, Mode, Range, Standard Deviation, Variance
Find the above values for age and % fat.

Modes :-
One mode - unimodal
Two modes - bimodal
Three modes - trimodal
more than one mode - multimodal

$$SD = \sqrt{\frac{\Sigma(x-\bar{x})^2}{n-1}}$$

$$SD = \sqrt{\text{variance}}$$

* Binning :-

Q) Perform sm

23, 23, 27, 29, 39, 41, 47, 49, 50, 52, 54, 54, 56, 57, 58, 60, 61

Perform smoothing operation using
→ Smoothing by bin means
→ Smoothing by bin median
→ Smoothing by bin boundaries
using a bin depth of 3.

Ans. Divide the elements into bins of depth 3.
↓
(frequency).

Bin 1 :- 23, 23, 27
Bin 2 :- 27, 39, 41
Bin 3 :- 47, 49, 50
Bin 4 :- 52, 54, 54
Bin 5 :- 56, 57, 58
Bin 6 :- 58, 60, 61

- **Smoothing by Bin Means :-**

Bin 1 :- 24, 24, 24 {Add the 3 elements of B1 and ÷ by 3}
Bin 2 :- 36, 36, 36
Bin 3 :- 49, 49, 49
Bin 4 :- 53, 53, 53
Bin 5 :- 57, 57, 57
Bin 6 :- 60, 60, 60

- **Smoothing by Bin Medians :-**

Bin 1 :- 23, 23, 23 {Find the median of the 3 elements of B1}
Bin 2 :- 39, 39, 39
Bin 3 : 49, 49, 49
Bin 4 :- 54, 54, 54
Bin 5 :- 57, 57, 57
Bin 6 :- 60, 60, 60

- **Smoothing by Bin Boundaries :-**

Bin 1 :- 23, 23, 27 {Find the middle element & close to which min/max value and change it to that value}.
Bin 2 :- 27, 41, 41
Bin 3 :- 47, 50, 50
Bin 4 :- 52, 54, 54
Bin 5 :- 56, 58, 58 or 56, 56, 58
Bin 6 :- 58, 61, 61

-) Equal width Partitioning :

Width 10

B1 - 1 and 10 = 5, 10
B2 - 11 and 20 = 13, 15, 11
~~21 and 30 = 35~~
B3 - 31 and 40 = ~~25~~ 35
B4 - 41 and 50 = 50
B5 - 61 and 60 = ~~58~~ 55
~~61 and 70 =~~
B6 - 71 and 80 = 72
~~81 and 90~~
B7 - 91 and 100 = 92

} either do by mean, by median, by boundary.

} 1 values no smooth.

Q) Find the missing values. [find by app. means of mean, median or mode

74
68
56
57
60
? → (63).
68
71
53
61
57
68
68
~~50~~ 60
57.

~~74+71~~

$$\frac{74+68+56+57+60+68+71+53+61+57+68+68}{14}$$

$$\frac{60+57}{14}$$

$$= 62.71 \simeq (63)$$

Data Sampling :-

91 95 103 115 112 101 102 97 115 99 102 97
100 107 116 107 110 113 95 93

① Simple Random Sampling w/o replacement. (size 5)

randomly :- 102 102 91 110 112 → Sample size - 15

Ways :- $^{20}C_5$.

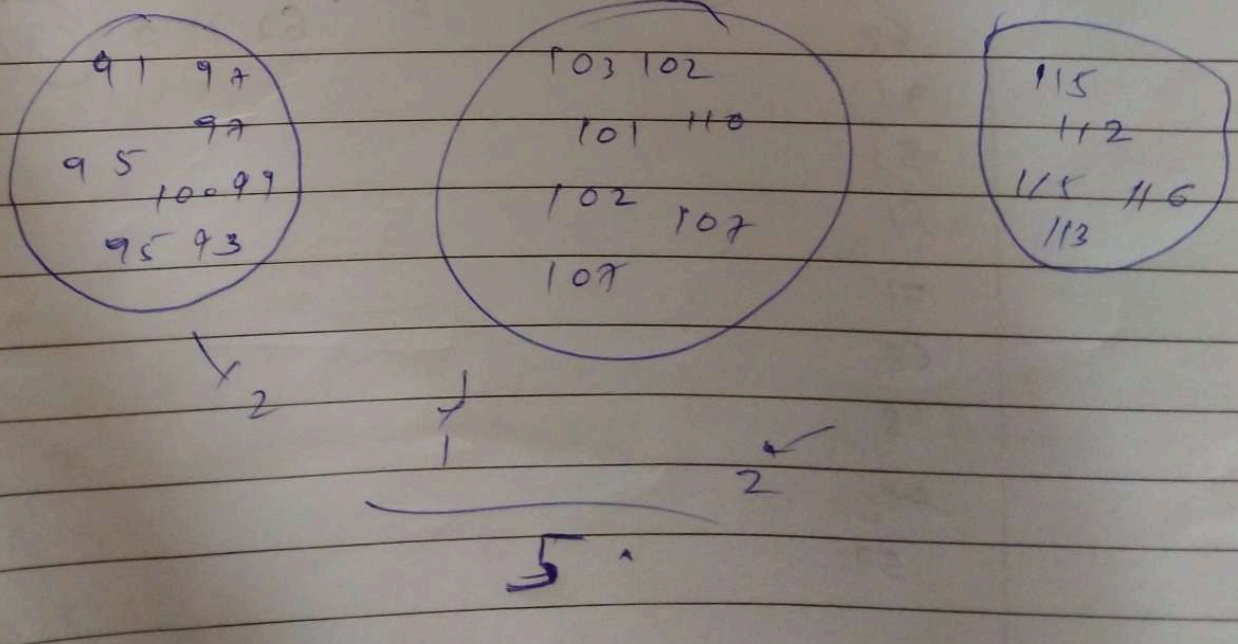
② Simple Random sampling with Replacement

103 103 103 103 103 → sample size - 5

$|P| = 20 \rightarrow 20 \times 20 \times 20 \times 20 \times 20$.

③ stratified sampling → heterogeneous sampling.

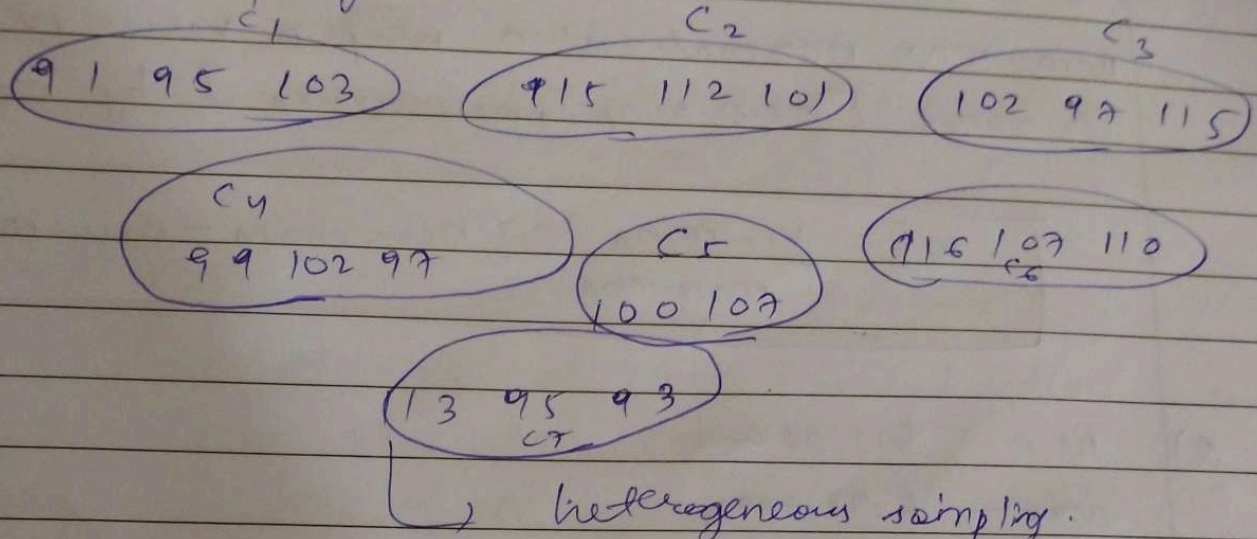
S1 Score ≤ 100 | 101 ≤ Score ≤ 110 | Score ≥ 100



retrieve sample of size - 5 from stratified sample.

91 97 103 115 112.

(4) Cluster sampling.



↳ heterogeneous sampling.

Out of 7 cluster any 5 can be chosen

↓

clusters.

→ sampling C_1 C_2 C_3 C_4 C_1

* Min-max Normalization :-

Min_A - minimum value of an attribute
Max_A - maximum value of an attribute, A.

Min-max normalization maps a value, v_i of A to v_i' in range $[new\_min_A, new\_max_A]$ by computing

$$v_i' = \frac{v_i - min_A}{max_A - min_A} (new\_max_A - new\_min_A) + new\_min_A$$

Q) Min = $12,000
Max = $98,000
Map income to the range [0.0, 1.0]
By min-max normalization, a value of $73,000 for income is transformed to :—

$$\frac{73,000 - 12,000}{98,000 - 12,000} (1.0 - 0.0) + 0 = 0.716$$

* Z-score Normalization :- (zero mean)

The value of an attribute, A are normalized based on the mean and standard deviation of A.

$$V_i' = \frac{V_i - \bar{A}}{\sigma A}$$

$\bar{A}$ = Mean
σA = S.D.

Q) Mean = $54,000
SD = $16,000
Z com, 73,600.

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}$$

$$= \frac{73,600 - 54,000}{16,000}$$

$$= 1.225$$

* Decimal Scale Normalization :-

The number of decimal points moved depends on the maximum absolute value of A.
A value of v_i, of A is normalized to v_i' by computing :-

$$v_i' = \frac{v_i}{10^j},$$

where j is the smallest integer such that $\max(|v_i'|) < 1$

Q) 'A' range from -986 to 917. The max. absolute value of A is 986. divide each value by 1000 (ie. $j = 3$) (i.e. $j = 3$) so that -986 normalizes to -0.986 and 917 Normalizes to 0.917.

2) Use to normalize :-

200, 300, 400, 600, 1000.

(a) min-max normalization by setting min = 0 and max = 1

New min A = 0

New-max A = 1

min A = 200

max A = 1000

$$v_i' = \frac{v_i - \text{min}_A}{\text{max}_A - \text{min}_A} (\text{new-max}_A - \text{new-min}_A) + \text{new-min}_A$$

$$v_i' = \frac{200 - 200}{1000 - 200} \times (1 - 0) + 0$$

(b) Z-score normalization

$$v_i' = \frac{v_i - \bar{A}}{\sigma A}$$

$\bar{A} = 500$

$\sigma A = 316.22$

$v_i = 200$

$$v_i' = \frac{200 - 500}{316.22} = -0.94$$

(c) Decimal Scaling :-

$$v_i' = \frac{v_i}{10^j}$$

$$v_i = \frac{200}{1000} = 0.2$$

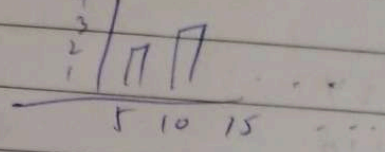
* Histogram :-

Buckets are of 3 types:-

1. Singleton
2. Equal Width
3. Equal frequency.

1, 1, 5, 5 . . . . . .

The no. has come for the no. of times :→ histogram



* Data Wrangling in Python :-

- Data wrangling is the proces of gathering, collecting, and transforming Raw data into another format for better understanding, decision-making, accessing, and analysis in less time. Data wrangling is also known as Data Munging.

→ Data Wrangling in Python deals with the below functionalities :-

1. Data exploration.
2. Dealing with missing values.
3. Reshaping data.
4. Filtering data.
5. ~~Etc~~

R-Programming

★ Data scraping :-

- Data scraping refers to the process of extracting data for

Q)
```
x <- c(1,2)
y <- c(3,NA)
x+y
```

O/P [1] 4 NA

Q)
```
x <- 2:4
y <- 6:10
x+y
```

O/P [1] 8 10 12 11 13

Q) Vector
```
x <- c(4,3,5,6,11,19)
x[x<5]+x[x<6]
```

[1] 8 6 9

In x[x<5]+x[x<6]:

longer object length is not a multiple of shorter object length.

Q) repeat_each <- ~~eq~~ rep (c(1,2,3), each = 3)
repeat_each

Ans- [1] 1 1 1 2 2 2 3 3 3

Q) x <- rep(c(1,2,3), times = 3)
x

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

Q) x <- rep(c(1,2,2), times = c(5,2,1))
x

[1] 1 1 1 1 1 2 2 3

Q) v <- seq(from 1, to = 3.3 ; by = 0.4
v
v[c(-1,-2)]

[1] 1.0 1.4 1.8 2.2 2.6 3.0
[1] 1.8 2.2 2.6 3.0

Q) b ← vector (2, 3.5, 4, 5.5)
b

Ans- Error in vector (2, 3.5, 4, 5.5)
unused arguments (4, 5.5)

Q) sum (0:9+1) - sum (0:(9+1))

[1] 0

Varname can start with letter or (.)

Q) v<-sample(-10:10, 10)
print(v)
print(max(v) - min(v)) (10 - (-9) = 19.
print(mean(v))
print(mean(v, trim=0.2))
print(median(v))

o/p [1] -9 -4 5 -5 10 -2 7 3 0 -8 [any 10 random nos b/w the range]

[1] 19

[1] 0.5

[1]

[1] 1.5

Q) v<- 1:10
sum(v[-1]) (1+10+(1+2...+10) - 1)
length(v[-5:-8])
v[seq(-2, -8, -2)]
v[c(-2, -4, -6, -8)]

[1] 54
[1] 6
[1] 1 3 5 7 9 10
[1] 1 3 5 7 9 10

Q) v1 <- 1:5
v2 <- v1[c(length(v1):1)]
sum(v2 - v1)

O/P [1] 0

Q) v <- sample(-5:5, 5)
v
v1 <- sort(v)
v1
v1[length(v1) - v1[1]]

O/P [1] -4 1 0 -5 -1

[1] -5 -4 -3 -2 -1

[1] 6

Q) v <- seq(-5, 5, 2)
v
append(v, 10)
append(v, 10, 3) [Add 10 at index position 3]

O/P Seq -5 -3 -1 1 3 5

[1] -5 -3 -1 1 3 5 10

[1] -5 -3 -1 10 1 3 5

Q) v <- c(1,2,3,4,5,6)
v <- v %% 2 → remainder.
sum(v)

O/P [1] 3

Q) x <- c(1,1,4,5,4,6)
unique(x) [once ~~=~~ value]

O/P [1] 1,4,5,6

Q) x <- c(1,1,4,5,4,6)
duplicated(x) [if repeated → True, otherwise false]
x[duplicated(x)]
x [!duplicated(x)]

O/P [1] FALSE TRUE FALSE FALSE TRUE FALSE

[1] 1 4

[1] 1 4 5 6

Q) m <- matrix(seq(from=10, to=18, by=2), nrow=3, ncol=3)

m
sum(m[1,])

	[,1]	[,2]	[,3]
[1,]	10	16	12
[2,]	12	18	14
[3,]	14	10	16

P [1] 38 [Add column 1].

Matrix :-

Q) x <- 11:13
y <- 10:12 → rows.
dim(t(rbind(x, y)))
dimension → (row, column)

O/P [1] 3 2

Q) a <- matrix(1:2, nrow=2, ncol=2)
sum(a[1,]+a[,1])

O/P [1] 5

Q) mat1 <- matrix(1:3, 3, 3, byrow=T)
→ True - Row wise
False - Column wise
mat2 <- t(matrix(1:3, 3, 3))
print(sum(mat1-mat2))

O/P [1] 0

Q) x <- matrix(1:7, nrow=3)
x

1 → 7 isn't a multiple of 3.

Q) dim(t(matrix(1:10)))

[1] 1 10
Col. Row

Q) m <- matrix(1:6, byrow=TRUE)
m[,1]+m[1,]
↓ 1st column ↓ 1st row

[1] 2 3 4 5 6 7

Q) m1 <- matrix(1:6, byrow = TRUE)
m2 <- matrix(1:6, byrow = FALSE)
m1 - m2

```
     [,1]
[1,]    0
[2,]    0
[3,]    0
[4,]    0
[5,]    0
[6,]    0
```

Q) a <- c('abc', 'def', 'ghi', 'jkl')
b <- c(11, 23, 355, 426)
mat <- matrix(a, 2, 2)
mat

```
     [,1]  [,2]
[1,] "abc" "ghi"
[2,] "def" "jkl"
```

Q) x <- matrix(c(1:7), 3,)
y <- matrix(c(1:7),, 3, byrow = T)
x+y

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    6   10    7
[3,]   10    7    4
```

logical - True, False
integer = 36L
Complex = 5+2i
numeric = -24, 36.
Char = ' ' " "

Data Types :-

Q) x <- c ("a", "b", "TRUE")
as.logical(x)

[1] NA NA TRUE

Q) x <- charToRaw("a"), charToRaw("b"))
as.logical(x)

[1] TRUE TRUE

Q) a <- c(48, 54.5, 33i)
class(a)

[1] "complex"
[1] 48.0+0i 54.5+0i 0.0+33i

Q) as.logical(sum(as.numeric(F)+as.numeric(T))

[1] TRUE

Q) dim(numeric(6)) 0 0 0 0 0 0

[1] NULL

Q) x <- c(10L, 10, 10i, '10')
class(x)

[1] "10" "10" "0+10i" "10"
[1] "character"

Q) print (sum (complex (500)))

[1] 0+0i

Q) print (complex (2))

[1] 0+0i 0+0i

Q) c <- c (TRUE, 24, FALSE, -34, 0)
c <- as.character (c)
c

[1] "1" "24" "0" "-34" "0"

Q) c <- c (TRUE, FALSE)
c <- as.character (c)
c
d <- c (TRUE, 24, FALSE, -34, 0)
d <- as.character (d)
d

[1] "TRUE" "FALSE" (when single T=T, F=F.
[1] "1" "24" "0" "-34" "0" [when with no, T→1, F=0]

Q) cd <- c ("TRUE", "FALSE", "FALSE")
class (cd)

[1] character

AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Q) in logical - 1 → True, 0 → False.

Q) x <- c(0,1,0,1,0,1)
y <- c(1,0,1,0,1,0)

print(sum(as.numeric(x && y) + as.numeric(x & y)))

[1] 0

Q) x <- c(10, 20, 30, 40)
y <- c(1L, 2L, 3L, 4L)
x && y
x & y
x || y
x | y

[1] TRUE
[1] TRUE TRUE TRUE TRUE

[1] TRUE
[1] TRUE TRUE TRUE TRUE

Q) x <- c(1,0,1,0)
y <- c(0L, 1L, 0L, 1L)
x && y + x & y

[1] FALSE TRUE FALSE TRUE

array (2, 3, 4)
↓ ↓ ↓
nrow ncol no. of matrices

__/__/__

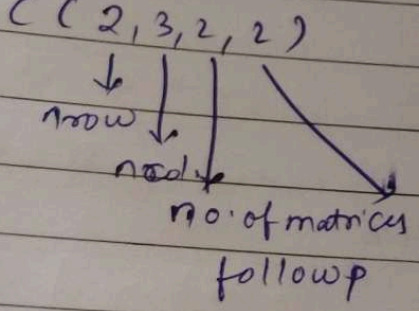
Array :

Q) thisarray <- c(1:10)
multiarray <- array(thisarray, dim = c(4,3,2))
multiarray [2,3,1] → 1st matrix
multiarray [2,3,2] → 2nd matrix

[1] 10
[2] 4

Q) thisarray <- c(1:18)
multiarray <- array(thisarray, dim = c(4,3,2))
multiarray [,,1]
↓
print matrix1.

Q) multiarray [,1,] → col.1 + col.2
↓ ↓
m1 m2

Q)


Q) c(2,3,2,2))
matriarray [2,3,1,]

List

Q) n = c(2,3,5)
s = c("aa", "bb", "cc", "dd", "ee")
b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
lst <- list(n, s, b)
lst

[[1]]
[1] 2 3 5

[[2]]
[1] "aa" "bb" "cc" "dd" "ee"

[[3]]
[1] TRUE FALSE TRUE FALSE FALSE

- ~~And~~ In list index position starts with 1.
  lst[[2]] → second list
  lst[[2]][4] → 4th element of second list.

Q) x <- c(5,6)
y <- matrix(1:4, nrow=2, ncol=2)
l <- list(x,y)
l[[2]][1,] → 1st row of 2nd list.

~~6~~ [1] 1 3

[[2]] [1,2] → 2nd matrix, 1st row 2nd column

10+(8) 18
10+ - (8)   8x = 20
x = 30/8 = 2.5 / __ / __
10+ 8x = 30

Q)
```
f <- function(x)
{
  if (x = 2)
  {
    return (2)
  }
  else
  {
    print ('+')
    f(x-1)
  }
}
print(f(6))
```

[1] "+"
[2] "+"
[1] "+"
[1] "+"
[1] 2

Q)
```
f <- function(x)
{
  if (x>0)
  {
  return (x + f(x-2))
  }
  else
  {
  return (0)
  }
}
Print( f(10))
```

Q) 1 1 1 1 1
2 2 2 2
3 3 3
4 4
5

```
stars=NULL
k<-5
for (i in 1:5)
{
  for (j in 1:k)
  {
    stars = c(stars,i)
  }
  print(stars)
  k<- k-1
  stars=NULL
}
```

Q) 1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

```
stars=NULL
for (i in 1:5){
  for (j in 1:i){
    stars = c(stars,i)
  }
  print(stars)
  stars=NULL
```

Q) 1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```
stars = NULL
k <- 1
for (i in 1:5) {
  for (j in 1:k) {
    stars = c(stars, j)
  }
  print(stars)
  k <- k + 1
  stars = NULL
}
```

Q) 1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```
stars = NULL
k <- 5
for (i in 1:5) {
  for (j in 1:k) {
    stars = c(stars, j)
  }
  print(stars)
  k <- k - 1
  stars = NULL
```

Q) 1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1

```
stars = NULL
for (i in 1:5){
   for (j in 1:5) {
    { if (i==j)
      { stars = c(stars,1) }
      else
      {
       stars = c(stars,0)
      }
    }
    print (stars)
    stars=NULL
  }
```

Q) 1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1

if(i≥j)

Q) 1 1 1 1 1
0 1 1 1 1
0 0 1 1 1
0 0 0 1 1
0 0 0 0 1

if(i≤j)

Q)

```
0
1 1
0 0 0
1 1 1 1
0 0 0 0 0
```

```
stars = NULL
for (i in 1:5){
  for (j in 1:i){
    if ((i%%2) == 0){
      stars = c(stars, 1) }
    else
    {
      stars c(stars, 0)
    }
  }
print(stars)
stars = NULL
}
```

Data Frame :-

```
x <- c('one', 'two', 'three', 'four')
y <- c(1, 2, 3, 4)
z <- c(4.5, 8.9, 3.3, 4.5)
df <- data.frame(x, y, z)
df
```

	x	y	z
1	one	1	4.5
2	two	2	8.9
3	three	3	3.3
4	four	4	4.5

class = "data.frame"

df[1]

one
two
three
four

df['x']

one
two
three
four.

df $x

one to thre four
one
one
one

class df$x → "character".

df[, c(2,3)]

print col.2, col.3.

df[c(2,3)]

col2 col3.

q1
```
x <- c('one', 'two', 'three', 'four')
y <- c(1,2,3,4)
z <- c(4.5, 8.9, 3.3, 4.5)
df <- data.frame(x,y,z)
df[c(1,2)]
```
⟹ 1st and 2nd values of df.

	x	y	z
1	one	1	4.5
2	two	2	8.9

q1
```
x <- c ...
y <- c ...
z <- c ...
df <- ...
df
df[c(seq(1, nrow(df), 2)),]
```

q)
```
x
y
z
df
rownames(df) <- c('first', 'second', 'third', 'four')
df
```

	x	y	z
first			
second			
third			
four			

Q1

```
df1 = data.frame(student_Id = c(1:4), subject = c(
      "OS", "DBMS", "Maths", "Java"))

df2 = data.frame(student_Id = c(5:7), subject =
      c("C++", "Python", "ANN"))

df3 <- rbind(df1, df2)

df3
```

	student_Id	subject
1	1	OS
2	2	DBMS
3	3	Maths
4	4	Java
5	5	C++
6	6	Python
7	7	ANN

Q1

```
df <- data.frame(student_Id = c(1:4), subject =
      c("OS", "DBMS", "Maths", "Java"))

newrow <- c(5, "C")
df[nrow(df)+1,] <- newrow
df
```

1	1	OS
2	2	DBMS
3	3	Maths
4	4	Java
5	5	C

Q) df <- data.frame(Student Id =)
df $ GPA <- c(8.5, 9.2, 8.9, 9.3)
df

St. Id	subject	GPA
1	OS	6.5
2	DBMS	9.2
3	Maths	8.9
4	Java	9.3

or df['GPA'] <- c(8.9, 9.2, 8.9, 9.3)

or GPA <- c(8.9, 9.2, 9.3, 8.9)
df1 <- cbind(df, GPA)

df <- data.frame(StudentId = c(1:4),
Subject = c("OS", "DBMS", "Maths", "Java"))
GPA <- c(8.5, 9.2, 8.9, 9.3)
df1 <- cbind(df[1], GPA, df[2])

df2 <- cb

Student Id	GPA	Subject

Q1) Program to find the HCF of two number.

```
hcf <- function (x, y) {
  if (x>y)
  {
   smaller = y
  }
   else
   {
    smaller = x
   }
   for ( i in 1:smaller)
   {
    if ((x%%i == 0) && (y%%i==0))
    {
     hcf = i
    }
   }
   return(hcf)
   }
   n1 = as.integer (readline (prompt = "Enter first no"))
   n2 = as.integer (readline (prompt = "Enter s. c"))
   print (paste ("The hcf of", n1, "and", n2 "is"
                                    hcf(n1,n2)))
```

Q) Palindrome

```
is_palindrome <- function(num){
  num_str <- as.character(num)
  reversed_num_str <- rev(strsplit(num_str, "")[[1]])
  if (identical(num_str, reversed_num_str)){
    return(True)
  }
  else
  {
  return(FALSE)
  }
}
```

Q) Perfect number :

```
is_perfect <- function(num)
{
if (num<=0)
{
return(FALSE)
}
divisors <- numeric(0)
for(i in 1:(num/2)){
  if(num %% i ==0){
    divisors <- c(divisors, i)
  }
}
```

```
if (sum(divisors) == num)
 {
  return(TRUE)
 }
 else
 {
 return (FALSE)
 }
}
```

Q1 Fibonacci

```
fibonacci <- function(n)
{
 fib<-c(0,1)

 for ( i in 3:n) {
     next_term <- fib[i-1] + fib[i-2]
      fib[i] <- c(fib, next_term)
     }
     return (fib[1:n])
 }
```