UNIT-1

Two Marks Questions:

1)Write any 2 applications of R Programming in Real World

- Data science
- Statistical computing

2) How to get help on any function of R? Give an example.

To get help on any function of R, type help(function-name) in R prompt. For example, if we need help on "if" logic, type,
> help("if")
then, help lines for the "if" statement is printed.

3) What is a comment? How do we write comment in R programming?

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program. Single comment is written using # in the beginning of the statement as follows:
My first program in R Programming
R does not support multi-line comments but you can perform a trick which is something as follows:
if(FALSE)
{
"This is a demo for multi-line comments and
it should be put inside either a single of double quote"
}

4) What are reserved words in R programming? Give 2 examples.

Reserved words in R programming are a set of words that have special meaning and cannot be used as an identifier.

if	else	repeat	while	function
for	in	next	break	TRUE

5) What are variables and constants in R programming?

Variables are used to store data, whose value can be changed according to our need. Constants, as the name suggests, are entities whose value cannot be altered.

6) What is the purpose of typeof() function in R? Give an example.

typeof() function in R Language is used to return the types of data used as the arguments.
> typeof(5)
[1] "double"

> typeof(5L)
[1] "integer"

7) Write the output of the following constants. a) 0xff b) 0XF + 1

```
> 0xff
[1] 255

> 0XF + 1
[1] 16
```

8) What is the output of the following?

a) fruits <- c("banana", "apple", "orange")

length(fruits)

[1] 3

b) v <- c(3,8,4,5,0,11, -9, 304)

sort.result <- sort(v)

print(sort.result)

[1] -9 0 3 4 5 8 11 304

9) What is the output of the following?

a) v <- c("Red","Blue","yellow","violet")

sort.result <- sort(v)

print(sort.result)

[1] "Blue" "Red" "violet" "yellow"

b) revsort.result <- sort(v, decreasing = TRUE)

print(revsort.result)

the vector will be printed in reverse order
[1] "yellow" "violet" "Red" "Blue"

10) How do you merge two lists?

You can merge many lists into one list by placing all the lists inside one list() function.

```
# Create two lists.
list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")
# Merge the two lists.
merged.list <- c(list1,list2)
# Print the merged list.
print(merged.list)
```

11) How do you convert list to a vector?

A list can be converted to a vector so that the elements of the vector can be used for further manipulation. All the arithmetic operations on vectors can be applied after the list is converted into vectors. To do this conversion, we use the **unlist()** function. It takes the list as input and produces a vector.

```
# Create lists.
list1 <- list(1:5)
print(list1)
list2 <-list(10:14)
print(list2)
# Convert the lists to vectors.
v1 <- unlist(list1)
v2 <- unlist(list2)
print(v1)
print(v2)
```

12) Give an example to remove a rows and a columns from a Matrix.

We can use the c() function to remove rows and columns in a Matrix.

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "mango", "pineapple"), nrow = 3, ncol =2)
#Remove the first row and the first column
thismatrix <- thismatrix[-c(1), -c(1)]
thismatrix
```

Output for above code will be:

```
[1] "mango" "pineapple"
```

13) Which operator is used to check if an item exists in a matrix? Give an example.

To find out if a specified item is present in a matrix, use the %in% operator:

Example: Check if "apple" is present in the matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2) "apple" %in% thismatrix
```

Output for above code will be:

```
[1] TRUE
```

14) What is the output of the following?

```
this matrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

length(thismatrix)
```

Output for above code will be:

```
[1] 4
```

15) Name any 2 characteristics of a data frame. [any 2]

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

16) What is the use of str() function in data frame?

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

17) What is the output of the following?

Data_Frame <- data.frame (Training = c("Strength", "Stamina", "Other"), Pulse = c(100, 150, 120), Duration = c(60, 30, 45))

dim(Data_Frame)

ncol(Data_Frame)

```
> dim(Data_Frame)
[1] 3 3

> ncol(Data_Frame)
[1] 3
```

18) What is a factor? Give an example.

Factors are the data objects which are used to categorize the data and store it as levels.
Examples of factors are:
- Demography: Male/Female
- Music: Rock, Pop, Classic, Jazz
- Training: Strength, Stamina

19) What is the use of na.rm=TRUE?

You can use the argument na. rm = TRUE to exclude missing values when calculating descriptive statistics in R.

20) Differentiate nominal data with ordinal data.

Nominal data is classified without a natural order or rank, whereas ordinal data has a predetermined or natural order. Nominal level data can only be classified, while ordinal level data can be classified and ordered.

21) List the classifications of data on scales.

- Nominal scale
- Ordinal scale
- Interval scale
- Ratio scale

22) Define skewness and kurtosis.

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. Kurtosis is a measure of the combined weight of a distribution's tails relative to the center of the distribution.

23) Compare histograms with barplots.

Bar graph	Histogram

Bar graph is the graphical representation of categorical data	Bar graph is the graphical representation of grouped data in continuous manner
There is equal space between each pair of consecutive bars	There is no space between the consecutive bars
The height of the bars shows the frequency and the width gap is zero	The frequency of the data is shown by the area of rectangular bars

Long Answer Questions:

1) Explain applications of R Programming in Real World (5)

• **Data Science:** Programming languages like R give a data scientist superpower that allow them to collect data in realtime, perform statistical and predictive analysis, create visualizations and communicate actionable results to stakeholders.

• **Statistical computing:** R is the most popular programming language among statisticians. In fact, it was initially built by statisticians for statisticians. It has a rich package repository with more than 9100 packages with every statistical function you can imagine. R's expressive syntax allows researchers - even those from non computer science backgrounds to quickly import, clean and analyze data from various data sources. R also has charting capabilities, which means you can plot your data and create interesting visualizations from any dataset.

• **Machine Learning**: R has found a lot of use in predictive analytics and machine learning. It has various package for common ML tasks like linear and non-linear regression, decision trees, linear and non-linear classification and many more. Everyone from machine learning enthusiasts to researchers use R to implement machine learning algorithms in fields like finance, genetics research, retail, marketing and health care.

2) Write a note on reserved words in R programming. (4)

Reserved words in R programming are a set of words that have special meaning and cannot be used as an identifier (variable name, function name etc.). Here is a list of reserved words in the R's parser.

if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

Among these words, if, else, repeat, while, function, for, in, next and break are used for conditions, loops and user defined functions. They form the basic building blocks of programming in R. TRUE and FALSE are the logical constants in R. NULL represents the absence of a value or an undefined value. Inf is for "Infinity", for example when 1 is divided by 0, whereas NaN is for "Not a Number", for example when 0 is divided by 0. NA stands for "Not Available" and is used to represent missing values. R is a case sensitive language, which means that TRUE and True are not the same.

3) Write the rules for writing Identifiers in R. Give examples for valid and invalid identifiers. (4)

Rules for writing Identifiers in R

1. Identifiers can be a combination of letters, digits, period (.) and underscore (_).
2. It must start with a letter or a period. If it starts with a period, it cannot be followed by a digit.
3. Reserved words in R cannot be used as identifiers.

Example:

Valid identifiers in R total, Sum, .fine.with.dot, this_is_acceptable, Number5
Invalid identifiers in R tot@l, 5um, _fine, TRUE, .0ne

4) What is the output of the following built-in constants? (4)

a) LETTERS b) letters c) month.name d) month.abb

> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"

> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> **month.name**
[1] "January" "February" "March" "April" "May" "June"
[7] "July" "August" "September" "October" "November" "December"

> **month.abb**
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"

5) Write a note on R objects (6)

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.

R is an object-oriented language. Everything in R is an object. When R does anything, it creates and manipulates objects. R's objects come in different types and flavors. The most basic ones are:

• **Vectors**: These are one-dimensional sequences of elements of the same mode. (More on modes later: see section 3.2.) For example, this could be vector of length 26 (i.e. one containing 26 elements) where each element is a letter in the alphabet.

• **Matrices & Arrays**: These are two dimensional rectangular objects (matrices) and higherdimensional rectangular objects (arrays). All elements of matrices or arrays have to be of the same mode.

• **Lists**: Lists are like vectors but they do not have to contain elements of the same mode. The first element of a list could be a vector of the 26 letters of the alphabet. The second element could contain a vector of all the prime numbers below 1000. A third could be a 2 by 7 matrix.

• **Data Frames**: Data frames are best understood as special matrices (technically they are a type of list). For most applications involving datasets you will use data frames. They are two dimensional containers with rows corresponding to 'observations' and columns corresponding to 'variables.'

• **Factors**: Factors are vectors to classify categorical data. They behave differently than vectors containing numerical, integer, or character elements.

• **Functions**: Functions are objects that take other objects as inputs and return some new object.

6) What are vectors? Explain its types with examples.

A vector is simply a list of items that are of the same type. To combine the list of items to a vector, use the **c()** function and separate the items by a comma. Vectors are the most basic R data objects and there are six types of atomic vectors. They are logical, integer, double, complex, character and raw.

Single Element Vector

Even when you write just one value in R, it becomes a vector of length 1 and belongs to one of the above vector types.

```
# Atomic vector of type character.
print("abc");
# Atomic vector of type double.
print(12.5)
```

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

```
[1] "abc"
[1] 12.5
```

Multiple Elements Vector

To create a vector with numerical values in a sequence, use the : operator

```
# Creating a sequence from 5 to 13.
v <- 5:13
print(v)
# Creating a sequence from 6.6 to 12.6.
v <- 6.6:12.6
print(v)
```

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

```
[1] 5 6 7 8 9 10 11 12 13
[1] 6.6 7.6 8.6 9.6 10.6 11.6 12.6
```

7) Write a note on vector arithmetic with an example.

Two vectors of same length can be added, subtracted, multiplied or divided giving the result as a vector output.

```
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)

# Vector addition.
add.result <- v1+v2
print(add.result)

# Vector subtraction.
sub.result <- v1-v2
print(sub.result)

# Vector multiplication.
```

```
multi.result <- v1*v2
print(multi.result)

# Vector division.
divi.result <- v1/v2
print(divi.result)
```

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

```
[1] 7 19 4 13 1 13
[1] -1 -3 4 -3 -1 9
[1] 12 88 0 40 0 22
[1] 0.7500000 0.7272727 Inf 0.6250000 0.0000000 5.5000000
```

8) What are Lists? Explain with an example. (4)

Lists are the R objects which contain elements of different types like − numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using list() function.

Following is an example to create a list containing strings, numbers, vectors and a logical values.

```
# Create a list containing strings, numbers, vectors and a logical
# values.
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
print(list_data)
```

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

```
[[1]]
[1] "Red"
[[2]]
[1] "Green"
[[3]]
[1] 21 32 11
[[4]]
[1] TRUE
[[5]]
[1] 51.23
[[6]]
[1] 119.1
```

9) What is a matrix? Explain with an example. (5)

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only characters or only logical values, they are not of much use. We use matrices containing numeric elements to be used in mathematical calculations.

A matrix can be created with the matrix() function. Specify the nrow and ncol parameters to get the amount of rows and columns:

```
# Create a matrix
thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
# Print the matrix
thismatrix
```

Output for above code will be:

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

You can also create a matrix with strings:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
thismatrix
```

Output for above code will be:

```
     [,1]     [,2]
[1,] "apple"  "cherry"
[2,] "banana" "orange"
```

10) What is an array in R? How do you create and access elements of an array? Explain with an example.(4)

Arrays are the R data objects which can store data in more than two dimensions. For example − If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only one data type elements.

An array is created using the **array()** function. It takes vectors as input and uses the values in the **dim** parameter to create an array. The following example creates an array of two 3x3 matrices each with 3 rows and 3 columns.

```
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2))
print(result)
```

You can access the array elements by referring to the index position. You can use the [] brackets to access the desired elements from an array. The syntax is as follows: **array[row position, column position, matrix level]**

```
thisarray <- c(1:24)
> multiarray <- array(thisarray, dim = c(4, 3, 2))
> print(multiarray)
```

11) What is Data Frame in R? How do you create and access items in it? Explain with an example. (5)

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

We can use the data.frame() function to create a data frame.

Example 1:

```
# Create a data frame
Data_Frame <- data.frame (
        Training = c("Strength", "Stamina", "Other"),
        Pulse = c(100, 150, 120),
        Duration = c(60, 30, 45)
)
# Print the data frame
Data_Frame
```

Result:
Training Pulse Duration
1 Strength 100 60
2 Stamina 150 30
3 Other 120 45

Access Items
We can use single brackets [ ], double brackets [[ ]] or $ to access columns from a data frame:
Example:

```
Data_Frame <- data.frame (
        Training = c("Strength", "Stamina", "Other"),
        Pulse = c(100, 150, 120),
        Duration = c(60, 30, 45)
)
Data_Frame[1]
Data_Frame[["Training"]]
Data_Frame$Training
```

12) What is the use of str() function in data frame? Explain with an example. (5)

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
)
# Get the structure of the data frame.
str(emp.data)
```

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" ...
```

13) With an example, explain the following with respect to data frame. (6)

a) Extract the first two rows and then all columns

Extract the first two rows and then all columns

```
# 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
)
# Extract first two rows.
result <- emp.data[1:2,]
print(result)
```

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

```
emp_id emp_name salary start_date
1 1 Rick 623.3 2012-01-01
2 2 Dan 515.2 2013-09-23
```

b) Extract 3rd and 5th row with 2nd and 4th column

Extract 3rd and 5th row with 2nd and 4th column

```
# 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
)
# Extract 3rd and 5th row with 2nd and 4th column.
result <- emp.data[c(3,5),c(2,4)]
print(result)
```

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

```
emp_name start_date
3 Michelle 2014-11-15
5 Gary 2015-03-27
```

14) Briefly explain sort() function and order() function. (6)

The **sort()** function **sorts** the elements of a vector or a factor in increasing or decreasing order. The syntax of the sort function is: **sort(x, decreasing = FALSE, na.last = NA, . . .)**

Here,

- x is the input vector or factor that has to be sorted.
- **decreasing** is a boolean that controls whether the input vector or factor is to be sorted in decreasing order (when set to **TRUE**) or in increasing order (when set to **FALSE**).
- **na.last** is an argument that controls the treatment of the **NA** values present inside the input vector/factor. If **na.last** is set as **TRUE**, then the **NA** values are put at the last. If it is set as **FALSE**, then the **NA** values are put first. Finally, if it is set as **NA**, then the **NA** values are removed.

Example:

```
sort(c(3,16,34,77,29,95,24,47,92,64,43), decreasing = FALSE)
[1] 3 16 24 29 34 43 47 64 77 92 95
```

The **order()** function returns the indices of the elements of the input objects in ascending or descending order. Here is the syntax of the order function.

```
order(. . . , na.last = TRUE, decreasing = FALSE, method = c("auto", "shell", "radix"))
```

Where:

. . . is a sequence of numeric, character, logical or complex vectors or is a classed R object. This is the first argument of the function and is the object(s) that has to be ordered.
na.last is the argument that controls the treatment of **NA** values. **decreasing** controls whether the order of the object will be decreasing or increasing.
method is a character string that specifies the algorithm to be used. method can take the value of **"auto"**, **"radix"**, or **"shell"**.

Example:
> a <- c(20,40,70,10,50,30,90,60)
> order(a)
[1] 4 1 6 2 5 8 3 7
> a[order(a)]
[1] 10 20 30 40 50 60 70 90

15) Briefly explain sample() function and apply() function. (6)

Sample() function in R, generates a sample of the specified size from the data set or elements, either with or without replacement. Sample() function is used to get the sample of a numeric and character vector and also dataframe. The syntax is: **sample(x, size, replace = FALSE, prob = NULL)**

x	Data Set or a vector of one or more elements from which sample is to be chosen
size	size of a sample
replace	Should sampling be with replacement?
prob	probability weights for obtaining the elements of the vector being sampled

Lets see an example that generates 10 random sample from vector of 1 to 20. With replacement =TRUE. which means value in the sample can occur more than once.
> sample(1:20, 10, replace=TRUE)
[1] 10 16 6 13 6 4 6 1 12 6

The apply() function is the most basic of all collection. The apply in R function can be feed with many functions to perform redundant application on a collection of object (data frame, list, vector, etc.). The purpose of apply() is primarily to avoid explicit uses of loop constructs.
apply() takes Data frame or matrix as an input and gives output in vector, list or array. Apply function in R is primarily used to avoid explicit uses of loop constructs. It is the most basic of all collections can be used over a matrix. This function takes 3 arguments:
apply(X, MARGIN, FUN)
Here:
-x: an array or matrix
-MARGIN: take a value or range between 1 and 2 to define where to apply the function:
-MARGIN=1`: the manipulation is performed on rows
-MARGIN=2`: the manipulation is performed on columns
-MARGIN=c(1,2)` the manipulation is performed on rows and columns
-FUN: tells which function to apply. Built functions like mean, median, sum, min, max and even user-defined functions can be applied>
The simplest example is to sum a matrice over all the columns. The code apply(m1, 2, sum) will apply the sum function to the matrix 5×6 and return the sum of each column accessible in the dataset.
m1 <- matrix(C<-(1:10),nrow=5, ncol=6)
m1
a_m1 <- apply(m1, 2, sum)
a_m1

16) List and explain any two apply functions with an example? (6)

lapply() function is useful for performing operations on list objects and returns a list object of same length of original set. lappy() returns a list of the similar length as input list object, each element of which is the result of applying FUN to the corresponding element of list. Lapply in R takes list, vector or data frame as input and gives output in list.

lapply(X, FUN)

Arguments:

-X: A vector or an object

-FUN: Function applied to each element of x

l in lapply() stands for list. The difference between lapply() and apply() lies between the output return. The output of lapply() is a list. lapply() can be used for other objects like data frames and lists. lapply() function does not need MARGIN.

A very easy example can be to change the string value of a matrix to lower case with tolower function. We construct a matrix with the name of the famous movies. The name is in upper case format.

```
movies <- c("SPYDERMAN","BATMAN","VERTIGO","CHINATOWN")
movies_lower <-lapply(movies, tolower)
str(movies_lower)
```

output:
List of 4
$: chr "spyderman"
$: chr "batman"
$: chr "vertigo"
$: chr "chinatown"

sapply() function takes list, vector or data frame as input and gives output in vector or matrix. It is useful for operations on list objects and returns a list object of same length of original set. Sapply function in R does the same job as lapply() function but returns a vector.
sapply(X, FUN)
Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x
Example: We can measure the minimum speed and stopping distances of cars from the cars dataset (the data were recorded in the year 1920 and a data frame with 50 observations on 2 variables.).
dt <- cars
> lmn_cars <- lapply(dt, min)
> smn_cars <- sapply(dt, min)
> lmn_cars

17) Explain briefly types of data in descriptive statistics. (5)

Types and scales of data in descriptive statistics
A **data set** is a grouping of information that is related to each other. A data set can be either qualitative or quantitative. A **qualitative data (categorical data) set** consists of words that can be observed, not measured. A **quantitative data (numerical data) set** consists of numbers that can be directly measured. Months in a year would be an example of qualitative, while the weight of persons would be an example of quantitative data.

Types of Categorical Data
1. **Nominal Data:** When there is no natural order between categories then data is nominal type. Example: Color of an Eye, Gender (Male & Female), Blood Type, Political Party, and Zipcode, Type of living accommodation (House, Apartment, Trailer, Other), Religious preference (Hindu, Buddhist, Muslim, Jewish, Christian, Other), etc.

2. **Ordinal Data:** When there is natural order between categories then data is ordinal type. But here, the difference between the values in order does not matter. Example: Exam Grades, Socio-economic status (poor, middle class, rich), Education-level (kindergarten, primary, secondary, higher secondary, graduation), satisfaction rating(extremely dislike, dislike, neutral, like, extremely like), Time of Day(dawn, morning, noon, afternoon, evening, night), Level of Agreement(yes, maybe, no), The Likert Scale(strongly disagree, disagree, neutral, agree, strongly agree), etc.

Types of Numerical Data
1. **Discrete Data**: The data is said to be discrete if the measurements are integers. It represents count or an item that can be counted. Example: Number of people in a family, the number of kids in class, the number of cricket players in a team, the number of cricket playing nations in the world. Discrete data is a special kind of data because each value is separate and different. With any data, if we can answer the below questions then it is discrete. a. Can you count it? b. Can it be divided into smaller and smaller parts?

2. **Continuous Data**: The data is said to be continuous if the measurements can take any value usually within some range. It is a scale of measurement that can consist of numbers other than whole numbers,

like decimals and fractions. Example: height, weight, length, temperature Continuous data usually require a tool, like a ruler, measuring tape, scale, or thermometer, to produce the values in a continuous data set.

18) List and explain various Scales of Measurement in descriptive statistics. (5)

Scales of Measurement:
Data can be classified as being on one of four scales: **nominal, ordinal, interval or ratio.** Each level of measurement has some important properties that are useful to know.

1. **Nominal Scale:** Nominal datatype defined above can be placed into this category. They don't have a numeric value and so it neither be added, subtracted, divided nor be multiplied. They also have no order; if they appear to have an order then you probably have ordinal variables instead.

2. **Ordinal Scale:** Ordinal datatype defined above can be placed into this category. The ordinal scale contains things that you can place in order. For example, hottest to coldest, lightest to heaviest, richest to poorest. So, if you can rank data by 1st, 2nd, 3rd place (and so on), then you have data that is on an ordinal scale.

3. **Interval Scale:** An interval scale has ordered numbers with meaningful divisions. Temperature is on the interval scale: a difference of 10 degrees between 90 and 100 means the same as 10 degrees between 150 and 160. Compare that to Olympic running race (which is ordinal), where the time difference between the winner and runner up might be 0.01 second and between second-last and last 0.5 seconds. If you have meaningful divisions, you have something on the interval scale.

4. **Ratio Scale:** The ratio scale has all the property of interval scale with one major difference: zero is meaningful. When the scale is equal to 0.0 then there is none of that scale. For example, a height of zero is meaningful (it means you don't exist). The temperature in Kelvin(0.0 K), 0.0 Kelvin really does mean "no heat". Compare that to a temperature of zero, which while it exists, it doesn't mean anything in particular (although admittedly, in the Celsius scale it's the freezing point for water).

19) Explain central tendency measures with R Programming.

Central tendency is a descriptive summary of a dataset through a single value that reflects the center of the data distribution. The three most widely used measures of central tendency are **mean, median** and **mode**.

The mean is the arithmetic average, and is a common statistic used with interval/ratio data. It is simply the sum of the values divided by the number of values. The mean function in R will return the mean. Let us use beaver1 data of built-in R. [The beaver1 data frame has 114 rows and 4 columns on body temperature measurements at 10-minute intervals which stores Body temperature measured by telemetry every for every 10 minutes]. Call beaver1 data being built-in R by typing beaver1. Data description is as follows;
time=Time of observation, in the form 0330 for 3:30am
temp=Measured body temperature in degrees Celsius.
activ=Indicator of activity outside the retreat.

```
> head(beaver1)
day time temp activ
1 346 840 36.33 0
2 346 850 36.34 0
3 346 900 36.35 0
4 346 910 36.42 0
5 346 920 36.55 0
6 346 930 36.69 0
```

> mean(beaver1$temp)
[1] 36.86219 (i.e The average body temperature of participants is 36 Celsius.)

The median is defined as the value below which are 50% of the observations. To find this value manually, you would order the observations, and separate the lowest 50% from the highest 50%. For data sets with an odd number of observations, the median is the middle value. For data sets with an even number of observations, the median falls half-way between the two middle values. median() command is used to calculate the sample median.

> median(beaver1$temp)
[1] 36.87
The half of the body temperature of participants is 36.87 Celsius or below . OR The half of the body temperature of participants is 36.87 Celsius or above .
The mode is a summary statistic that is simply the value which occurs most frequently, when there are discreet values for a variable.

> library(DescTools)
> Mode(beaver1$temp) #with capital M
[1] 36.89

The most of the body temperature of participants is 36.89 Celsius.

20) Explain various measures of variation with R programming.

While measures of central tendency are used to estimate the central value of a dataset, measures of dispersion are important for describing the spread of data. Two data sets can have an equal mean (that is, measure of central tendency) but vastly different variability. The most commonly used measures of variation are range , interquartile range and standard deviation.
The range of a set of data is the difference between the largest and smallest values. Since it only depends on two of the observations, it is most useful in representing the dispersion of small data sets.
>range<-range(beaver1$temp)
>range[2]-range[1] #Shows the difference between minimum and maximum values
[1] 1.2
The difference between minimum temperature and maximum temperature is 1.2 Celsius.

Quantiles are cut points dividing the range of a probability distribution into contiguous intervals with equal probabilities, or dividing the observations in a sample in the same way.
> quantile(beaver1$temp)
0% 25% 50% 75% 100%
36.3300 36.7600 36.8700 36.9575 37.5300
> quantile(beaver1$temp,0.75) #3rd quantile
75%
36.9575
The 75% of the body temperature of participants is 36.96 Celsius or below . OR The 25% of the body temperature of participants is 36.96 Celsius or above.

Interquartile Range is the difference between 3rd quantile and 1st quantile. It is a measure of where the bulk of the values lie. It refers to spread of 50% of the data.
> IQR(beaver1$temp)
[1] 0.1975

The **variance** is a measure of how far each value in the data set is from the mean. Standard deviation is the measure of spread most commonly used in statistical practice when the mean is used to calculate central tendency. Thus, it measures spread around the mean.
> var(beaver1$temp)

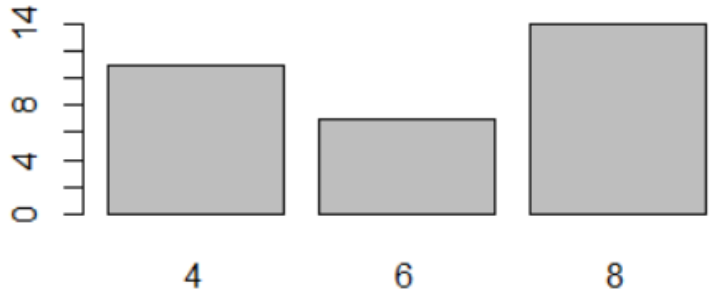
[1] 0.03741196

The interpretation of variance is not easy because the value you obtain is in the squared unit.
Standard deviation is a square root of variance.
> sd(beaver1$temp)
[1] 0.1934217

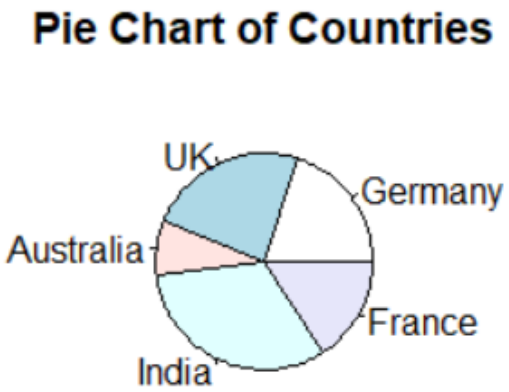
21) Briefly explain Graphics in R? (6)

Graphics is a great strength of R. The graphics package is part of the standard distribution and contains many useful functions for creating a variety of graphic displays.

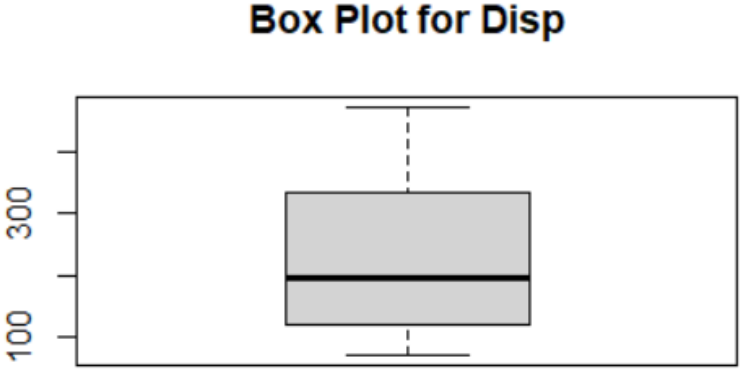
A **bar chart** or **bar graph** is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. It assumes that the heights of your bars are conveniently stored in a vector.



A **pie chart** (or a circle chart) is a circular statistical graphic which is divided into slices to illustrate numerical proportion. It is mainly used to represent categorical variables.

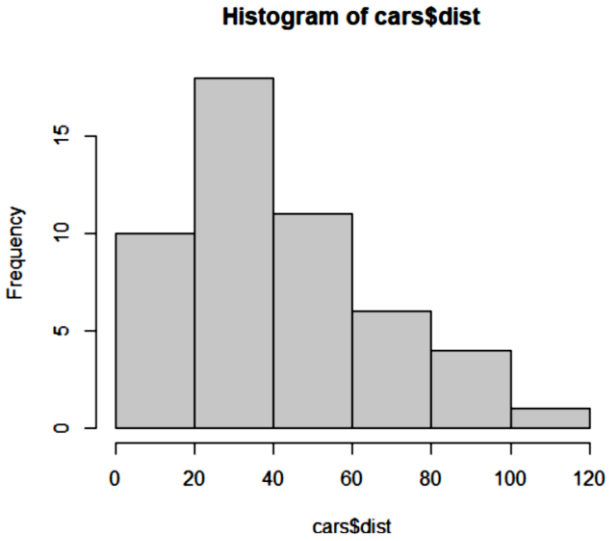


Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively.



A **histogram** gives an idea about the distribution of a quantitative variable. The idea is to break the range of values into intervals and count how many observations fall into each interval. Histograms are a bit

similar to barplots, but histograms are used for quantitative variables whereas barplots are used for qualitative variables.



A **stem and leaf plot**, also known as stem and leaf diagram or stem and leaf display is a classical representation of the distribution of quantitative data, similar to a histogram but in text, where the data is divided into the stem (usually the first or firsts digits of the number) and the leaf (the last digit). The stem and leaf plot in R can be useful when dealing with few observations (15 – 150 data points).

22) Explain stem and leaf plot with an example.

A **stem and leaf plot**, also known as stem and leaf diagram or stem and leaf display is a classical representation of the distribution of quantitative data, similar to a histogram but in text, where the data is divided into the stem (usually the first or firsts digits of the number) and the leaf (the last digit). The stem and leaf plot in R can be useful when dealing with few observations (15 – 150 data points).
Suppose that you have the following vector:
data<-c(12,15,16,21,24,29,30,31,32,33,45,46,49,50,52,58,60,63,64,65)
In order to manually create a stem plot you have to divide the data into stems (in this case, the first digit of the number) and leaves (the second digit). Hence, the plot will be as follows:
In consequence, in this example you can read 1|2561|256 as 12, 15 and 16, 2|1492|149 as 21, 24 and 29 and so on.
The **stem** function allows you to create a stem and leaf plot in R. The syntax of the function is as follows:
stem(x, # Numeric vector
scale = 1, # Length of the plot
width = 80, # Width of the plot
atom = 1e-08) # Tolerance parameter
You can create a simple stem plot typing:
>stem(data)
The output is the text displayed in the following block. Note that, to clarify, in the comments we show the corresponding values to each stem.

Stem		Leaf
1	I	256
2	I	149
3	I	0123
4	I	569
5	I	058
6	I	0345

\

Output

```
The decimal point is 1 digit(s) to the right of the |

0 | 256                 # <-- 12, 15, 16
2 | 1490123             # <-- 21, 24, 29, 30, 31, 32, 33
4 | 569028              # <-- 45, 46, 49, 50, 52, 58
6 | 0345                # <-- 60, 63, 64, 65
```

UNIT-II

Two Marks Questions:

1) Define database and DBMS

database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures.

2) Mention different tasks that are allowed by DBMS users.

- Data definition
- Data retrieval
- Data updating
- User administration

3) What is Flat File Database

A Flat file database is also known as a **text database**. It is the most important type of database used to store data in plain text files such as MS Excel.

4) Mention any 2 advantages of Flat file database.

1. It is an excellent option for small databases.
2. It requires fewer hardware and software components.

5) Mention any 2 disadvantages of Flat file database

1. Flat file database is harder to update.
2. Harder to change the data format.

6) Mention any 4 characteristics of Database Management System [any 4]

- Real World Entity
- Self-Describing Nature
- Support ACID Properties
- Multiple User Interfaces
- Backup and Recovery
- Supports Multiple Views
- Stores Any Kind of Data
- Security

7) Write any 4 advantages of Database Management System

- Reducing Data Redundancy
- Sharing of Data
- Data Integrity
- Data Security

8) Write any 4 limitations of Database Management System

- More Expensive

- High Complexity
- Database Failure
- Huge Size

9) Mention any 4 functions of Database Management System [any 4]

- Data Dictionary Management
- Data Storage Management
- Data transformation and presentation:
- Security Management
- Backup and Recovery Management
- Data Integrity Management

10) What is relational database?

A relational database is a type of database that stores and provides access to data points that are related to one another.

11) Write any 2 differences between flat file database vs. relational database

Flat file database	**Relational database**
A flat-file database is used to store data in a single table structure.	A relational database is used to store data in a multiple-table structure.
It includes software like FileMaker, Berkeley DB.	It includes software like Oracle, MySQL.
It is represented using a Data dictionary.	It is represented using a Schema.
It is less secure.	It is more secure.

12) Name the different types of DBMS Architecture

- 1-Tier Architecture
- 2-tier architecture and
- 3-tier architecture.

13) Name any 4 types of database users

- Database Administrator (DBA)
- System Analyst
- Sophisticated Users
- Data Base Designers
- Casual Users / Temporary Users

14) What is Associative Data Model?

The associative Data Model is a model in which the data is divided into two parts. Everything which has independent existence is called an entity and the relationship among these entities are called association.

15) Mention any 4 different types of keys in DBMS

- Primary Key
- Candidate Key
- Super Key
- Foreign Key

16) What is Primary Key and Candidate Key?

A primary key is the column or columns that contain values that uniquely identify each row in a table. Candidate keys are defined as the set of fields from which the primary key can be selected.

17) What is Foreign Key and Composite Key?

Foreign keys are the column of the table used to point to the primary key of another table. Whenever a primary key consists of more than one attribute, it is known as a composite key.

18) What is first normal form?

It is the first step in Normalisation. Therefore, the foremost rule for a table to be in a normalized form is that it should have only atomic values.

19) What is second normal form?

A relation is said to be in 2nd Normal Form in DBMS (or 2NF) when it is in the First Normal Form but has no non-prime attribute functionally dependent on any candidate key's proper subset in a relation.

20) Expand BCNF and SQL.

Boyce–Codd normal form (BCNF)

Structured Query Language

Long Answer Questions:

1) Write a note on Flat file database (5)

A Flat file database is also known as a **text database**. It is the most important type of database used to store data in plain text files such as MS Excel. Flat file databases were **developed by IBM** in the early **1970s**. In the Flat file database, each line of the plain text file holds only one record. These records are separated using delimiters, such as tabs and commas. The advantage of a flat-file database is that it is easy to understand and helps us to sort the results easily.

Advantages of Flat file database

3. All records are stored in one place.
4. Easy to understand and configure using various standard office applications.
5. It is an excellent option for small databases.
6. It requires fewer hardware and software components.

Disadvantages of flat-file database

3. Flat file database is harder to update.
4. Harder to change the data format.
5. It is a poor database for complex queries.
6. It increased Redundancy and inconsistency

2) Explain the characteristics of Database Management System (4/5/6)

➢**Real World Entity:** DBMS these days is very realistic and real-world entities are used to design its architecture. Also, behaviour and attributes are used by DBMS. To simplify it we can take an example of an organization database where the employee is an entity and his employee id is an attribute.

➢ **Self-Describing Nature:** A DBMS should be of Self- Describing nature as it not only contains the database itself but also the metadata. Metadata (data about data) defines and describes not only the extent, type, structure, and format of all data but also the relationship between data. This data represents itself that what actions should be taken on it.

➢ **Support ACID Properties:** Any DBMS is able to support ACID (Accuracy, Completeness, Isolation, and Durability) properties. It is made sure in every DBMS that the real purpose of data should not be lost while performing transactions like delete, insert, and update.

• **Atomicity** defines all the elements that make up a complete database transaction.
• **Consistency** defines the rules for maintaining data points in a correct state after a transaction.
• **Isolation** keeps the effect of a transaction invisible to others until it is committed, to avoid confusion.
• **Durability** ensures that data changes become permanent once the transaction is committed.

➢ **Backup and Recovery:** There are many chances of failure of the whole database. At that time no one will be able to get the database back and The only solution is to take a backup of the database and whenever it is needed, it can be stored back. A database must have this characteristic to enable more effectiveness.

➢ **Stores Any Kind of Data:** A database management system should be able to store any kind of data. It should not be restricted to employee name, salary, and address. Any kind of data that exists in the real world can be stored in DBMS because we need to work with all kinds of data that is present around us.

➢ **Security:** DBMS provides security to the data stored in it because all users have different rights to access the database. Some of the users can access the whole database while others can access a small part of the database. For example, a computer network lecturer can only access files that are related to computer subjects but the HOD of the department can access files of all subjects that are related to their department.

3) Explain the advantages of Database Management System (4/5/6)

- **Reducing Data Redundancy**: The file-based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes multiple copies of the same file which lead to data redundancy. This is prevented in a database as there is a single database and any change in it is reflected immediately. Because of this, there is no chance of encountering duplicate data.
- **Sharing of Data**: In a database, the users of the database can share the data among themselves. There are various levels of authorization to access the data, and consequently, the data can only be shared based on the correct authorization protocols being followed. Many remote users can also access the database simultaneously and share the data between themselves.
- **Data Integrity**: Data integrity means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All of these databases contain data that is visible to multiple users. So, it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.
- **Data Security**: Data Security is a vital concept in a database. Only authorized users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorized users are not allowed to access the database under any circumstances as it violates integrity constraints.

- **Privacy**: The privacy rule in a database means only authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he is allowed to.
- **Automatic Backup and Recovery**: Database Management System automatically takes care of backup and recovery. The users don't need to back up data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.
- **Data Consistency**: Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is the same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

4) Explain the limitations of Database Management System (4/5/6)

- **More Expensive:** Creating and managing a database is quite costly. High-cost software and hardware is required for the database. Also, highly trained staff is required to handle the database and it also needs continuous maintenance. All of these end up making a database quite a costly venture.
- **High Complexity:** A Database Management System is quite complex as it involves creating, modifying, and editing a database. Consequently, the people who handle a database or work with it need to be quite skilled, or valuable data can be lost.
- **Database handling staff required:** Databases and DBMS are quite complex. Hence, skilled personnel is required to handle the database so that it works in optimum condition. This is a costly venture as these professionals need to be very well paid.
- **Database Failure:** All the relevant data for any company is stored in a database. So it is imperative that the database works in optimal condition and there are no failures. A database failure can be catastrophic and can lead to the loss or corruption of very important data.
- **High Hardware Cost:** A database contains a vast amount of data. So large disk storage is required to store all this data. Sometimes extra storage may even be needed. All this increases hardware costs by a lot and makes a database quite expensive.
- **Huge Size:** A database contains a large amount of data, especially for bigger organizations. This data may even increase as more data is updated in the database. All of these lead to the large size of the database. If the database is bigger, it is more difficult it is to handle and maintain. It is also more complex to ensure data consistency and user authentication across big databases.
- **Upgradation Costs:** Often new functionalities are added to the database. This leads to database upgradation. All of these upgrades cost a lot of money. Moreover, it is also quite expensive to train database managers and users to handle these new upgrades.
- **Cost of Data Conversion:** If the database is changed or modified in some manner, all the data needs to be converted to the new form. This cost may even exceed the database creation and management costs sometimes. This is the reason most organizations prefer to work on their old databases rather than upgrade to new ones.

5) Explain the functions of Database Management System (4/5/6)

1. **Data Dictionary Management**: DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary. So, all programs that access the data in the database work through the DBMS. The DBMS uses the data dictionary to look up the required data component structures and relationships which relieves you from coding such complex relationships in each program. Additionally, any changes made in a database structure are automatically recorded in the data dictionary, thereby freeing you from having to modify all of the programs that access the changed structure.
2. **Data Storage Management**: One of the DBMS functionalities is creating and managing the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics. A modern DBMS system provides storage not only for the data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture formats, and so on. Data storage management is also important for database performance tuning. Performance tuning relates to the activities that make the database perform more efficiently in terms of storage and access speed.
3. **Data transformation and presentation**: The DBMS transforms entered data into required data structures. The DBMS relieves you of the chore of making a distinction between the logical data format and the physical data format. The DBMS formats the physically retrieved data to make it conform to the user's logical expectations. For example, imagine an enterprise database used by a multinational company. An end-user in England would expect to enter data such as July 11, 2009, as "11/07/2009." In contrast, the same date would be entered in the United States as "07/11/2009." Regardless of the data presentation format, the DBMS system must manage the data in the proper format for each country.
4. **Security Management**: The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform. This is especially important in multiuser database systems.
5. **Multi-User Access Control**: Multiuser access control is another important DBMS Function. To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database.
6. **Backup and Recovery Management**: The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity.
7. **Data Integrity Management**: The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity.
8. **Database locking and Concurrency**: Conflicts can arise in a database when multiple users or applications attempt to change the same data at the same time. Locking and concurrency techniques reduce the potential for conflicts while maintaining the integrity of the data. Locking prevents other users and applications from accessing data while it is being updated. In some

databases, locking applies to the entire table, which creates a negative impact on application performance. Other databases, such as Oracle relational databases, apply locks at the record level, leaving the other records within the table available, helping ensure better application performance. Concurrency manages the activity when multiple users or applications invoke queries at the same time on the same database. This capability provides the right access to users and applications according to policies defined for data control.

9. **Facilitate Database Access Languages and Application Programming Interfaces**: The DBMS provides data access through a query language. A query language is a non-procedural language—one that lets the user specify what must be done without having to specify how it is to be done. Structured Query Language (SQL) is the de facto query language and data access standard supported by the majority of DBMS vendors.
10. **Provide Database Communication Interfaces**: Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer. In this environment, communications can be accomplished in several ways:
    - End users can generate answers to queries by filling in screen forms through their preferred Web browser.
    - The DBMS can automatically publish predefined reports on a website.
    - The DBMS can connect to third-party systems to distribute information via e-mail or other productivity applications

6) Differentiate flat file database vs. relational database (4/5)

Flat file database	Relational database
A flat-file database is used to store data in a single table structure.	A relational database is used to store data in a multiple-table structure.
A flat-file database can be accessed by a variety of software applications.	A relational database uses a Relational Database Management System (RDBMS) to access the data.
Flat file database includes software like FileMaker, Berkeley DB, and Borland Reflex.	The relational database includes software like Oracle, MySQL, and PostgreSQL.
It is simple, portable, easy to use, and inexpensive	It is more powerful and more efficient as well as more expensive than the Flat file database.
It is represented using a Data dictionary.	It is represented using a Schema.
It contains files, records, characters, and fields.	It contains the entity's attributes and relationships.
In a Flat file database, there is a problem of data redundancy.	In a Relational database, there is no problem with data redundancy.
It is less secure.	It is more secure than the Flat file database.

7) Write a note on types of DBMS Architecture (5/6)

The DBMS architecture shows how data in the database is viewed by the users. It is not concerned about how the data are handled and processed by the DBMS. The concept of DBMS depends upon its architecture. The architecture can be designed as centralized, decentralized, or hierarchical.

Types of DBMS Architecture

1-Tier Architecture

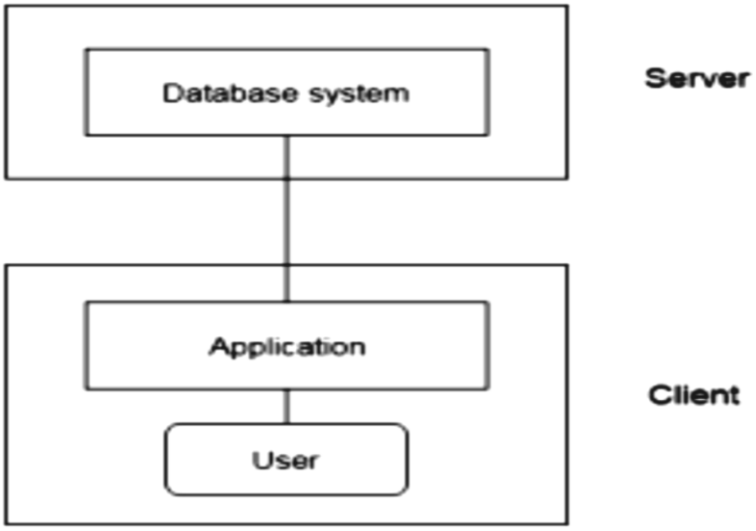
2-tier architecture and
3-tier architecture.

1-Tier Architecture

DBMS is the simplest architecture of a Database in which the client, server, and Database all reside on the same machine.

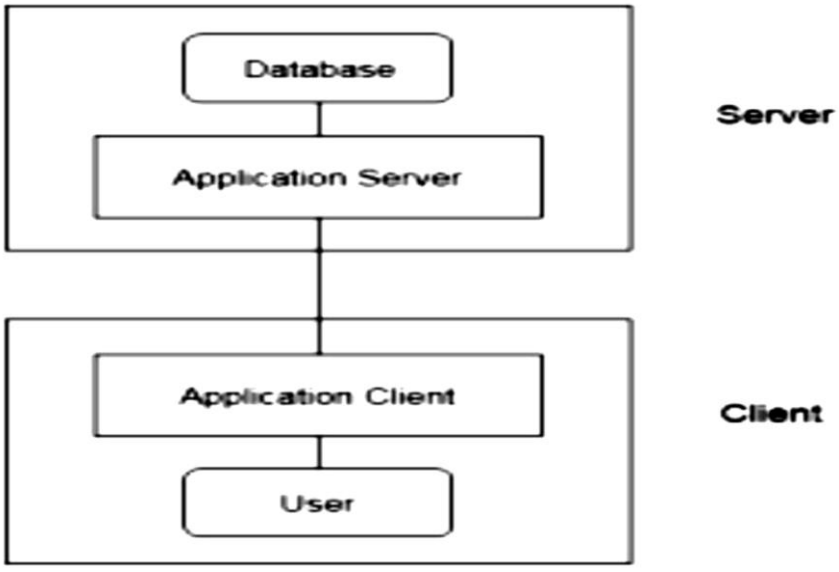
Here, the user can directly sit on the DBMS and uses it. Any changes done here will directly be done on the database itself. The 1-Tier architecture is used for the development of the local application, where programmers can directly communicate with the database for a quick response.

2-Tier Architecture



A 2-Tier Architecture is a Database architecture where the presentation layer runs on a client and data is stored on a server called the second tier. Two-tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.

3-Tier Architecture



In this architecture, the client can't directly communicate with the server. The application on the client-end interacts with an application server which further communicates with the database system. The 3-Tier architecture is used in the case of the large web application.

A 3-tier architecture has the following layers:
1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server

8) Explain different types of database users (4/5/6)

1. Database Administrator (DBA) : A database administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of the database. The DBA will then create a new account id and password for the user if he/she needs to access the database. DBA is also responsible for providing security to the database and he allows only authorized users to access/modify the database.

- DBA also monitors the recovery and backup and provides technical support.
- The DBA has a DBA account in the DBMS which is called a system or superuser account.
- DBA repairs damage caused due to hardware and/or software failures.

2. Naive / Parametric End Users: Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results. For example, Railway's ticket booking users are naive users. Clerks in any bank is a naive users because they don't have any DBMS knowledge but they still use the database and perform their given task.

3. System Analyst: A System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end-users are satisfied.

4. Sophisticated Users: Sophisticated users can be engineers, scientists, and business analysts, who are familiar with the database. They can develop their own database applications according to their requirement. They don't write the program code but they interact with the database by writing SQL queries directly through the query processor.

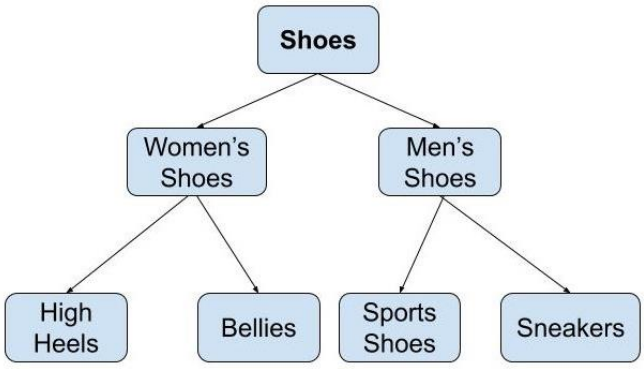
5. Data Base Designers: Database Designers are the users who design the structure of a database which includes tables, indexes, views, constraints, triggers, and stored procedures. He/she controls what data must be stored and how the data items are related.

6. Application Programmer: The application Programmer is the back-end programmer who writes the code for the application programs. They are computer professionals. These programs could be written in programming languages such as Visual Basic, Developer, C, Java, etc.

7. Casual Users / Temporary Users: Casual Users are the users who occasionally use/access the database but each time when they access the database they require the new information, for example, a Middle or higher-level manager.

9) Briefly explain hierarchical model (5/6)

The hierarchical Model was the first DBMS model. This model organizes the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding a child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemaps of a website, etc.



Hierarchical Model

Features of a Hierarchical Model

1. **One-to-many relationship:** The data here is organized in a tree-like structure where the one-to many relationship is between the datatypes. Also, there can be only one path from the parent to any node. Example: In the above example, if we want to go to the node sneakers we only have one path to reach there i.e through the men's shoes node.
2. **Parent-Child Relationship:** Each child node has a parent node but a parent node can have more than one child node. Multiple parents are not allowed.
3. **Deletion Problem:** If a parent node is deleted then the child node is automatically deleted.
4. **Pointers:** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. Example: In the above example the 'shoes' node points to the two other nodes ' women's shoes' node and the 'men's shoes' node.

Advantages of Hierarchical Model

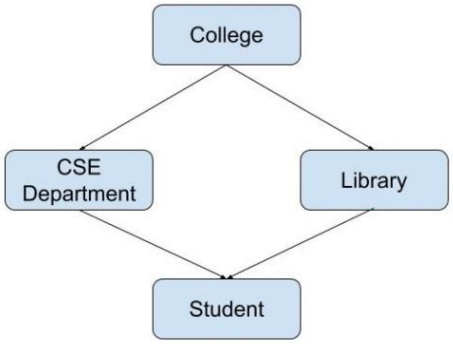
- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages of Hierarchical Model

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted.

10) Explain Network Model (5/6)

This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph. Example: In the example below we can see that the node student has two parents i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.



Network Model

Features of a Network Model

1. **Ability to Merge more Relationships:** In this model, as there are more relationships so data is more related. This model has the ability to manage one-to-one relationships as well as many-to many relationships.
2. **Many paths:** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
3. **Circular Linked List:** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

Advantages of Network Model

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in the parent record is reflected in the child record.

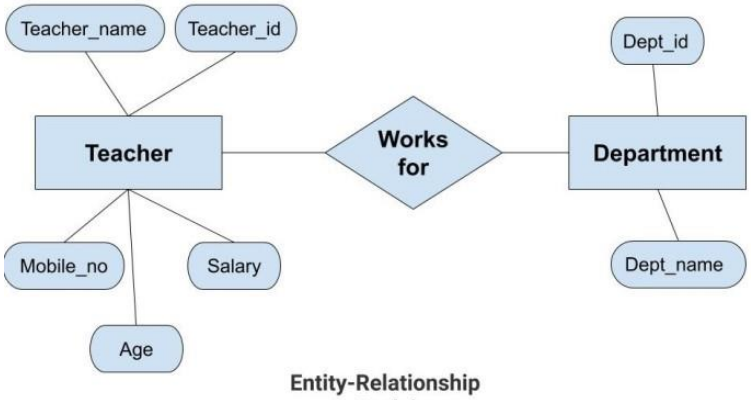
Disadvantages of Network Model

- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.
- Any change like updating, deletion, or insertion is very complex.

11) Briefly explain Entity-Relationship Model (5/6)

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram. We use the ER diagram as a visual tool to represent an ER Model. ER diagram has the following three components:

- **Entities:** Entity is a real-world thing. It can be a person, place, or even a concept. Example: Teachers, Students, Courses, buildings, departments, etc are some of the entities of a School Management System.
- **Attributes:** An entity contains a real-world property called an attribute. These are the characteristics of that attribute. Example: The entity teacher has the property like teacher id, salary, age, etc.
- **Relationship:** Relationship tells how two attributes are related. Example: A teacher works for a department.



Entity-Relationship Model

In the above diagram, the entities are teachers and departments. The attributes of the Teacher entity are Teacher_Name, Teacher_id, Age, Salary, and Mobile_Number. The attributes of entity Department entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

Features of ER Model

- **Graphical Representation for Better Understanding:** It is very easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
- **ER Diagram:** ER diagram is used as a visual tool for representing the model.
- **Database Design:** This model helps the database designers to build the database and is widely used in database design.

Advantages of ER Model

- **Simple:** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- **Effective Communication Tool:** This model is used widely by the database designers for communicating their ideas.
- **Easy Conversion to any Model:** This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

Disadvantages of ER Model

- **No industry standard for notation:** There is no industry standard for developing an ER model. So one developer might use notations which are not understood by other developers.
- **Hidden information:** Some information might be lost or hidden in the ER model. As it is a highlevel view so there are chances that some details of information might be hidden.

12) Write a note on Relational Model. (5/6)

- The relational model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of rows and columns. The basic structure of a relational model is tables. So, the tables are also called relations in the relational model. Example:

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Features of Relational Model

- **Tuples:** Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
- **Attribute or field:** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above example, we have different attributes of the employee like Salary, Mobile_no, etc.

Advantages of Relational Model

- **Simple:** This model is simpler as compared to the network and hierarchical model.
- **Scalable:** This model can be easily scaled as we can add as many rows and columns we want.
- **Structural Independence:** We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data we can say that structural independence has been achieved.

Disadvantages of the Relational Model

- Hardware Overheads: For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.
- Bad Design: The relational model is very easy to design and use. So, the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows.

13) Write a note on Database Security (4/5)

- Database Security means keeping sensitive information safe and preventing the loss of data. The security of the database is controlled by Database Administrator (DBA).
- The following are the main control measures are used to provide security of data in databases:

1. Authentication: Authentication is the process of confirming whether the user logs in only according to the rights provided to him to perform the activities of the database. A particular user can log in only up to his privilege but he can't access the other sensitive data. The privilege of accessing sensitive data is restricted by using Authentication. Using these authentication tools for biometrics such as retina and figure prints can prevent the database from unauthorized/malicious users.

2. Access Control: The security mechanism of DBMS must include some provisions for restricting access to the database by unauthorized users. Access control is done by creating user accounts and controlling the login process by the DBMS. So, that database access of sensitive data is possible only to those people (database users) who are allowed to access such data and to restrict access to unauthorized persons. The database system must also keep the track of all operations performed by certain users throughout the entire login time.

3. Inference Control: This method is known as the countermeasure to the database security problem. It is used to prevent the user from completing any inference channel. This method protects sensitive information from indirect disclosure. Inferences are of two types, identity disclosure, and attribute disclosure.

4. Flow Control: This prevents the information from flowing in a way that it reaches unauthorized users. Channels are the pathways for information to flow implicitly in ways that violate the privacy policy of a company and are called convert channels.

5. Database Security applying Statistical Method: Statistical database security focuses on the protection of confidential individual values stored in and used for statistical purposes and used to retrieve the summaries of values based on categories. They do not permit retrieving individual information. This allows to access the database to get statistical information about the number of employees in the company but not to access the detailed confidential/personal information about the specific individual employee.

6. Encryption: This method is mainly used to protect sensitive data (such as credit card numbers, OTP numbers) and other sensitive numbers. The data is encoded using some encoding algorithms. An unauthorized user who tries to access this encoded data will face difficulty in decoding it, but authorized users are given decoding keys to decode data.

14) Explain different types of keys in DBMS (4/5)

- **Primary Key**: A primary key is the column or columns that contain values that uniquely identify each row in a table. A database table must have a primary key for the option to insert, update, restore, or delete data from a database table. A primary key is either an existing table column or a column that is specifically generated by the database according to a defined sequence. For example, students are routinely assigned unique identification (ID) numbers, and all adults receive government-assigned and uniquely-identifiable Social Security numbers. The primary key's main features are:
    - It must contain a unique value for each row of data.
    - It cannot contain null values.

- **Candidate Key**: A candidate key is the most minimal subset of fields that uniquely identifies a tuple. Candidate keys are defined as the set of fields from which the primary key can be selected. It is an attribute or set of attributes that can act as a primary key for a table to uniquely identify each record in that table. EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered candidate key.

- **Super Key**: Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key. In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

- **Foreign Key**: Foreign keys are the column of the table used to point to the primary key of another table. The table that contains the primary key is called the Parent table and the one which contains the foreign key is called the child table. Example: Consider the tables Employee(Emp_id, Name, SSN, Passport_no, License_no, Dept_id) and Department(Dept_id, Location), here Dept_id in the Table Employee is called a foreign key.

- **Composite Key**: Whenever a primary key consists of more than one attribute, it is known as a composite key. **For example,** in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination.

- **Alternate Key**: There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. In other words, the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key. For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate

keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

- **Artificial Key**: The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order**. For example,** the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

15) What is normalization? Explain 1NF and 2NF. (5)

Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

First Normal Form (1NF)

In this Normal Form, we tackle the problem of atomicity. Here atomicity means values in the table should not be further divided. In simple terms, a single cell cannot hold multiple values. If a table contains a composite or multi-valued attribute, it violates the First Normal Form.

Example:

Emp_ID	Emp_Name	Cell_No	Salary
1ASP001	Bindu	+918554645678 +916783456678	61,980
1ASP002	Deepa	+917829107721	89,800
1ASP003	Dinkar	+919554745688 +918783456688	56,790
1ASP004	Jitha	+919945784567	45,570
1ASP005	Sam	+919987564488 +918788676488	93,880

In the above table, we can clearly see that the Phone Number column has two values. Thus, it violated the 1st NF. Now if we apply the 1st NF to the above table, we get the below table as the result.

Emp_ID	Emp_Name	Cell_No	Salary
1ASP001	Bindu	+918554645678	61,980
1ASP001	Bindu	+916783456678	61,980
1ASP002	Deepa	+917829107721	89,800
1ASP003	Dinkar	+919554745688	56,790
1ASP003	Dinkar	+918783456688	56,790
1ASP004	Jitha	+919945784567	45,570
1ASP005	Sam	+919987564488	93,880
1ASP005	Sam	+918788676488	93,880

Second Normal Form (2NF)

- The first condition in the 2nd NF is that the table has to be in the 1st Normal Form. The table also should not contain partial dependency.
- In the second normal form, all non-key attributes are fully functionally dependent on the primary key. The Second Normal Form eliminates partial dependencies on primary keys.

Example

Student_Project

Student_ID	Project_ID	Student_Name	Project_Name

S101	P01	Ada	Health Monitoring
S103	P06	Ana	IoT Devices
S110	P09	Jacob	Data Analytics
S116	P11	Jane	Cloud Deployment
S119	P07	Alex	Geo Location

In the above table, we have partial dependency; let us see how −

The prime key attributes are **StudentID** and **ProjectID**.

As stated, the non-prime attributes i.e. **StudentName** and **ProjectName** should be functionally dependent on part of a candidate key, to be Partial Dependent.

The **StudentName** can be determined by **StudentID**, which makes the relation Partial Dependent.

The **ProjectName** can be determined by **ProjectID**, which makes the relation Partial Dependent.

Therefore, the **StudentProject** relation violates the 2NF in Normalization and is considered a bad database design. To eliminate partial dependencies this table has to be decomposed into two as follows:

Student_Info

Student_ID	Project_ID	Student_Name
S101	P01	Ada
S103	P06	Ana
S110	P09	Jacob
S116	P11	Jane
S119	P07	Alex

Project Info

Project_ID	Project_Name
P01	Health Monitoring
P06	IoT Devices
P09	Data Analytics
P11	Cloud Deployment
P07	Geo Location

16) Explain the features of MYSQL (4/5/6) [max any 6 or 7 based on the marks]

Open-Source: MySQL is open-source, which means this software can be downloaded, used, and modified by anyone. It is free to use and easy to understand. The source code of MySQL can be studied and changed based on the requirements. It uses GPL, i.e. GNU General Public license which defines rules and regulations regarding what can and can't be done using the application.

Quick and Reliable: MySQL stores data efficiently in the memory ensuring that data is consistent, and not redundant. Hence, data access and manipulation using MySQL are quick.

Scalable: Scalability refers to the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines, and so on. MySQL server was developed to work with large databases.

Data Types: It contains multiple data types such as unsigned integers, signed integers, float (FLOAT), double (DOUBLE), character (CHAR), variable character (VARCHAR), text, blob, date, time, DateTime, timestamp, year, and so on.

Character Sets: It supports different character sets, including latin1 (cp1252 character encoding), German, other Unicode character sets, etc.

Secure: It provides a secure interface since it has a flexible password system, and ensures that it is verified based on the host before accessing the database. The password is encrypted while connecting to the server.

Support for large databases: It comes with support for large databases, which could contain about 40 to 50 million records, 150,000 to 200,000 tables, and up to 5,000,000,000 rows.

Client and Utility Programs: MySQL server also comes with many client and utility programs. This includes Command line programs such as 'mysqladmin' and graphical programs such as 'MySQL Workbench'. MySQL client programs are written in a variety of languages. Client library (code encapsulated in a module) can be written in C or C++ and available for clients with C bindings.

Compatible with many operating systems: MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, and many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or another computer (communication via a local network or the Internet).

Allows roll-back: MySQL allows transactions to be rolled back, commit, and crash recovery.

Memory efficiency: Its efficiency is high because it has a very low memory leakage problem.

High Performance: MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast-loading utilities because of the different cache memory.

High Productivity: MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.

GUI Support: MySQL provides a unified visual database graphical user interface tool named "**MySQL Workbench**" to work with database architects, developers, and Database Administrators. MySQL Workbench provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. MySQL has full GUI support from MySQL Server version 5.6 and higher.

Dual Password Support: MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password, which allows us to transition to the new password.

UNIT-III

Two marks questions

1) Expand SQL, CRUD

SQL:Structured Query Language

CRUD: Create, Read, Update, Delete

2) Name the four basic SQL operations.

Create, Read, Update, Delete

3) Expand DDL, DML

DDL – Data Definition Language

DML – Data Manipulation Language

4) Expand DQL, DCL

1. DQL – Data Query Language
2. DCL – Data Control Language

5) Why DROP statement is used in SQL? Give example.

- DROP: This command is used to delete objects from the database.

DROP TABLE table_name;

table_name: Name of the table to be deleted.

6) What are data constraints? Name any two table level constraints.

The data constraints are rules which are enforced on the data being entered, and prevents the user from entering invalid data into the database tables.

- Primary key
- Foreign key

7) Differences between Grant and Revoke commands

Grant	Revoke
This DCL command grants permissions to the user on the database objects.	This DCL command removes permissions if any granted to the users on database objects.
It assigns access rights to users.	It withdraws the access rights from users.
For each user, you need to specify the permissions	If access for one user is removed; all the particular permissions provided by that users to others will be removed.
When the access is decentralized granting permissions will be easy.	If decentralized access removing the granted permissions is difficult.

8) Name any two String data types.

- CHAR
- VARCHAR

9) Name any two Numeric data Types.

- SMALLINT
- BIGINT

10) Name any two Date and Time data Types.

- DATE
- TIME

11) Name any two Aggregate functions[any 2]

FIRST(), LAST(), MAX(), MIN()

12) Name any two Scalar functions[any 2]

UCASE(), LCASE(), MID(), LEN(), ROUND(), NOW()

13) What is the use of COMMIT and ROLLBACK statements in SQL?

The COMMIT statement lets a user save any changes or alterations on the current transaction. These changes then remain permanent. The ROLLBACK statement lets a user undo all the alterations and changes that occurred on the current transaction after the last COMMIT.

14) What is JOIN Operation in SQL? Give its types.

JOIN means "to combine two or more tables". In SQL, the JOIN clause is used to combine the records from two or more tables in a database.

Types of SQL JOIN

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL JOIN

15) Name the two types of file organization mechanism which are followed by the indexing methods to store the data.

1. Sequential File Organization or Ordered Index File

2. Hash File organization

Long answer questions

1) Explain the features of SQL (5/6)

- **Highly Accessible**: SQL is compatible with most databases, such as Microsoft SQL Server, Oracle Database, MySQL, SAP HANA, SAP Adaptive Server, and Microsoft Access. Most relational database management systems have SQL support, and they come with additional features as well. One can also use SQL to create application extensions for procedural programming.
- **SQL has great transactional support**. SQL can support enormous records and handle tons of transactions at the same time.
- **SQL has great security features**. Permissions are easy to manage with SQL. It's very simple to provide and verify your permissions, ensuring the security of your data.
- **SQL suits organizations of any size:** Is your company compatible with SQL? The answer is probably yes. SQL is perfect for small and big businesses. It can easily scale upwards to accommodate all your future growth.
- **Rapid query processing**: SQL processes a large amount of data in a very short amount of time. The core operations such as deletion, insertion, and any kind of data manipulation happen at lightning speeds because of their efficiency.
- **Requires little coding knowledge**. If all you want from your database is simple data retrieval, you don't need to learn to code. All you need to do is learn the select, insert, update, and into functions and their syntax. The language is not complex or verbose.
- **SQL is a standardized language**: SQL is open-source and supported by its community of users. You can find documentation and troubleshooting guidance no matter where you are in the world.
- **SQL is portable**: SQL can be used on desktop computers, gaming systems, laptops, and servers. You can even embed it within applications according to your personal needs and preferences.

2) Explain the steps involved in Processing an SQL Statement (5/6)

Processing a SQL Statement: The steps involved are common to all three techniques, although each technique performs them at different times. The following illustration shows the steps involved in processing an SQL statement

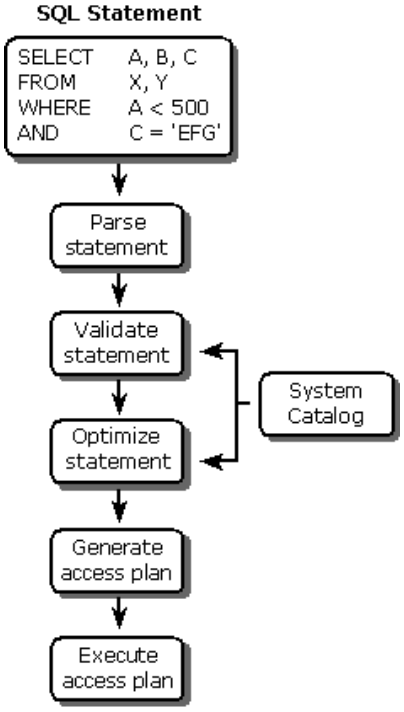


Fig. 3.2 Steps involved in Processing an SQL Statement

To process an SQL statement, a DBMS performs the following five steps:

1. The DBMS first parses the SQL statement. It breaks the statement up into individual words, called tokens, makes sure that the statement has a valid verb and valid clauses, and so on. Syntax errors and misspellings can be detected in this step.

2. The DBMS validates the statement. It checks the statement against the system catalog. Do all the tables named in the statement exist in the database? Do all of the columns exist and are the column names unambiguous? Does the user have the required privileges to execute the statement? Certain semantic errors can be detected in this step.

3. The DBMS generates an access plan for the statement. The access plan is a binary representation of the steps that are required to carry out the statement; it is the DBMS equivalent of executable code.

4. The DBMS optimizes the access plan. It explores various ways to carry out the access plan. Can an index be used to speed a search? Should the DBMS first apply a search condition to Table A and then join it to Table B, or should it begin with the join and use the search condition afterward? Can a sequential search through a table be avoided or reduced to a subset of the table? After exploring the alternatives, the DBMS chooses one of them.

5. The DBMS executes the statement by running the access plan.

3) Explain the statements with examples i) CREATE ii) INSERT (6)

i] create

The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
);
```

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

SQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

Example

```
CREATE TABLE Persons (
   PersonID int,
   LastName varchar(255),
   FirstName varchar(255),
   Address varchar(255),
   City varchar(255)
);
```

ii] insert

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

4) What is the use of ALTER statement in SQL? Explain. (5)

The ALTER TABLE Command

ALTER TABLE – ADD: ADD is used to add columns to an existing table. Sometimes we may require to add additional information, in that case, we do not require to create the whole database again.

Syntax:

ALTER TABLE table_name

ADD (Columnname_1 datatype,

Columnname_2 datatype,

…

Columnname_n datatype);

ALTER TABLE – DROP

DROP COLUMN is used for deleting unwanted columns from the table.

Syntax:

ALTER TABLE table_name

DROP COLUMN column_name;

ALTER TABLE-MODIFY

It is used to modify the existing columns in a table. Multiple columns can also be modified at once.

Syntax:

ALTER TABLE table_name

MODIFY column_name column_type(size);

Examples:

Let the name of the table be Student (Roll_no, Name). To ADD 2 columns AGE and COURSE to table Student.

ALTER TABLE Student ADD (AGE number(3),COURSE varchar(40));

- To modify the column COURSE in table Student

ALTER TABLE Student MODIFY COURSE varchar(20);

After running the above query maximum size of the Course Column is reduced to 20 from 40.

- To drop the column COURSE in table Student.

ALTER TABLE Student DROP COLUMN COURSE;

ALTER (RENAME): Sometimes we may want to rename our table to give it a more relevant name. For this purpose, we can use **ALTER TABLE** to rename the name of the table.

ALTER TABLE table_name RENAME TO new_table_name;

Columns can be also be given a new name with the use of **ALTER TABLE**.

ALTER TABLE table_name RENAME COLUMN old_name TO new_name;

Example: Change the name of column NAME to FIRST_NAME in table Student.

ALTER TABLE Student RENAME COLUMN NAME TO FIRST_NAME;

5) Explain SELECT statement in SQL and its variations. (5)

SELECT is the most commonly used statement in SQL. The SELECT Statement in SQL is used to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called the result set.
With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table. Though select clause is the first clause and is one of the last clauses of the select statement that the database server evaluates. The reason for this is that before we can determine what to include in the final result set, we need to know all of the possible columns that could be included in the final result set.
Basic Syntax:
SELECT column1,column2 FROM table_name
column1, column2: names of the fields of the table
table_name: from where we want to fetch
This query will return all the rows in the table with fields column1 , column2.

- To fetch the entire table or all the fields in the table:
  SELECT * FROM table_name;
- Query to fetch the fields ROLL_NO, NAME, AGE from the table Student:
  SELECT ROLL_NO, NAME, AGE FROM Student;

Using SELECT in INSERT INTO Statement
We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table.The use of this statement is similar to that of INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table. The different ways of using INSERT INTO SELECT statement are shown below:
Inserting all columns of a table: We can copy all the data of a table and insert it into a different table.
INSERT INTO first_table SELECT * FROM second_table;
first_table: name of the first table.
second_table: name of the second table.
We have used the SELECT statement to copy the data from one table and the INSERT INTO statement to insert it into a different table.

Inserting specific columns of a table: We can copy only those columns of a table that we want to insert into a different table.

Syntax:
INSERT INTO first_table(names_of_columns1) SELECT names_of_columns2 FROM second_table;
first_table: name of the first table.
second_table: name of the second table.
names of columns1: name of columns separated by comma (,) for table 1.
names of columns2: name of columns separated by comma (,) for table 2.
We have used the SELECT statement to copy the data of the selected columns only from the second table and the INSERT INTO statement to insert it in the first table.

Example:
INSERT INTO Student(ROLL_NO,NAME,Age) SELECT ROLL_NO, NAME, Age FROM
StudentDetails;

Output:
This query will insert the data in the columns ROLL_NO, NAME and Age of the table LateralStudent in the table Student and the remaining columns in the Student table will be filled by null which is the default value of the remaining columns.

Copying specific rows from a table: We can copy specific rows from a table to insert into another table by using the WHERE clause with the SELECT statement. We have to provide appropriate conditions in the WHERE clause to select specific rows.
INSERT INTO table1 SELECT * FROM table2 WHERE condition;
first_table: name of the first table.
second_table: name of the second table.
condition: condition to select specific rows.

INSERT INTO Student SELECT * FROM StudentDetails;

Output:
This query will insert all the data of the table StudentDetails in the table Student.

INSERT INTO Student SELECT * FROM StudentDetails WHERE Age = 18;
Output:
This query will select only the first row from table StudentDetails to insert into the table Student.

6) Explain the statements with examples i) UPDATE ii) DELETE (6)

UPDATE Statement
The UPDATE statement in SQL is used to update the data of an existing table in the database. We can update single columns as well as multiple columns using the UPDATE statement as per our requirement.
Basic Syntax
UPDATE table_name SET column1 = value1, column2 = value2,...
WHERE condition;
table_name: name of the table
column1: name of first , second, third column....
value1: new value for first, second, third column....
condition: condition to select the rows for which the values of columns need to be updated.
NOTE: In the above query the **SET** statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in **all** the rows will be updated. So the WHERE clause is used to choose the particular rows.

Example:
Updating single column: Update the column NAME and set the value to 'PRATIK' in all the rows where Age is 20.
UPDATE Student SET NAME = 'PRATIK' WHERE Age = 20;
Output:
This query will update the name field of those rows with age =20 to PRATHIK. If there are two rows with age=20 the name fields of those two rows will be updated to PRATHIK.

- **Updating multiple columns:** Update the columns NAME to 'PRATIK' and ADDRESS to 'SIKKIM' where ROLL_NO is 1.
- UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'SIKKIM' WHERE ROLL_NO = 1;
- **Excluding the WHERE clause:** If we exclude the WHERE clause from the update query then all of the rows will get updated.
  UPDATE Student SET NAME = 'PRATIK';
- Be Upon the execution of the above query, the name field of all the records will be updated to PRATIK.

DELETE Statement

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

Basic Syntax:

DELETE FROM table_name WHERE some_condition;

table_name: name of the table

some_condition: condition to choose a particular record.

Note: We can delete single as well as multiple records depending on the condition we provide in the WHERE clause. If we exclude the WHERE clause then all of the records will be deleted and the table will be empty.

Examples:

- **Deleting single record**: Delete the rows where NAME = 'Ram'. This will delete only the first row.
  DELETE FROM Student WHERE NAME = 'Ram';

Output:

The above query will delete only the record whose name field contains the value 'Ram'.

- **Deleting multiple records**: Delete the rows from the table Student where Age is 20. This will delete 2 rows(third row and fifth row).

DELETE FROM Student WHERE Age = 20;

Output:

The above query will delete the records with Age =20. If there are 3 records with age=20 all the three will be deleted.

- **Delete all of the records:** There are two queries to do this as shown below,

Query-1: DELETE FROM Student;

Query-2: DELETE * FROM Student;

Output:

All of the records in the table will be deleted, there are no records left to display. The table **Student** will become empty.

7) Explain any five constraints in SQL. (5) [any 5]

SQL Server contains the following types of constraints:

- Not Null Constraint
- Check Constraint
- Default Constraint
- Unique Constraint
- Primary Constraint
- Foreign Constraint

Not Null Constraint

A Not null constraint restricts the insertion of null values into a column. If we are using a Not Null Constraint for a column then we cannot ignore the value of this column during an insert of data into the table.

Column Level

Syntax

```
1. CREATE TABLE Table_Name
2. (
3.     Column_Name Datatype CONSTRAINT Constraint_Name NOT NULL,
```

```
4. );
```

Example

```
1. Create Table My_Constraint
2. (
3.    IID int NOT NULL,
4.    Name nvarchar(50) CONSTRAINT Cons_NotNull not null,
5.    Age int Not Null,
6. )
```

Table Level

Syntax

```
1. ALTER TABLE Table_Name
2. ALTER COLUMN Column_Name Datatype NOT NULL
```

Example

```
1. Alter Table My_Constraint
2. Alter Column IId int Not Null
```

Check Constraint

A Check constraint checks for a specific condition before inserting data into a table. If the data passes all the Check constraints then the data will be inserted into the table otherwise the data for insertion will be discarded. The CHECK constraint ensures that all values in a column satisfies certain conditions.

Column Level

Syntax

```
1. Create Table Table_Name
2. (
3.    Column_Name Datatype Constraint Constraint_Name Check(Condition)
4. )
```

Example

```
1. Create Table Constraint_

2. (
3.    IId int Constraint Constraint_Name Check(IId>100)
4. )
```

Table Level

Syntax

```
1. Alter Table Table_Name
2. Add Constraint Constraint_Name Check(Condition)
```

Example

1. Alter table Constraint_
2. Add constraint Cons_Name Check(IId>150)

Default Constraint

Specifies a default value for when a value is not specified for this column. If in an insertion query any value is not specified for this column then the default value will be inserted into the column.

Column Level

Syntax

```
1. Create Table Table_Name
2. (
3.    Column_Name DataType Constraint Constraint_Name Default(Value),
4. )
```

Example

```
1. Create Table My_Table1
2. (
3.    IId int default(1500),
4.    Name Nvarchar(50)Constraint Name_Default Default('Pankaj'),
5.    Age Int,
6.    Salary Int Default(100)
7. )
```

Table Level

Syntax

```
1. Alter Table Tabel_Name
2. Add Constraint Constraint_Name Default(Value) for[Column_Name]
```

Example

```
1. Alter Table My_Table1
2. Add Constraint cons_Default Default(40) for[Age]
```

Unique Constraint

It ensures that each row for a column must have a unique value. It is like a Primary key but it can accept only one null value. In a table one or more column can contain a Unique Constraint.

Column Level

Syntax

```
1. Create Table Table_Name
```

```
2. (
3.    Column_Name Datatype Constraint Constraint_Name Unique
4. )
```

Example

```
1. Create Table MY_Tab
2. (
3.    IId int constraint Unique_Cons Unique ,
4.    Name nvarchar(50)
5. )
```

Table Level

Syntax

```
1. Alter Table_Name
2. Add Constraint Constraint_Name Unique(Column_Name)
```

Example

```
1. Alter Table My_Tab
2. Add Constraint Unique_Cons_ Unique(Name)
```

Primary Key Constraint

A Primary key uniquly identifies each row in a table. It cannot accept null and duplicate data. One or more of the columns of a table can contain a Primary key.

Column Level

Syntax

```
1. Create Table Table_Name
2. (
3.    Column_Name Datatype Constraint Constraint_Name Primary Key,
4. )
```

Example

```
1. Create Table Employee
2. (
3.    IId int constraint Const_primary_IId primary key,
4.    Name nvarchar(50)
5. )
```

Table Level

Syntax

1. Alter Table Table_Name
2. Add constraint Constraint_Name Primary Key(Column_Name)

Example

1. Alter Table Employee
2. Add constraint Constraint_Name Primary Key(IId)

Foreign Key Constraint

A Foreign Key is a field in a database table that is a Primary key in another table. A Foreign key creates a relation between two tables. The first table contains a primary key and the second table contains a foreign key.

Column Level

Syntax

1. Create Table Table_Name
2. (
3. Column_Name Datatype Constraint Constraint_Name References Reference_Table_Name(Reference_Column_Name)
4.)

Example

1. Create Table Employee_
2. (
3. IId int constraint Cons_Reference References My_Constraint(IId),
4. Age int,
5. Salary int
6.)

Table Level

Syntax

1. ALTER TABLE Table_Name
2. ADD CONSTRAINT Constraint_Name FOREIGN KEY(Column_Name)
3. REFERENCES Reference_Table (Column_Name)

Example

1. ALTER TABLE Employee_
2. ADD CONSTRAINT Cons_Emp_Foreign FOREIGN KEY(IId)
3. REFERENCES My_Constraint(IId)

8) Explain the concept of Primary Key and Foreign Key with the help of examples. (5)

A **primary key** constrain is a column or group of columns that uniquely identifies every row in the table of the relational database management system. It cannot be a duplicate, meaning the same value should not appear more than once in the table.

A table can not have more than one primary key. Primary key can be defined at the column or the table level. If you create a composite primary key, it should be defined at the table level.

Syntax:

Below is the syntax of Primary Key:

```
CREATE TABLE <Table-Name>
(
Column1 datatype,
Column2 datatype,  PRIMARY KEY (Column-Name)
.
);
```

Here,

- Table_Name is the name of the table you have to create.
- Column_Name is the name of the column having the primary key.

Example:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   PRIMARY KEY (ID)
);
```

Foreign key is a column that creates a relationship between two tables. The purpose of the Foreign key is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table. Every relationship in the database should be supported by a foreign key.

Syntax:

Below is the syntax of Foreign Key:

```
CREATE TABLE <Table Name>(
column1    datatype,
column2    datatype,
constraint (name of constraint)
FOREIGN KEY [column1, column2...]
REFERENCES [primary key table name] (List of primary key table column) ...);
```

Here,

- The parameter Table Name indicates the name of the table that you are going to create.
- The parameters column1, column2… depicts the columns that need to be added to the table.
- Constraint denotes the name of constraint you are creating.
- References indicate a table with the primary key.

Ex:

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   OrderNumber int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

9) Explain ORDER BY clause and GROUP BY clause with the help of examples. (6)

ORDER BY

The ORDER BY clause sorts the result-set in ascending or descending order. It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order.

Syntax:

```
SELECT column1, column2   FROM table_name  WHERE condition
ORDER BY column1, column2... ASC|DESC;
```

Where

ASC: It is used to sort the result set in ascending order by expression.

DESC: It sorts the result set in descending order by expression.

Example: SELECT * FROM CUSTOMER ORDER BY NAME;

The above query will display the records of the table Customer in ascending order of Name

```
SELECT *  FROM CUSTOMER   ORDER BY NAME DESC;
```

The above query will display the records of the table Customer in descending order of Name

GROUP BY

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

10) Write a note on Granting and Revoking permissions.(5) [refer notes for detailed ans.]

GRANTING AND REVOKING PERMISSIONS:

The objects created by one user are not accessible by another user unless the owner of those objects gives such permission to other users. These permissions can be given by using the Grant statement. Once a user can grant permission to another user if he is the owner of the object or has permission to grant access to other users.

Granting permissions using the GRANT statement:

The Grant statement provides various types of access to database objects such as tables, views, and sequences.

Syntax:GRANT {object privileges}
ON objectname
TO username
[WITH GRANT OPTION];

Object Privileges:

Each object privilege that is granted authorizes the grantee to perform some operation on the object. The user can grant all the privileges or grant only specific object privileges.

- ALTER
- DELETE
- INDEX
- INSERT
- SELECT
- UPDATE

Different ways of granting privileges to the users:

- **Granting SELECT Privilege to a User in a Table**: GRANT SELECT ON Users TO 'Amit'@'localhost;
- **Granting more than one Privilege to a User in a Table**: GRANT SELECT, INSERT, DELETE, UPDATE ON Users TO 'Amit'@'localhost;
- **Granting All the Privileges to a User in a Table**: GRANT ALL ON Users TO 'Amit'@'localhost;
- **Granting a Privilege to all Users in a Table**: GRANT SELECT ON Users TO '*'@'localhost;

In the above example, the "*" symbol is used to grant select permission to all the users of the table "users".

Checking the Privileges Granted to a User: SHOW GRANTS FOR 'Amit'@localhost';

Different ways of revoking privileges from a user:

1. **Revoking SELECT Privilege to a User in a Table**: REVOKE SELECT ON users TO 'Amit'@localhost';
2. **Revoking more than Privilege to a User in a Table**: REVOKE SELECT, INSERT, DELETE, UPDATE ON Users TO 'Amit'@'localhost;
3. **Revoking All the Privilege to a User in a Table**: REVOKE ALL ON Users TO 'Amit'@'localhost;
4. **Revoking a Privilege to all Users in a Table**: REVOKE SELECT ON Users TO '*'@'localhost;

11) Briefly explain any five SQL datatype.(5) [any 5]

String Data types:

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1

VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
TINYTEXT	Holds a string with a maximum length of 255 characters

Numeric Data Types

Data type	Description
BIT(size)	A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1.
SMALLINT(size)	A small integer. The signed range is from -32768 to 32767. The unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255)
BIGINT(size)	A large integer. The signed range is from -9223372036854775808 to 9223372036854775807. The unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255)

Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
TIME(fsp)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in the four-digit format: 1901 to 2155, and 0000.

12) Explain SQL Arithmetic and Comparison operators. (6)

SQL Arithmetic Operators

Assume **'variable a'** holds 10 and **'variable b'** holds 20, then

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	a + b will give 30
- (Subtraction)	Subtracts the right-hand operand from the left-hand operand.	a - b will give -10
* (Multiplication)	Multiplies values on either side of the operator.	a * b will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	b / a will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns the remainder.	b % a will give 0

SQL Comparison Operators

Assume **'variable a'** holds 10 and **'variable b'** holds 20, then −

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then the condition becomes true.	(a = b) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then the condition becomes true.	(a != b) is true.
<>	Checks if the values of two operands are equal or not, if values are not equal then the condition becomes true.	(a <> b) is true.
>	Checks if the value of the left operand is greater than the value of the right operand, if yes then the condition becomes true.	(a > b) is not true.
<	Checks if the value the of left operand is less than the value of the right operand, if yes then the condition becomes true.	(a < b) is true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes then the condition becomes true.	(a >= b) is not true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes the hen condition becomes true.	(a <= b) is true.
!<	Checks if the value of the left operand is not less than the value of the right operand, if yes then the condition becomes true.	(a !< b) is false.
!>	Checks if the value of the left operand is not greater than the value of the right operand, if yes the hen condition becomes true.	(a !> b) is true.

13) Explain SQL Logical Operators. (4)

Sl. No.	Operator & Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	The ANY operator is used to compare a value to any applicable value in the list as per the condition.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.
IN	The IN operator is used to compare a value to a list of literal values that have been specified.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard operators.

NOT	The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.**
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

14) Explain any five Aggregate functions. (5)

Aggregate functions: These functions are used to do operations from the values of the column and a single value is returned. AVG(), COUNT(), FIRST(), LAST(), MAX(), MIN(), and SUM() are the aggregate functions in SQL.

FIRST(): The FIRST() function returns the first value of the selected column.
Syntax:
SELECT FIRST(column_name) FROM table_name;
Example: SELECT FIRST(MARKS) AS MarksFirst FROM Students;

MarksFirst
90

LAST(): The LAST() function returns the last value of the selected column. It can be used only in MS ACCESS.
Syntax: SELECT LAST(column_name) FROM table_name;
Example: SELECT LAST(MARKS) AS MarksLast FROM Students;
Output:

MarksLast
85

MAX(): The MAX() function returns the maximum value of the selected column.
Syntax: SELECT MAX(column_name) FROM table_name;
Example: SELECT MAX(MARKS) AS MaxMarks FROM Students;
Output:

MaxMarks
95

MIN(): The MIN() function returns the minimum value of the selected column.
Syntax: SELECT MIN(column_name) FROM table_name;
Example: SELECT MIN(AGE) AS MinAge FROM Students;
Output:

MarksLast
85

SUM(): The SUM() function returns the sum of all the values of the selected column.
Syntax: SELECT SUM(column_name) FROM table_name;
Example: SELECT SUM(MARKS) AS TotalMarks FROM Students;
Output:

TotalMarks
400

15) Explain any five Scalar functions. (5)

Scalar functions: These functions are based on user input, and return a single value. UCASE(), LCASE(), MID(), LEN(), ROUND(), NOW(), FORMAT() are some scalar functions in SQL.

UCASE(): It converts the value of a field to uppercase.
Syntax: SELECT UCASE(column_name) FROM table_name;
Example: SELECT UCASE(NAME) FROM Students WHERE ID=2;
Output:

UCASE(NAME)
SHAHEEN

LCASE(): It converts the value of a field to lowercase.
Syntax: SELECT LCASE(column_name) FROM table_name;
Example: SELECT LCASE(NAME) FROM Students WHERE ID=5;
Output:

LCASE(NAME)
biju

MID(): The MID() function extracts texts from the text field.
Syntax: SELECT MID(column_name,start,length) AS some_name FROM table_name;
specifying length is optional here, and start signifies start position (starting from 1)
Queries:
Fetching the first four characters of names of students from the Students table.
SELECT MID(NAME,1,4) FROM Students WHERE ID=4;
Output:

MID(NAME)
Stev

LEN(): The LEN() function returns the length of the value in a text field.
Syntax:
SELECT LENGTH(column_name) FROM table_name;
Example:
SELECT LENGTH(NAME) FROM Students WHERE ID=4;
Output:

LEN(NAME)
6

FORMAT(): The FORMAT() function is used to format how a field is to be displayed.
Syntax:
SELECT FORMAT(column_name,format) FROM table_name;
Example: SELECT NAME, FORMAT(Now(),'YYYY-MM-DD') AS Date FROM Students WHERE ID=1;
Output:

DATE
2022-07-19

16) Write a note on JOIN Operation in SQL.(6) [refer notes for example and detailed ans]

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN

- RIGHT JOIN
- FULL JOIN

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

17) What is Views in SQL? Explain with an example. (6)

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

SQL CREATE VIEW Examples

The following SQL creates a view that shows all customers from Brazil:

Example

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
```

```
FROM Customers
WHERE Country = 'Brazil';
```

UNIT-IV

Two marks questions

1) What is SAS?

SAS[Statistical Analysis Software.] is a command-driven statistical software suite widely used for statistical data analysis and visualization.

2) Mention the parts of SAS architecture?

- Client Tier
- Middle Tier
- Back tier

3) List any four datatypes for SAS data sets.

- BIGINT
- SMALLINT
- DATE
- TIME

4) Mention two types of SAS variables.

- Numeric variables
- Character variables

5) Write the syntax of DATA statement.

```
DATA data_set_name;          #Give a name to the dataset
INPUT var1,var2, var3;       #Declare variables in the dataset.
NEW_VAR;                     #Define new variables.
LABEL;                       #Give variables a label
DATALINES;                   #Provide data
RUN;
```

6) What is permanent SAS library?

A permanent SAS library is one that resides on the external storage medium of your computer and is not deleted when the SAS session terminates.

7) What is temporary SAS library?

A temporary SAS library is one that exists only for the current SAS session or job.

8) Differentiate informats with formats in SAS.

Informats is used to tell SAS how to read a variable whereas Formats is used to tell SAS how to display or write values of a variable

9) Mention two methods to import external data into SAS.

a) PROC IMPORT
b) Get External File - INFILE

10) Give the syntax for sorting data sets in SAS.

PROC SORT DATA = original dataset OUT = Sorted dataset;

BY variable name;

11) Write an example to concatenate two data sets in SAS. [refer notes]

12) List the various categories of functions in SAS.

- Mathematical
- Date and Time
- Character
- Truncation

13) Mention any two-character built-in functions in SAS.

- lowcse_
- upcase_

14) Mention any two truncation built-in functions in SAS.

- ceil_
- floor_

15) List any two date and time built-in functions in SAS.

DATE()	**returns the current date as a SAS date value**
DATETIME()	returns the current date and time of day
DAY(date)	returns the day of the month from a SAS date value
MONTH(date)	returns the month from a SAS date value

16) List any two mathematical built-in functions in SAS.

MAX(argument,argument, …)	returns the largest value of the numeric arguments
MIN(argument,argument, …)	returns the smallest value of the numeric arguments

17) What is the purpose of WHERE statement in SAS? Write its syntax.

The WHERE statement is a substitute to IF statement when it comes to subsetting a data set.

Syntax:
WHERE (condition is true) => It refers to subsetting a dataset.

18) List any two comparison operators used in SAS with their purpose.

Symbol	Mnemonic	Meaning
=	EQ	equals
^= or ~=	NE	not equal
>	GT	greater than
<	LT	less than

19) Write the syntax of DO statement in SAS.

```
DO index-variable = start TO stop BY increment;
  action statements;
END;
```

20) Write an example for if then delete statement in SAS.

```
Data readin;
Input R_Num Subj1-Subj3;
cards;
112 21 22 23
105 23 24 26
95 25 25 26
125 26 26 24
122 15 25 16
86 26 16 15
;
Data readin1;
Set readin;
IF R_Num LT 100 THEN DELETE;
run;
proc print data=readin1;
run;
```

Obs	R_Num	Subj1	Subj2	Subj3
1	112	21	22	23
2	105	23	24	26
3	125	26	26	24
4	122	15	25	16

Long answer questions

1) Briefly explain SAS architecture with diagram. (6)

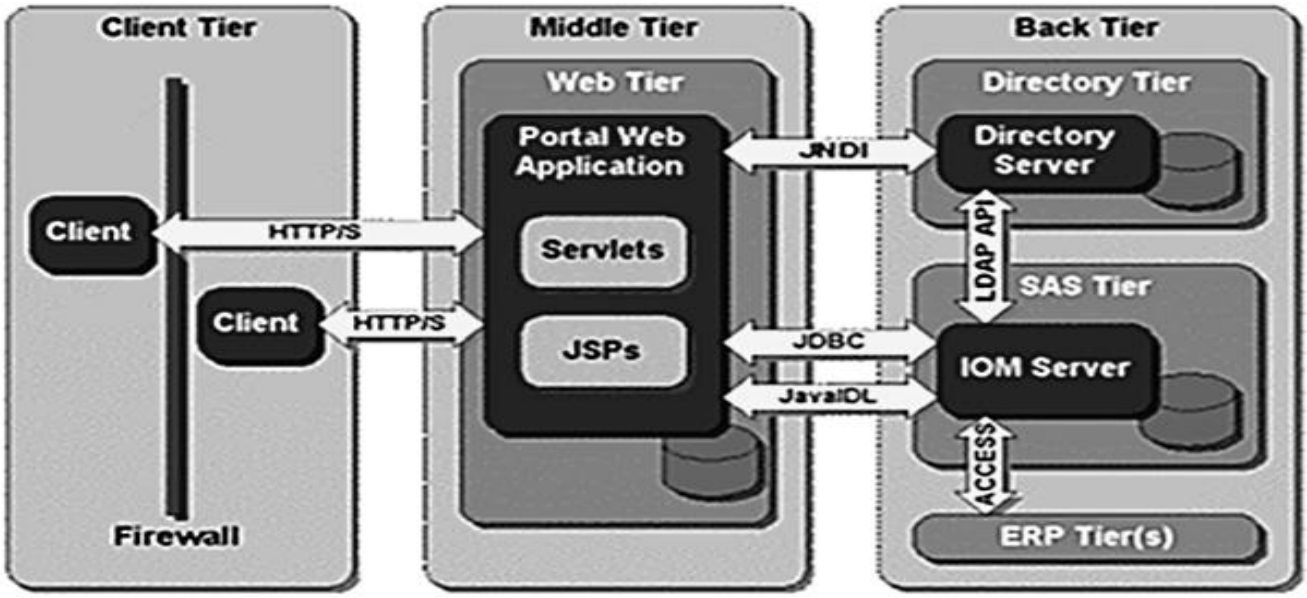
SAS architecture is divided mainly of three parts:

- Client Tier
- Middle Tier
- Back tier

Client tier: Client tier is where the application is installed on a machine, where the user is sitting. It consists of the components which are used to view the portal and its content. It also includes a standard web browser

that is used to interact with the portal over standard HTTP or HTTPS protocol. It also helps you to make the SAS web application firewall friendly.

Middle tier: The middle tier offers a centralized access point for enterprise information. All access to content is processed by components operating of this tier. The separation of the business logic with display logic helps you to leverage the logic of the middle tier. Moreover, centralized points of access make it easier to enforce security rules, administer the portal and manage code changes.



Architecture of SAS

The middle tier hosts the following functions:

SAS Information Delivery Portal Web Application: It is the collection of JSP, Java servlets, JavaBeans, and other classes and resources. These components help you to access information stored in the enterprise directory to create a customizable interface for the user.

Servlet Engine: The servlet engine is also called a servlet container. It is responsible for managing the SAS Information Delivery Portal Web Application. The servlet engine offers a run time environment. It provides concurrency, deployment, lifecycle management, etc.

Web server: Web server offers service for the servlet engine which can be used to host website. This should be accessed using the portal.

Back Tier: The back tier is an area where the data and computation servers run which may contain business objects. It is an enterprise directory server. The enterprise directory server maintains metadata about content which is located throughout the enterprise.

2) How do you use SAS? Explain its various steps. (6)

To effectively use SAS software you need to follow four steps which are: **Access Data, Management Data, Analyze, Present**

Access Data: SAS allows you to access data in any desired format that you want. You can access data that is stored anywhere, whether it is in a file on your system or data that is stored in another database system. It can be oracle file, SAS database file, Raw Database file or a simple XLS /CSV file. It will help you to access this data with ease.

Manage Data: SAS offers great data management capabilities. You can subset/slice data based on certain conditions, create variable, clean & validate data. There are other tools which allow you to perform the same task. However, SAS helps you to perform this job with ease.
SAS has well-defined libraries and processes which makes the programming process easy. Moreover, creating variable or subset data is just one step process. This saves you from writing complex algorithms by just a single line of code.

Analyze: You can do various kinds of analyze using SAS:

- It checks Frequency of Mean calculation
- Regression and Forecasting
- Decision Tree

All these analyzes can easily handle by SAS. It is the best tool for accurate forecasting.

Present: If you visualize data correctly, it is effortless for the audience to relate to it. It is essential that your tool present the data in a suitable manner. That's what SAS does for you. It has excellent presentation capabilities. You can:
1. List reports
2. Summary reports
3. Graph reports
4. Print reports

3) Explain different datatypes for SAS data sets. (4/5/6)

Data Type Definition (Keyword)	SAS Data Set Data Type	Description	Data Type Returned
BIGINT	DOUBLE	64-bit double precision, floating-point number. **Note:** There is potential for loss of precision.	DOUBLE
CHAR(n)	CHAR(n)	Fixed-length character string. **Note:** Cannot contain ANSI SQL null values.	CHAR(n)
DATE	DOUBLE	64-bit double precision, floating-point number. By default, applies the DATE9 SAS format.	DOUBLE
DECIMAL|NUMERIC(p,s)	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
DOUBLE	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
FLOAT(p)	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
INTEGER	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
NCHAR(n)	CHAR(n)	Fixed-length character string. By default, sets the encoding to Unicode UTF-8.	CHAR(n)

NVARCHAR(n)	CHAR(n)	Fixed-length character string. By default, sets the encoding to Unicode UTF-8.	CHAR(n)
REAL	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
SMALLINT	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
TIME(p)	DOUBLE	64-bit double precision, floating-point number. By default, applies the TIME8 SAS format.	DOUBLE
TIMESTAMP(p)	DOUBLE	64-bit double precision, floating-point number. By default, applies the DATETIME19.2 SAS format.	DOUBLE
TINYINT	DOUBLE	64-bit double precision, floating-point number.	DOUBLE
VARCHAR(n)	CHAR(n)	Fixed-length character string. **Note:** Cannot contain ANSI SQL null values.	CHAR(n)

4) With syntax and example explain DATA statement. (5)

Data step loads the needed data set into SAS memory and finds the correct variables of the data set. It also captures the records. We can use data steps to:

- Enter data into SAS data sets
- Compute Values
- Check or correct data
- Produce new data sets

The syntax for DATA statement is:

Syntax

```
DATA data_set_name;          #Give a name to the dataset
INPUT var1,var2, var3;       #Declare variables in the dataset.
NEW_VAR;                     #Define new variables.
LABEL;                       #Give variables a label
DATALINES;                   #Provide data
RUN;
```

Example:

Following example show how to define a variable, naming the data set, creating new variables and entering the data. In this example, you can see that string variable have a $ at the end, and numeric values are without it.

```
INPUT ID $ NAME $ SALARY DEPARTMENT $;
comm = SALARY*1.50;
LABEL ID = 'Emp_ID' comm = 'COMMISION';
DATALINES;
1 Tom 5000 IT
2 Harry 6000 Operations
3 Michelle 7000 IT
```

```
4 Dick 8000 HR
5 John 9000 Finance
;
RUN;
```

5) Explain the necessary steps required to write a SAS program with an example. (6) [REFER NOTES]

6) Write a note on Informats in SAS. Give example. (4)

In SAS, formats and informats are pre-defined patterns for interpreting or displaying data values. They use a common set of pattern codes. There are three main types of built-in informats in SAS: character, numeric, and date. Generically, the informat/format codes follow these patterns:

Type	Format/Informat Name	What it Does
Character	**$w.**	Reads in character data of length **w.**
Numeric	**w.d**	Reads in numeric data of length w with d decimal points
Date	**MMDDYYw.**	Reads in a date of length w. assuming the order MDY

The '$' indicates a character informat. INFORMAT refers to the sometimes-optional SAS informat name. The 'w' indicates the width (bytes or number of columns) of the variable. The 'd' is used for numeric data to specify the number of digits to the right of the decimal place. All informats must contain a decimal point(.)

Example:

```
DATA Employee_Info;
input Emp_ID Emp_Name$ Emp_Vertical$ DOJ;
INFORMAT DOJ ddmmyy10.;
datalines;
101 Mak SQL 18/08/2013
102 Rama SAS 25/06/2015
103 Priya Java 21/02/2010
104 Karthik Excel 19/05/2007
105 Mandeep SAS 11/09/2016
;
Run;
PROC PRINT DATA=Employee_Info;
Run;
```

7) Write a note on Formats in SAS. Give example. (4)

Informats are the instructions for reading data, whereas formats are the instructions used to display or output data. Defining a format for a variable is how you tell SAS to display the values in the variable. Formats can be used in both Data Steps and PROC Steps whereas Informat can be used only in Data Steps. Formats are grouped into the same three classes as informats (character, numeric, and date-time) and also always contain a dot.

The general form of a format statement is:

- FORMAT variable-name FORMAT-NAME.;

Let us go back to our code having dataset Employee_Info to see if we can display the date correctly using FORMAT command.

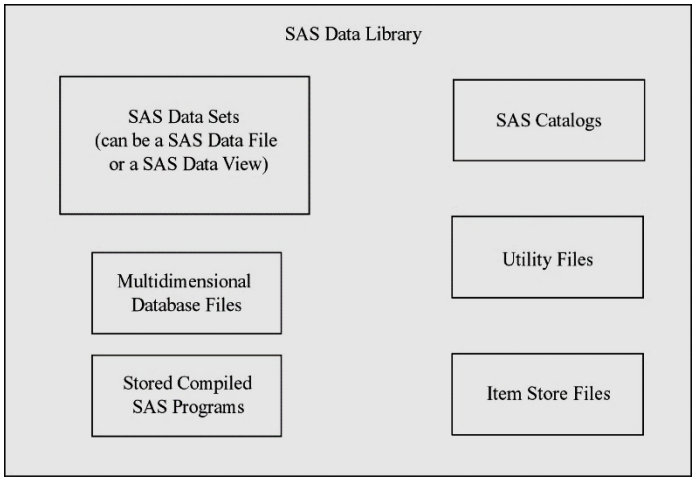
```
DATA Employee_Info;
input Emp_ID Emp_Name$ Emp_Vertical$ DOJ;
INFORMAT DOJ ddmmyy10.;
FORMAT DOJ ddmmyy10.;
datalines;
101 Mak SQL 18/08/2013
102 Rama SAS 25/06/2015
103 Priya Java 21/02/2010
104 Karthik Excel 19/05/2007
105 Mandeep SAS 11/09/2016
;
Run;

PROC PRINT DATA=Employee_Info;
Run;
```

8) Mention the types of files in a SAS library with diagram. (4)

SAS files can be any of the following file types:

- SAS data set (SAS data file or SAS view)
- SAS catalog
- stored compiled SAS program
- SAS utility file
- access descriptors
- multi-dimensional database files such as MDDB, FDB, and DMDB files
- item store files



9) What is permanent and temporary SAS library? (4)

A permanent SAS library is one that resides on the external storage medium of your computer and is not deleted when the SAS session terminates. Permanent SAS libraries are stored until you delete them. The library is available for processing in subsequent SAS sessions. When working with files in a permanent SAS library, you generally specify a libref as the first part of a two-level SAS filename. The libref tells SAS where to find or store the file.

A temporary SAS library is one that exists only for the current SAS session or job. SAS files that are created during the session or job are held in a special work space that might or might not be an external storage

medium. This work space is generally assigned the default libref WORK. Files in the temporary WORK library can be used in any DATA step or SAS procedure during the SAS session, but they are typically not available for subsequent SAS sessions. Normally, you specify that data sets be stored in or retrieved from this library by specifying a one-level name. Files held in the WORK library are deleted at the end of the SAS session if it ends normally.

10) Write a note on importing a delimited file into SAS. (5)

Importing a Delimited File into SAS

1. Importing a Tab-Delimited File into SAS

The program below is similar to the code of importing excel file. The only difference is DBMS = DLM and delimter = '09'x.

```
PROC IMPORT DATAFILE= "c:\deepanshu\sampledata.txt"
OUT= outdata
DBMS=dlm
REPLACE;
delimiter='09'x;
GETNAMES=YES;
RUN;
```

2. Importing a Comma-Delimited File with TXT extension

To get comma separated file with a txt extension into SAS, specify delimeter = ','

```
PROC IMPORT DATAFILE= "c:\deepanshu\sampledata.txt"
OUT= outdata
DBMS=dlm
REPLACE;
delimiter=',';
GETNAMES=YES;
RUN;
```

3. Importing a Comma-Delimited File with CSV extension

To get comma separated file into SAS, specify DBMS= CSV

```
PROC IMPORT DATAFILE= "c:\deepanshu\sampledata.txt"
OUT= outdata
DBMS=csv
REPLACE;
GETNAMES=YES;
RUN;
```

4. Importing a Space-Delimited File

To extract a space delimited file, specify delimiter = '20'x

```
PROC IMPORT DATAFILE= "c:\deepanshu\sampledata.txt"
OUT= outdata
DBMS=dlm
REPLACE;
delimiter='20'x;
GETNAMES=YES;
RUN;
```

5. Importing a file containing multiple delimiter

If two or more delimiters, such as comma and tabs, quote them following delimiter = option

```
PROC IMPORT DATAFILE= "c:\deepanshu\sampledata.txt"
OUT= outdata
DBMS=dlm
REPLACE;
delimiter=',''09'x ';
GETNAMES=YES;
RUN;
```

11) How do you import an external file in SAS? Explain. (5)

In SAS, there is one more method called **INFILE** to import an external file. It's a manual method of importing an external file as you need to specify variables and its types and length.

1. Reading a CSV File

INFILE statement - To specify path where data file is saved.
DSD - To set the default delimiter from a blank to comma.
FIRSTOBS=2 : To tell SAS that first row contains variable names and data values starts from second row.

```
data outdata;
infile 'c:\users\deepanshu\documents\book1.csv' dsd firstobs=2;
input id age gender $ dept $;
run;
```

2. Reading a TAB Delimited File

We can use DLM='09'x to tell SAS that we are going to import a tab delimited file. The TRUNCOVER statement tells SAS to assign the raw data value to the variable even if the value is shorter than expected by the INPUT statement.

```
infile 'c:\deepanshu\dummydata.txt' DSD dlm='09'x truncover;
  input employee :$30. DOJ :mmddyy8. state :$20.;
run;
```

12) How do you concatenate data sets in SAS? Explain. (5)

Multiple SAS data sets can be concatenated to give a single data set using the **SET** statement. The total number of observations in the concatenated data set is the sum of the number of observations in the original data sets. The order of observations is sequential. All observations from the first data set are followed by all observations from the second data set, and so on. Ideally all the combining data sets have same variables, but in case they have different number of variables, then in the result all the variables appear, with missing values for the smaller data set. The basic syntax for SET statement in SAS is –

SET data-set 1 data-set 2 data-set 3.....;

Here **data-set1,data-set2** are dataset names written one after another.

Example

Consider the employee data of an organization which is available in two different data sets, one for the IT department and another for Non-It department. To get the complete details of all the employees we concatenate both the data sets using the SET statement shown as below.

```
DATA ITDEPT;
INPUT empid name $ salary ;
DATALINES;
1 Rick 623.3
3 Mike 611.5
```

```
6 Tusar 578.6
;
RUN;

DATA NON_ITDEPT;
INPUT empid name $ salary ;
DATALINES;
2 Dan 515.2
4 Ryan 729.1
5 Gary 843.25
7 Pranab 632.8
8 Rasmi 722.5
RUN;

DATA All_Dept;
SET ITDEPT NON_ITDEPT;
RUN;

PROC PRINT DATA = All_Dept;
RUN;
```

13) How do you merge data sets in SAS? Explain. (5)

Multiple SAS data sets can be merged based on a specific common variable to give a single data set. This is done using the **MERGE** statement and **BY** statement. The total number of observations in the merged data set is often less than the sum of the number of observations in the original data sets. It is because the variables form both data sets get merged as one record based when there is a match in the value of the common variable.

There are two Prerequisites for merging data sets given below −

- input data sets must have at least one common variable to merge on.
- input data sets must be sorted by the common variable(s) that will be used to merge on.

The basic syntax for MERGE and BY statement in SAS is −

MERGE Data-Set 1 Data-Set 2

BY Common Variable

Here,

- Data-set1,Data-set2 are data set names written one after another.
- Common Variable is the variable based on whose matching values the data sets will be merged.

Example

Consider two SAS data sets one containing the employee ID with name and salary and another containing employee ID with employee ID and department. In this case to get the complete information for each employee we can merge these two data sets. The final data set will still have one observation per employee but it will contain both the salary and department variables.

Data set 1

ID NAME SALARY

1 Rick 623.3

2 Dan 515.2

3 Mike 611.5

4 Ryan 729.1

5 Gary 843.25

6 Tusar 578.6

7 Pranab 632.8

8 Rasmi 722.5

Data set 2

ID DEPT

1 IT

2 OPS

3 IT

4 HR

5 FIN

6 IT

7 OPS

8 FIN

Merged data set

ID NAME SALARY DEPT

1 Rick 623.3 IT

2 Dan 515.2 OPS

3 Mike 611.5 IT

4 Ryan 729.1 HR

5 Gary 843.25 FIN

6 Tusar 578.6 IT

7 Pranab 632.8 OPS

8 Rasmi 722.5 FIN

The above result is achieved by using the following code in which the common variable (ID) is used in the BY statement. Please note that the observations in both the datasets are already sorted in ID column.

```
DATA SALARY;
INPUT empid name$ salary ;
DATALINES;
1 Rick 623.3
2 Dan 515.2
3 Mike 611.5
4 Ryan 729.1
5 Gary 843.25
6 Tusar 578.6
7 Pranab 632.8
8 Rasmi 722.5;
DATA DEPT;
RUN;
```

```
INPUT empid DEPT$ ;
DATALINES;
1 IT
2 OPS
3 IT
4 HR
5 FIN
6 IT
7 OPS
8 FIN
;
RUN;
```

```
DATA All_details;
MERGE SALARY DEPT;
BY (empid);
RUN;
PROC PRINT DATA =
All_details;
RUN;
```

14) With syntax and example, Explain the concept of sorting data sets in SAS.(5)

Data sets in SAS can be sorted on any of the variables present in them. This helps both in data analysis and performing other options like merging etc. Sorting can happen on any single variable as well as multiple variables. The SAS procedure used to carry out the sorting in SAS data set is named **PROC SORT**. The result after sorting is stored in a new data set and the original data set remains unchanged.

The basic syntax for sort operation in data set in SAS is −

PROC SORT DATA = original dataset OUT = Sorted dataset;
BY variable name;

Following is the description of the parameters used −

- variable name is the column name on which the sorting happens.
- Original dataset is the dataset name to be sorted.
- Sorted dataset is the dataset name after it is sorted.

Example

Let's consider the following SAS data set containing the employee details of an organization. We can sort the data set on salary by using the code given below.

```
DATA Employee;
INPUT empid name $ salary DEPT$ ;
DATALINES;
1 Rick 623.3 IT
2 Dan 515.2 OPS
3 Mike 611.5 IT
4 Ryan 729.1 HR
5 Gary 843.25 FIN
6 Tusar 578.6 IT
7 Pranab 632.8 OPS
8 Rasmi 722.5 FIN
;
```

```
RUN;

PROC SORT DATA = Employee OUT = Sorted_sal ;
BY salary;
RUN ;

PROC PRINT DATA = Sorted_sal;
RUN ;
```

15) Explain different mathematical functions in SAS. (5)

These are the functions used to apply some mathematical calculations on the variable values.
Examples: The below SAS program shows the use of some important mathematical functions.

```
data Mathfunctions;
v1=21; v2=42; v3=13; v4=10; v5=29;

/* Get Maximum value */
max_val = MAX(v1,v2,v3,v4,v5);

/* Get Minimum value */
min_val = MIN (v1,v2,v3,v4,v5);

/* Get Median value */
med_val = MEDIAN (v1,v2,v3,v4,v5);

/* Get a random number */
rand_val = RANUNI(0);

/* Get Square root of sum of the values */
SR_val= SQRT(sum(v1,v2,v3,v4,v5));

proc print data = Mathfunctions noobs;
run;
```

When the above code is run, we get the following output –

v1	v2	v3	v4	v5	max_val	min_val	med_val	rand_val	SR_val
21	42	13	10	29	42	10	21	0.85005	10.7238

16) With syntax and programming example, explain DO statement in SAS. (6)

The Do statement is the primary control block of DATA Step coding. It answers the essential questions of when and how often a block of code is to be executed.

Three types of do Groups

1. Iterative
2. Do until
3. Do while

The DO statement designates a group of statements that are to be executed as a unit, usually as a part of IF-THEN/ELSE statements.

The iterative DO statement executes a group of statements repetitively based on the value of an index variable. If you specify an UNTIL clause or a WHILE clause, then the execution of the statements is also based on the condition that you specify in the clause.

The DO UNTIL statement executes a group of statements repetitively until the condition that you specify is true. The condition is checked after each iteration of the loop.

Syntax

DO index-variable = start TO stop BY increment;
action statements;
END;

The DO WHILE statement executes a group of statements repetitively as long as the condition that you specify remains true. The condition is checked before each iteration of the loop.

Example1:

```
DATA NEW;
DO I= 1 TO 100;
OUTPUT; /*writes current observation to the SAS data set*/
END;
RUN;
PROC PRINT DATA=NEW;
RUN;
```

17) With syntax and programming example, explain any two forms of IF statement in SAS. (6)

If Statement:

Comparison Operators used while using conditional statements

Symbol	Mnemonic	Meaning
=	EQ	equals
^= or ~=	NE	not equal
>	GT	greater than
<	LT	less than
>=	GE	greater than or equals
<=	LE	less than or equals
in	IN	selecting multiple values

Syntax
IF (condition is true) => It means subsetting a dataset.

Example:

```
Data readin;
Input R_Num Subj1-Subj3;
cards;
112 21 22 23
105 23 24 26
95 25 25 26
125 26 26 24
122 15 25 16
86 26 16 15
;

Data readin1;
Set readin;
IF R_Num GE 100;
run;
```

Obs	R_Num	Subj1	Subj2	Subj3
1	112	21	22	23
2	105	23	24	26
3	125	26	26	24
4	122	15	25	16

IF-THEN DELETE:

Syntax:
IF (condition is true) THEN (delete the given statements);

***Example*:**

```
Data readin;
Input R_Num Subj1-Subj3;
cards;
112 21 22 23
105 23 24 26
95 25 25 26
125 26 26 24
122 15 25 16
86 26 16 15
;
Data readin1;
Set readin;
IF R_Num LT 100 THEN DELETE;
run;
```

Obs	R_Num	Subj1	Subj2	Subj3
1	112	21	22	23
2	105	23	24	26
3	125	26	26	24
4	122	15	25	16

18) With syntax and programming example, explain where statement in SAS. (6)

WHERE Expression : The WHERE statement is a substitute to IF statement when it comes to subsetting a data set.

Syntax:
WHERE (condition is true) => It refers to subsetting a dataset.

Example: Suppose you want to select only section A students. You need to filter Section variable equals to A using where clause.

```
data readin;
input name $ Section $ Score;
cards;
Raj  A 80
Atul A 77
Priya B 45
Sandeep A 95
Rahul C 84
Shreya C 44
;
run;

data readin1;
set readin;
where Section EQ "A";
run;
proc print data=readin1;
run;
```

where section EQ “A” => This would tell SAS to select only section which is equals to values “A”. You can also write where section = “A”. This statement serves the same purpose.

Obs	name	Section	Score
1	Raj	A	80
2	Atul	A	77
3	Sandeep	A	95