

KENDRIYA VIDYALAYA SANGATHAN PATNA REGION



Study Materials
Session: 2024-25
(Based on Latest CBSE Syllabus)
CLASS: XII COMPUTER SCIENCE (083)

OUR CHIEF PATRONS



Mr. Manish Kumar Prabhat
(A.C. KVS RO PATNA)



Mr. Anurag Bhatnagar
(D. C. KVS RO PATNA)



Mr. Purnendu Mandal
(A. C. KVS RO PATNA)

REVIEW TEAM

Sh. Rishi Raman Principal, PM SHRI K.V. NO1. AFS DARBHANGA, (Chief coordinator)

MEMBERS:

Ms. Anshu Sinha (PM SHRI K.V. Garhara)

Md. Amir (PM SHRI K.V. Jawahar Nagar)

Sh. Pawan Kumar (K.V. Gopalganj)

Miss. Rimjhim (PM SHRI K.V. Bettiah)

Ms. Rachana Gupta (PM SHRI K.V. NO1. AFS DARBHANGA)

CONTENTS

<u>S.NO.</u>	<u>TOPIC</u>	<u>PAGE NO</u>
<u>1.</u>	<u>PYTHON REVISION TOUR AND FUNCTIONS</u>	<u>3</u>
<u>2.</u>	<u>Introduction to files, Text file, Binary file, CSV file, Data Structure</u>	<u>13</u>
<u>3.</u>	<u>Computer Networks</u>	<u>34</u>
<u>4.</u>	<u>DATABASE MANAGEMENT SYSTEM</u>	<u>40</u>

STUDY MATERIAL

PYTHON REVISION TOUR AND FUNCTIONS

Topic :-Keywords, Operators

Keywords:- .Keywords are the words that have special meaning reserved by programming language. – They are reserved for special purpose and cannot be used for normal identifier names. – E.g. in, if, break, class, and, continue, True, False

• **Operators:-**Operators are the symbol, which triggers some computation when applied on operand. – **Unary Operator:** those operators that require only one operand.

- Unary Plus +
- Unary Minus –
- Bitwise Complement ~
- Logical Negation not – **Binary Operator:** those operators that require only two operand.
- Arithmetic Operator +, -, *, /, %
- Bitwise Operator &, ^, |
- Shift Operator >>, <<
- Identity Operator is, is not

Q.1. What is Python variable? Identify the variables that are invalid and state the reason Class, do, while, 4d, a+

Ans: - A variable in python is a container to store data values. a) do, while are invalid because they are python keyword b) 4d is invalid because the name can't be started with a digit. c) a+ is also not valid as no special symbol can be used in name except underscore (_).

Q2. What is None literal in Python?

Ans: Python has one special literal called —None. It is used to indicate something that has not yet been created. It is a legal empty value in Python

CONTROL STATEMENTS

Conditional statements, iterative computation and control flow. Practice of CBSE sample papers based questions on this topic

Control statements are used to control the flow of execution depending upon the specified condition/logic. There are three types of control statements

1. Decision Making Statements (if, elif, else)
2. Iteration Statements (while and for Loops)
3. Jump Statements (break, continue, pass)

Que1. Find the output of the following:-

```
a=int(input("Enter any integer number :"))
```

```
if(a==0):  
    print("Number is Zero")  
elif(a>0):
```

```
print("Number is Positive")
```

```
else:
```

```
    print("Number is negative")
```

ANS:- Enter any integer number :5

Number is Positive

QUE-2 Find the output of following code:-

```
n=1  
while(n<4):  
    print("Govind ", end=" ")  
    n=n+1
```

ANS:-OUTPUT Govind Govind Govind

QUE -3 for i in range(1,6):

```
    print(i, end=' ')
```

ANS:- Output 1 2 3 4 5

QUE-4 for i in range(1,11):

```
    if(i==3):  
        print("hello", end=' ' )  
        continue  
        if(i==8):  
            break  
        if(i==5):  
            pass  
    else:  
        print(i, end=' ');
```

ANS:- 1 2 hello 4 6 7

LIST, TUPLE, DICTIONARY

Creation of a list, tuple & dictionary, Traversal of a list, tuple & dictionary Operations on a list ,tuple and & Dictionary .

Lists in Python

List is a standard data type of Python that can store a sequence of values belonging to any type.

• List is mutable (modifiable) sequence i.e. element can be changed in place.

• Example – List=[1,2,3,4,5] – List1=['p','r','o','b','l','e','m'] –
List2=['pan','ran','oggi','blade','lemon','egg','mango']

Tuples in Python:-It is a sequence of immutable objects. It is just like a list.

Difference between a tuple and a list is that the tuple cannot be changed like a list.

List uses square bracket whereas tuple use parentheses.

L=[1,2,3,4,5] Mutable Elements of list can be changed T=(1,2,3,4,5)

Immutable Elements of tuple can not be changed

Dictionary in Python:- is an unordered collection of data values, used to store data values along with the keys. Dictionary holds key: value pair. Key value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon: , whereas each key is separated by a comma, .

dict={ —a": —alpha", —o": —omega", —g": llgamma } }

Q1. Find the error in following code. State the reason of the error.

```
aLst = { _a':1 , 'b':2, _c':3 }  
print (aLst[_a','b'])
```

Ans: The above code will produce KeyError, the reason being that there is no key same as the list [_a','b'] in dictionary aLst.

Q2. Find and write the output of the following

```
list=['p','r','o','b','l','e','m']  
list[1:3]=[]  
print(list)  
list[2:5]=[]  
print(list)
```

ANS:- [_p','b','l','e','m'] [_p','b']

STRINGS

Traversal, operations – concatenation, repetition, membership; functions / methods. Practice of CBSE sample papers based questions on this topic

String:-String are character enclosed in quotes of any type like single quotation marks, double quotation marks and triple quotation marks. – Computer – Computer – Computer • String are immutable • Empty string has 0 characters. • String is sequence of characters, each character having unique position or index

FUNCTIONS

Functions: scope, functions using libraries: mathematical and string functions

Definition: Functions are the subprograms that perform specific task. Functions are the small modules. Types of Functions: There are three types of functions in python:

- Built in functions
- Functions defined in modules
- User defined functions

Scope of a variable:- is the portion of a program where the variable is recognized. Parameters and variables defined inside a function is not visible from outside. Hence, they have a local scope. There are two types of scope for variables: i) Local Scope ii) Global Scope

Local Scope: Variable used inside the function. It cannot be accessed outside the function. In this scope, the lifetime of variables inside a function is as long as the function executes. They are destroyed once we return from the function. Hence, a function does not remember the value of a variable from its previous calls.

Global Scope: Variable can be accessed outside the function. In this scope, Lifetime of a variable is the period throughout which the variable exists in the memory.

Example: def my_func():

```
    x = 10
```

Q.1. What is default parameter?

Ans: A parameter having default value in the function header is known as a default parameter.

Q2. Can a function return multiple values in python?

Ans: YES.

Q3. Rewrite the correct code after removing the errors:

```
def SI(p,t=2,r):
```

```
    return (p*r*t)/100
```

Ans: - def SI(p, r, t=2):

```
    return(p*r*t)/100
```

Q4. How many values a python function can return? Explain how?

Ans: Python function can return more than one values.

```
def square_and_cube(X):
```

```
    return X*X, X*X*X, X*X*X*X
```

```
    a=3
```

```
    x,y,z=square_and_cube(a)
```

```
    print(x,y,z)
```

User defined functions, parameter passing, passing strings, lists, tuples, dictionaries to functions

User defined functions:- The functions those are defined by the user are called user defined functions. The syntax to define a function is:

```
def function-name ( parameters) :
```

Keyword def marks the start of function header. A function name to uniquely identify it. Function naming follows the same rules of writing identifiers in Python. Parameters (arguments) through which we pass values to a function. They are optional. A colon (:) to mark the end of function header. One or more valid python statements that make up the function body. Statements must have same indentation level. An optional return statement to return a value from the function. #statement(s) Example:

```
def display(name):
```

```
    print("Hello " + name + " How are you?")
```

Q1. Find the output of the following

```
L1 = [100,900,300,400,500]
```

```
START = 1 SUM = 0
```

```
for C in range(START,4):
```

```
    SUM = SUM + L1[C]
```

```
    print(C, ":", SUM)
```

```
    SUM = SUM + L1[0]*10
```

```
print(SUM)
```

ANS:- O/P 1:900 1900 3200 3:3600 4600

Q-2.What is the difference between actual and formal parameters ?

ANS:- The list of identifiers used in a function call is called actual parameter(s) whereas the list of parameters used in the function definition is called formal parameter(s).

Actual parameter may be value / variable or expression. Formal parameter is an identifier. Actual parameters are those parameters which are used in function call statement and formal parameters are those parameters which are used in function header (definition).

e.g. def sum(a,b): # a and b are formal parameters

```
    return a+b
```

```
    x,y=5,10
```

res=sum(x,y) # x and y are actual parameters

ERROR FINDING AND OUTPUT BASED QUESTIONS

Q5. Rewrite the following code in Python after removing all syntax error(s). Underline each correction done in the code.

```
p=30
```

```
for c in range(0,p)
```

```
    If c%4==0:
```

```
        print (c*4)
```

```
Elseif c%5==0:
```

```
    print (c+3)
```

```
else print(c+10)
```

Ans:

```
p=30
```

```
for c in range(0,p):   Error 1
```

```
if c%4==0:   Error 2
```

```
    print (c*4)
```

```
elif c%5==0: Error 3
```

```
    print (c+3)
```

```
else: Error 4
    print(c+10)
```

Q6. Rewrite the following code in python after removing all syntax errors. Underline each correction done in the code:

```
Def func(a):
for i in (0,a):

    if i%2 =0:
        s=s+1
else if i%5= =0
    m=m+2
else: n=n+i
    print(s,m,n)
    func(15)
```

Ans:

```
def func(a): Error 1
for i in range(0,a): Error 2
    if i%2 ==0: Error 3
        s=s+1
elif i%5= =0 Error 4
    m=m+2
else: n=n+i
    print(s,m,n)
func(15)
```

Q7. What possible outputs(s) are expected to be displayed on screen at the time of execution of the program from the following code. Select which option/s is/are correct import random

```
print(random.randint(15,25) , end=' ')
print((100) + random.randint(15,25) , end = '')
print((100) -random.randint(15,25) , end = '')
print((100) *random.randint(15,25) )
```

(i) 15 122 84 2500 (ii) 21 120 76 1500 (iii) 105 107 105 1800 (iv) 110 105 105 1900

Ans: (i) and (ii) are correct answers Hint:- random.randrange(15,25) function returns all numbers between 15 to 25 (including both)

Q8. What possible outputs(s) are expected to be displayed on screen at the time of execution of the program from the following code? Also specify the minimum and maximum values that can be assigned to the variable End.

```
import random
```



```
Colours = ["VIOLET","INDIGO","BLUE","GREEN", "YELLOW","ORANGE","RED"]
```

```
End = randrange(2)+3
```

```
Begin = randrange(End)+1
```

```
for i in range(Begin,End):
```

```
    print(Colours[i],end="&")
```

(i) INDIGO&BLUE&GREEN& (ii) VIOLET&INDIGO&BLUE& (iii) BLUE&GREEN&YELLOW& (iv) GREEN&YELLOW&ORANGE&

Ans: (i) INDIGO&BLUE&GREEN& Minimum Value of End = 3 Maximum Value of End = 4

Q9. Write a statement in Python to declare a dictionary whose keys are 1,2,3 and values are Monday, Tuesday and Wednesday respectively.

Ans: Dict1 = { 1:"Monday", 2:"Tuesday", 3: "Wednesday" }

QUESTIONS : FUNCTIONS - OUTPUT AND ERROR

QA. Identify the errors, underline it and correct the errors

a) Def Sum(a=1,b)

```
    return a+b
```

```
    print ("The sum =" Sum(7, -1)
```

b) def main ()

```
    print ("hello")
```

c) def func2() :

```
    print (2 + 3)
```

```
    func2(5)
```

Q1. Find the output of the following numbers:

```
Num = 20
```

```
Sum = 0
```

```
for i in range (10, Num, 3):
```

```
    Sum+=i
```

```
    if i%2==0:
```

```
        print (i*2)
```

```
    else:
```

```
        print (i*3)
```

Q2. Find the output of the following

```
Text="gmail@com"
```

```
L=len(Text)
```

```
ntext=""
```

```
for i in range (0,L):
```

```
    if text[i].isupper():
```

```
        ntext=ntext+text[i].lower()
```

```
    elif text[i].isalpha():
```

```
        ntext=ntext+text[i].upper()
```

```
    else:
```

```
        ntext=ntext+'bb'
```

Q3. Find the output of the following-

```
def power (b , p):
```

```
    r = b ** P
```

```
    return r
```

```
def calcSquare(a):
    a = power (a, 2)
return a
n = 5
result = calcSquare(n)
print (result)
```

Q4. Find the output of the following-

```
import math
print (math. floor(5.5))
```

Q5. Find the output

```
def gfg(x,l=[ ]):
    for I in range(x):
        l.append(i*i)
        print(l)

gfg(2)
gfg(3,[3,2,1])
gfg(3)
```

SOLUTION: FUNCTIONS - OUTPUT AND ERROR

Ans Aa: def sum(a=1,b) :__
 return a+b (indentation)
 print ("The sum =", Sum(7,-1))

Ans Ab: def main ():
 print ("hello")

Ans Ac: def func2() :
 print (2 + 3)

func2() no parameter is to be passed

1. output: 20 2. Output: GMAILbbCOM

39
 32
 57

3. output: 25

4. output: 6

5. output: [0,1]
 [3,2,1,0,1,4]
 [0,1,0,1,4]

Python Exception Handling

Error in Python can be of two types i.e. Syntax errors and Exceptions. Errors are problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which change the normal flow of the program.

Difference between Syntax Error and Exceptions

Syntax Error: As the name suggests this error is caused by the wrong syntax in the code. It leads to the termination of the program.

Example:

There is a syntax error in the code . The 'if' statement should be followed by a colon (:), and the 'print' statement should be indented to be inside the 'if' block.

Exceptions: Exceptions are raised when the program is syntactically correct, but the code results in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Example:

Here in this code a s we are dividing the 'marks' by zero so a error will occur known as 'ZeroDivisionError'

Try and Except Statement – Catching Exceptions

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

Example: Here we are trying to access the array element whose index is out of bound and handle the corresponding exception.

In the above example, the statements that can cause the error are placed inside the try statement (second print statement in our case). The second print statement tries to access the fourth element of the list which is not there and this throws an exception. This exception is then caught by the except statement.

Catching Specific Exception

A try statement can have more than one except clause, to specify handlers for different exceptions. Please note that at most one handler will be executed. For example, we can add IndexError in the above code. The general syntax for adding specific exceptions are –

```
try:
    # statement(s)
except IndexError:
    # statement(s)
except ValueError:
    # statement(s)Example
```

```
def fun(a):
    if a < 4:

        b = a/(a-3)
    print("Value of b = ", b)
```

```
try:
    fun(3)
    fun(5)
except ZeroDivisionError:
    print("ZeroDivisionError Occurred and Handled")
except NameError:
    print("NameError Occurred and Handled")
```

LONG ANSWER QUESTION

Q.1. Write a function in **Display** which accepts a list of integers and its size as arguments and replaces elements having even values with its half and elements having odd values with twice its value .

eg: if the list contains

5, 6, 7, 16, 9

then the function should rearranged list as

10, 3,14,8, 18

```
Ans:- def Display (X, n):
    for i in range(n):
        if X[i] % 2 == 0:
            X[i] /= 2
        else:
            X[i] *= 2
    print (X)
```

Write the output of following python code

```
T="Happy New Year 2021"
L=len(T)
```

```

ntext=""
for i in range (0,L):
    if T[i].isupper():
        ntext=ntext+T[i].lower()
    elif T[i].isalpha():
        ntext=ntext+T[i].upper()
else:
    ntext=ntext+"*"
print (ntext)

```

Ans:- **hAPPY*nEW*yEAR*******

Q3. What do you understand by local and global scope of variables? How can you access a global variable inside the function, if function has a variable with same name.

Ans: Variables that are defined inside a function body have a local scope, and those defined outside have a global scope. This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions.

Ex: **Local Scope:** A variable created inside a function is available inside that function:

```

def myfunc():
    x = 300
    print(x)
myfunc()

```

Global Scope: A variable created in the main body of the Python code is a global variable and belongs to the global scope. Global variables are available from within any scope, global and local.

```

x = 300
def myfunc():
    print(x)
myfunc()
print(x)

```

If your function has a local variable with same name as global variable and you want to modify the global variable inside function then use '**global**' keyword before the variable name.

Q4. Write a user defined function **GenNum(a, b)** to generate odd numbers between a and b (including b).

Ans . def getNum(a,b):
for i in range(a,b+1):
if i%2==1:
print(i)

Q.5. Write definition of a method/function **AddOdd(VALUES)** to display sum of odd values from the list of VALUES.

Ans def AddOdd(Values):
for i in range(a,b+1):
n=len(NUMBERS)
if i%2==1: s=0
print(i) for i in range(n):
if (i%2!=0):
s=s+NUMBERS[i]
print(s)

Q. 6. Write definition of a Method **MSEARCH(STATES)** to display all the state names from a list of STATES, which are starting with alphabet M.

For example:

If the list STATES contains ["MP", "UP", "MH", "DL", "MZ", "WB"] The following should get displayed MP

MH

MZ

```
Ans def MSEARCH(STATES):
    for i in STATES:
        if i[0]=='M':
            print(i)
```

Q.7. Write a python function generatefibonacci(n) where n is the limit, using a generator function Fibonacci (max)(where max is the limit n) that produces Fibonacci series.

```
def Fibonacci (max):
    a, b = 0, 1
    while a <=max:
        yield a,
        a, b = b, a+b
def generatefibonacci(n):
    for i in Fibonacci (n):
        print( i)
```

Q.8. Write a definition of a method COUNTNOW(PLACES) to find and display those place names, in which here are more than 7 characters.

For example:

If the list PLACES contains. ["MELBORN","TOKYO","PINKCITY","BEIZING","SUNCITY"]

The following should get displayed : PINKCITY

```
Ans. l=["MELBORN","TOKYO","PINKCITY","BEIZING","SUNCITY"]
```

```
def countno(m):
    length=len(m)
    for i in range(0,length):
        if len(m[i])>7:
            print(m[i])
            countno(l)
```

Unit 1: Computational Thinking and Programming – 2

Introduction to Files

Files are essential for storing data permanently. In Python, files can be manipulated using various operations such as reading, writing, and updating.

Types of Files

1. **Text Files:** Store data in a human-readable format (e.g., .txt, .csv).
2. **Binary Files:** Store data in a binary format, suitable for non-text data (e.g., images, audio).
3. **CSV Files:** Comma-separated values, commonly used for tabular data. They are text files but follow a specific structure.

Relative and Absolute Paths

- **Absolute Path:** Full path from the root directory to the file (e.g., C:\Users\User\Documents\file.txt).
- **Relative Path:** Path relative to the current working directory (e.g., file.txt).

Text File Handling

Opening a Text File

To open a text file, use the open() function:

```
file = open('example.txt', 'r') # Opens the file in read mode
```

Text File Open Modes

- 'r': Read (default mode).
- 'r+': Read and write.
- 'w': Write (creates a new file or truncates an existing file).
- 'w+': Write and read.
- 'a': Append (adds data at the end).
- 'a+': Append and read.

Closing a Text File

Always close files after operations:

```
file.close()
```

Using the with Clause

Automatically manages file closing:

```
with open('example.txt', 'r') as file:
```

```
    content = file.read()
```

Writing/Appending Data to a Text File

```
# Writing to a file
```

```
with open('example.txt', 'w') as file:
```

```
    file.write('Hello, World!\n')
```

```
# Appending to a file
```

```
with open('example.txt', 'a') as file:
```

```
    file.write('Appending a new line.\n')
```

Reading from a Text File

```
# Reading from a file
```

```
with open('example.txt', 'r') as file:
```

```
    content = file.read() # Read entire file
```

```
    line = file.readline() # Read one line
```

```
    lines = file.readlines() # Read all lines into a list
```

Seek and Tell Methods

seek() Method

The seek() method changes the current file position to a specified byte offset. It allows you to move the file pointer to different locations within the file.

Syntax:

```
file.seek(offset, whence)
```

- **offset**: This is the number of bytes to move the pointer from the position specified by whence.
- **whence** (optional): This parameter specifies the reference point for the offset:
 - 0: The beginning of the file (default).
 - 1: The current file position.
 - 2: The end of the file.

Example of seek()

```
# Example demonstrating the use of seek()
with open('example.txt', 'w') as file:
    file.write('Hello, World!\n')
    file.write('This is a test file.\n')
```

Reading and using seek

```
with open('example.txt', 'r') as file:
    print(file.read(5)) # Output: Hello
    file.seek(0)       # Move to the beginning of the file
    print(file.read(5)) # Output: Hello
    file.seek(7)       # Move to the 8th byte (0-indexed)
    print(file.read(5)) # Output: World
```

In this example:

- After writing to the file, we read the first 5 characters.
- Then we use seek(0) to move the pointer back to the start of the file and read again.
- By using seek(7), we jump directly to the 8th byte and read the next 5 bytes.

tell() Method

The tell() method returns the current position of the file pointer in bytes. This is useful for tracking where you are in the file during reading or writing operations.

Syntax:

```
file.tell()
```

Example of tell()

```
# Example demonstrating the use of tell()
with open('example.txt', 'r') as file:
    print(file.tell()) # Output: 0 (at the start of the file)
    file.read(5)      # Read the first 5 characters
    print(file.tell()) # Output: 5 (after reading 5 characters)
    file.read(10)     # Read the next 10 characters
    print(file.tell()) # Output: 15 (position after reading 15 characters)
```

In this example:

- tell() initially returns 0, indicating the start of the file.
- After reading 5 characters, it returns 5, showing the new position.
- After reading an additional 10 characters, tell() returns 15, indicating the total number of bytes read so far.

Manipulation of Data in a Text File

To modify data, read the content, make changes, and write it back:

```
with open('example.txt', 'r') as file:
    content = file.readlines()
```

Modify content

```
content[0] = 'Modified Line\n'
```

with open('example.txt', 'w') as file:

```
file.writelines(content)
```

Binary File Handling

Basic Operations on Binary Files

```
# Open binary file in write mode
with open('example.bin', 'wb') as file:
    file.write(b'Hello, Binary World!') # Writing bytes
```

Open Modes for Binary Files

- 'rb': Read binary.
- 'rb+': Read and write binary.
- 'wb': Write binary.
- 'wb+': Write and read binary.
- 'ab': Append binary.
- 'ab+': Append and read binary.

Using the pickle Module

The pickle module allows you to serialize and deserialize Python objects.

Example:

```
import pickle

# Writing to a binary file
data = {'name': 'Alice', 'age': 25}
with open('data.pkl', 'wb') as file:
    pickle.dump(data, file)

# Reading from a binary file
with open('data.pkl', 'rb') as file:
    loaded_data = pickle.load(file)
    print(loaded_data) # Output: {'name': 'Alice', 'age': 25}
```

CSV File Handling

Using the csv Module

Python provides the csv module for handling CSV files.

Writing to a CSV File

```
import csv

# Writing to a CSV file
data = [['Name', 'Age'], ['Alice', 25], ['Bob', 30]]

with open('data.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(data) # Writing multiple rows
```

Reading from a CSV File

```
import csv

# Reading from a CSV file
with open('data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row) # Each row is a list
```


Multiple-Choice Questions (MCQs)

1. Which of the following modes is used to open a file for reading only?

- A) w
- B) a
- C) r
- D) wb

○ Answer: C (r)

2. What does the w+ mode do when opening a file?

- A) Opens a file for reading only.
- B) Opens a file for writing and reading, truncating the file.
- C) Opens a file for appending data.
- D) Opens a file for writing and reading without truncating.

○ Answer: B (Opens a file for writing and reading, truncating the file.)

3. Which method is used to read a single line from a text file?

- A) read()
- B) readlines()
- C) readline()
- D) getline()

○ Answer: C (readline())

4. What will be the output of the following code?

```
f = open('data.txt', 'w')
f.write('Hello\nWorld')
f.close()
f = open('data.txt', 'r')
print(f.read())
```

- A) Hello World
- B) Hello\nWorld
- C) Error
- D) Hello

○ Answer: B (Hello\nWorld)

5. Which of the following is the correct way to close a file in Python?

- A) end()
- B) close()
- C) exit()
- D) stop()

○ Answer: B (close())

6. What does the with statement ensure when opening a file?

- A) The file is deleted after use.
- B) The file is automatically closed after the block.
- C) The file can only be read.
- D) The file can be opened multiple times.

○ Answer: B (The file is automatically closed after the block.)

7. Which method is used to write multiple lines to a text file?

- A) write()
- B) writelines()
- C) writeline()
- D) append()

○ Answer: B (writelines())

8. Which of the following modes opens a binary file for reading?

- A) rb

B) wb

C) ab

D) r

- **Answer:** A (rb)

9. **What will the following code output?**

```
f = open('numbers.bin', 'wb')
```

```
f.write(b'\x01\x02\x03')
```

```
f.close()
```

A) Error

B) 1, 2, 3

C) Binary data saved

D) 1 2 3

- **Answer:** C (Binary data saved)

10. **Which method is used to load data from a binary file using the pickle module?**

A) pickle.load()

B) pickle.read()

C) pickle.load_data()

D) pickle.get()

- **Answer:** A (pickle.load())

11. **Which of the following is used to write data into a CSV file?**

A) csv.write()

B) csv.writer()

C) csv.append()

D) csv.add()

- **Answer:** B (csv.writer())

12. **What does the reader() function do in the CSV module?**

A) Reads a single value from the CSV file.

B) Reads the entire CSV file.

C) Reads each row from the CSV file as a list.

D) Writes data to the CSV file.

- **Answer:** C (Reads each row from the CSV file as a list.)

13. **Which of the following file modes allows appending data to an existing file?**

A) w

B) r

C) a

D) wb

- **Answer:** C (a)

14. **Which of the following paths specifies the location of a file from the root directory?**

A) Relative path

B) Absolute path

C) Shortcut path

D) Virtual path

- **Answer:** B (Absolute path)

15. **What will be the output of the following code?**

```
with open('sample.txt', 'w') as f:
```

```
    f.write('Python\n')
```

```
    f.write('Programming')
```

```
with open('sample.txt', 'r') as f:
```

```
    print(f.read())
```

A) Python

B) Python\nProgramming

C) Programming

D) Python Programming

- **Answer:** B (Python\nProgramming)

16. **Which method is used to append data to a binary file?**

- A) ab
- B) wb
- C) rb
- D) rb+

- **Answer:** A (ab)

17. **Which of the following modes allows reading and writing to a file without truncating it?**

- A) rb+
- B) wb+
- C) r+
- D) a+

- **Answer:** C (r+)

18. **What is the purpose of the seek() method?**

- A) Reads data from a specific location
- B) Moves the file pointer to a specified position
- C) Writes data to a specified position
- D) Closes the file

- **Answer:** B (Moves the file pointer to a specified position)

19. **Which of the following methods reads all lines from a file and returns a list?**

- A) read()
- B) readlines()
- C) readline()
- D) gets()

- **Answer:** B (readlines())

20. **Which command is used to import the CSV module in Python?**

- A) import csv
- B) from csv import *
- C) import CSV
- D) include csv

- **Answer:** A (import csv)

Very Short Answer Questions

1. **Define a text file and give one example of its use.**

A text file stores data in plain text format and is often used for configuration files.

2. **What does the 'a+' mode do when opening a text file?**

The 'a+' mode opens a file for reading and appending, creating the file if it does not exist.

3. **How do you ensure that a file is closed automatically after its use?**

By using the with statement, which ensures the file is closed after the block of code is executed.

4. **Write the command to read the first line of a text file named example.txt.**

line = open('example.txt', 'r').readline()

5. **What method would you use to read the entire content of a text file at once?**

The read() method.

6. **How do you write a list of strings to a text file?**

Using the writelines() method, e.g., f.writelines(list_of_strings).

7. **What is the purpose of the tell() method?**

It returns the current position of the file pointer in the file.

8. **What is a binary file? Provide one application.**

A binary file stores data in a format not meant for human reading, used for storing images or audio files.

9. **What file mode allows both reading and writing to a binary file without truncating it?**

The rb+ mode.

10. **Explain the role of the pickle module in Python.**
The pickle module serializes Python objects into binary format for storage and deserialization.
11. **Write a code snippet to append an integer to a binary file.**

```
import pickle
with open('data.bin', 'ab') as f:
    pickle.dump(42, f)
```
12. **What does the load() method do in the pickle module?**
It reads a Python object from a binary file.
13. **Define a CSV file. What is its primary use?**
A CSV file is a comma-separated values file used for storing tabular data, allowing easy import/export in spreadsheets.
14. **How do you write multiple rows to a CSV file in Python?**
Using the writerows() method from the csv module.
15. **What does the reader() function from the CSV module return?**
It returns a reader object that iterates over lines in the specified CSV file.
16. **How can you open a CSV file for reading in Python?**

```
import csv
with open('file.csv', 'r') as f:
    reader = csv.reader(f)
```
17. **Differentiate between absolute and relative paths with examples.**
An absolute path specifies the full path from the root (e.g., C:/Users/Name/file.txt), while a relative path specifies the path relative to the current working directory (e.g., file.txt).
18. **What is the result of calling f.seek(0) in a file handling context?**
It moves the file pointer to the beginning of the file.
19. **Which method would you use to write a single row to a CSV file?**
The writerow() method.
20. **How can you safely read from a file while handling potential errors?**
By using a try-except block within the with statement to catch exceptions during file operations.

Short Questions

1. **What is a text file, and how does it differ from a binary file?**
A text file contains human-readable data, while a binary file contains data in a format not intended for human readability, such as images or executable files.
2. **How do you open a file for writing without truncating its content?**
Use the mode a+ to open the file for reading and appending, preserving existing content.
3. **Explain the function of the open() method in file handling.**
The open() method is used to open a file and returns a file object, allowing you to perform read/write operations.
4. **Write a code snippet to read the first three lines of a text file named data.txt.**

```
with open('data.txt', 'r') as f:
    for _ in range(3):
        print(f.readline())
```
5. **What is the purpose of the writelines() method in text file operations?**
The writelines() method is used to write a list of strings to a text file.
6. **What will happen if you open a file in w mode that already exists?**
The existing file will be truncated to zero length, effectively deleting its previous content.
7. **How can you move the file pointer to the end of the file?**
By using the seek(0, 2) method, which sets the pointer to the end of the file.
8. **Explain the significance of the pickle module in Python.**
The pickle module allows you to serialize and deserialize Python objects, enabling storage of complex data types in binary files.

9. **What file mode would you use to read and write binary files without truncating them?**
Use rb+ to read and write without truncating the binary file.
10. **Write a line of code to append a new entry to a binary file using the pickle module.**
import pickle
with open('data.bin', 'ab') as f:
 pickle.dump(new_entry, f)
11. **What does the reader() function from the csv module do?**
It reads data from a CSV file and returns an iterable object that contains each row as a list.
12. **How do you write multiple rows to a CSV file?**
By using the writerows() method from the csv module.
13. **Describe the difference between absolute and relative paths with examples.**
An absolute path specifies the complete path from the root directory (e.g., C:/Users/Name/Documents/file.txt), while a relative path is based on the current working directory (e.g., file.txt).
14. **How can you safely read from a text file and handle potential errors?**
Use a try-except block within a with statement to catch exceptions during file operations.
15. **What will happen if you attempt to read from a file opened in w mode?**
It will raise an error because the w mode does not allow reading from the file.
16. **How do you import the csv module in Python?**
By using the statement import csv.
17. **What is the effect of using the seek(0) method in a file?**
It moves the file pointer back to the beginning of the file.
18. **Write a code snippet to create a CSV file and write the following data into it: Name, Age, City.**
import csv
with open('data.csv', 'w', newline='') as f:
 writer = csv.writer(f)
 writer.writerow(['Name', 'Age', 'City'])
19. **What happens if you try to open a file in rb mode that does not exist?**
It will raise a FileNotFoundError.
20. **Explain the purpose of the close() method in file handling.**
The close() method is used to close an open file, ensuring that any changes are saved and releasing system resources.

Programming Questions

1. **Write a Python program to create a text file named example.txt and write "Hello, World!" into it.**
with open('example.txt', 'w') as f:
 f.write("Hello, World!")
2. **How can you read the entire content of example.txt and print it?**
with open('example.txt', 'r') as f:
 content = f.read()
 print(content)
3. **Write a program that appends the line "Welcome to Python!" to example.txt.**
with open('example.txt', 'a') as f:
 f.write("Welcome to Python!\n")
4. **Write a function that reads and prints the first three lines of a text file.**
def read_first_three_lines(filename):
 with open(filename, 'r') as f:
 for _ in range(3):
 print(f.readline())
 read_first_three_lines('example.txt')
5. **Explain and demonstrate the use of the tell() method in a text file.**
with open('example.txt', 'r') as f:

```
print(f.tell()) # Prints the initial position (0)
f.read(5)      # Reads first 5 characters
print(f.tell()) # Prints the new position (5)
```

6. **Write a program to read all lines from a text file and store them in a list.**

```
with open('example.txt', 'r') as f:
    lines = f.readlines()
    print(lines)
```

7. **Write a code snippet to create a binary file and store a list of numbers using the pickle module.**

```
import pickle

numbers = [1, 2, 3, 4, 5]
with open('numbers.bin', 'wb') as f:
    pickle.dump(numbers, f)
```

8. **Demonstrate how to read a list of numbers from a binary file using pickle.**

```
import pickle

with open('numbers.bin', 'rb') as f:
    numbers = pickle.load(f)
    print(numbers) # Outputs: [1, 2, 3, 4, 5]
```

9. **Write a Python program that opens a binary file and appends a new number using the pickle module.**

```
import pickle

new_number = 6
with open('numbers.bin', 'ab') as f:
    pickle.dump(new_number, f)
```

10. **Explain the difference between wb and wb+ modes when opening a binary file.**

- wb: Opens the file for writing only, truncating the file if it exists.
- wb+: Opens the file for both reading and writing, truncating the file if it exists.

11. **Write a program to create a CSV file and write the following data into it: Name, Age, City.**

```
import csv

with open('data.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['Name', 'Age', 'City'])
    writer.writerow(['Alice', 22, 'New York'])
    writer.writerow(['Bob', 25, 'Los Angeles'])
```

12. **How do you read data from a CSV file and print each row?**

```
import csv

with open('data.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

13. **Write a code snippet to append a new row to the existing data.csv file.**

```
import csv

with open('data.csv', 'a', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['Charlie', 30, 'Chicago'])
```

14. **Explain and demonstrate the use of the seek() method in file handling.**

```
with open('example.txt', 'r') as f:
    print(f.read(5)) # Reads first 5 characters
    f.seek(0)      # Moves pointer back to the start
    print(f.read(5)) # Reads first 5 characters again
```

15. **Write a Python program to count the number of lines in a text file.**

```
def count_lines(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
        return len(lines)
print(count_lines('example.txt'))
```

16. **How can you open a CSV file for reading and print only the second column of each row?**

```
import csv

with open('data.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row[1]) # Prints the second column (Age)
```

17. **Write a program to update an entry in a CSV file. Assume the first column is a unique identifier.**

```
import csv

def update_age(filename, name, new_age):
    rows = []
    with open(filename, 'r') as f:
        reader = csv.reader(f)
        for row in reader:
            if row[0] == name:
                row[1] = new_age # Update age
            rows.append(row)

    with open(filename, 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerows(rows)
```

```
update_age('data.csv', 'Alice', 23)
```

18. **What is the purpose of the newline="" argument when opening a CSV file?**
It prevents extra blank lines from being added when writing to the CSV file, especially on Windows.

19. **Write a Python program to read a binary file and display its content in a human-readable format.**

```
import pickle

with open('numbers.bin', 'rb') as f:
    while True:
        try:
            number = pickle.load(f)
            print(number)
        except EOFError:
            break
```

20. **Write a program that demonstrates the use of both read() and readlines() on a text file.**

```
with open('example.txt', 'r') as f:
    print("Using read():")
```

```
print(f.read())
f.seek(0) # Reset pointer
print("Using readlines():")
print(f.readlines())
```

Scoring Tips

1. Understand File Handling Basics

- **Familiarize Yourself with File Concepts:** Understand what files are, the difference between text and binary files, and the purpose of CSV files. Knowing these concepts will help you answer fundamental questions accurately.

2. Know File Open Modes

- **Memorize File Modes:** Ensure you know the different modes for opening files (e.g., r, w, a, r+, rb, wb, etc.) and when to use each one. You can create a quick reference chart and revise it regularly.

3. Practice Using the with Statement

- **Use the with Statement:** Practice using the with statement for file operations as it automatically handles file closing. This is often preferred and may fetch you extra marks.

4. Master Reading and Writing Methods

- **Understand Different Reading Methods:** Be comfortable using read(), readline(), and readlines() for reading files, and write() and writelines() for writing data. Practice scenarios where you need to decide which method to use based on the requirements.

5. Binary Files and the pickle Module

- **Understand the pickle Module:** Get comfortable using the pickle module for serializing and deserializing Python objects. Write and test code snippets to save and load various data types.

6. CSV File Operations

- **CSV Module:** Practice using the csv module for reading from and writing to CSV files. Make sure you understand the difference between writer(), writerow(), and writerows(), as well as how to use reader().
- **Real-Life Applications:** Understand how CSV files are used in data handling, such as spreadsheets and databases, to contextualize your learning.

7. Absolute vs. Relative Paths

- **Know the Differences:** Understand absolute and relative paths with examples. Be prepared to identify or write both types of paths in questions.

8. Write Clean, Efficient Code

- **Code Structure:** Write clean and well-commented code. Indentation and readability matter; they can help you earn partial marks even if your program has minor errors.

9. Practice Previous Year Papers

- **Solve Past Papers:** Regularly practice previous year question papers and sample papers to get familiar with the question format. Time yourself to simulate exam conditions.

10. Focus on Error Handling

- **Handle Exceptions:** Understand how to handle exceptions when dealing with files (e.g., file not found, permission errors). Knowing how to write try-except blocks can enhance your answers.

11. Revise Regularly

- **Frequent Revision:** Regularly revisit the concepts, especially file operations, to reinforce your understanding and memory retention.

12. Create a Checklist

- **Before the Exam:** Create a checklist of important file handling concepts and methods to review before the exam. This will ensure you cover everything during your last-minute revision.

A **stack** is a linear data structure that follows the **Last In First Out (LIFO)** principle. This means that the last element added to the stack will be the first one to be removed. Stacks are used in various applications, including function call management, expression evaluation, and backtracking algorithms.

Characteristics of a Stack

- **LIFO Order:** The last element added is the first to be removed.
- **Two main operations:**
 - **Push:** Add an element to the top of the stack.
 - **Pop:** Remove the element from the top of the stack.

Operations on Stack

1. **Push Operation**
 - Adds an element to the top of the stack.
 - If the stack is full (in fixed size implementations), this operation can cause an overflow.
2. **Pop Operation**
 - Removes the top element from the stack.
 - If the stack is empty, this operation can cause an underflow.
3. **Peek Operation (Optional)**
 - Returns the top element of the stack without removing it.
4. **isEmpty Operation (Optional)**
 - Checks if the stack is empty.
5. **Size Operation (Optional)**
 - Returns the number of elements in the stack.

Simple Implementation of Stack Using List

In this implementation, we'll define functions to perform the stack operations: **push**, **pop**, **peek**, **is_empty**, and **size**.

```
# Initialize an empty stack
stack = []

# Function to push an element onto the stack
def push(item):
    stack.append(item)
    print(f"Pushed {item} onto the stack.")

# Function to pop an element from the stack
def pop():
    if is_empty():
        return "Stack Underflow! Cannot pop from an empty stack."
    return stack.pop()

# Function to peek at the top element of the stack
def peek():
    if is_empty():
        return "Stack is empty! Cannot peek."
    return stack[-1]

# Function to check if the stack is empty
def is_empty():
    return len(stack) == 0
```

```

# Function to get the size of the stack
def size():
    return len(stack)

# Function to display the current stack
def display():
    return stack

# Example usage of the stack functions
push(10)      # Output: Pushed 10 onto the stack.
push(20)      # Output: Pushed 20 onto the stack.
push(30)      # Output: Pushed 30 onto the stack.

print("Current Stack:", display()) # Output: Current Stack: [10, 20, 30]

popped_item = pop()          # Pops 30
print("Popped Item:", popped_item) # Output: Popped Item: 30

print("Current Stack after popping:", display()) # Output: Current Stack: [10, 20]

top_item = peek()           # Peeks at the top element
print("Top Element:", top_item) # Output: Top Element: 20

print("Size of stack:", size()) # Output: Size of stack: 2

# Attempt to pop until the stack is empty
pop() # Pops 20
pop() # Pops 10
print(pop()) # Output: Stack Underflow! Cannot pop from an empty stack.

```

Explanation of the Code

1. **Initialization:**
 - We create an empty list called stack to store the stack elements.
2. **Push Function:**
 - push(item): This function appends the item to the end of the list (top of the stack).
3. **Pop Function:**
 - pop(): This function checks if the stack is empty. If not, it removes and returns the last element from the list (top of the stack). If the stack is empty, it returns an underflow message.
4. **Peek Function:**
 - peek(): This function checks if the stack is empty. If not, it returns the last element without removing it. If the stack is empty, it returns a message indicating that.
5. **is_empty Function:**
 - is_empty(): This function checks if the stack is empty by evaluating the length of the list.
6. **Size Function:**
 - size(): This function returns the number of elements currently in the stack.
7. **Display Function:**
 - display(): This function returns the current elements in the stack.

Competency-Based MCQs on Stack Data Structure

1. **What is a stack?**
 - A) A linear data structure that follows LIFO
 - B) A linear data structure that follows FIFO
 - C) A non-linear data structure

D) A data structure that allows random access

Answer: A) A linear data structure that follows LIFO

2. **Which operation adds an element to the top of the stack?**

A) pop

B) push

C) peek

D) top

Answer: B) push

3. **Which operation removes the top element from the stack?**

A) push

B) pop

C) peek

D) isEmpty

Answer: B) pop

4. **What will be the result of popping an element from an empty stack?**

A) Returns None

B) Returns an error

C) Returns 0

D) Nothing happens

Answer: B) Returns an error

5. **In which scenario would you use a stack?**

A) For implementing a queue

B) For tracking function calls in recursion

C) For sorting elements

D) For searching elements

Answer: B) For tracking function calls in recursion

6. **What will be the top element of the stack after performing the following operations?**

○ Push(10)

○ Push(20)

○ Pop()

○ Push(30)

A) 10

B) 20

C) 30

D) None of the above

Answer: C) 30

7. **How can you implement a stack using a list in Python?**

A) Use append() for push and remove() for pop

B) Use insert() for push and pop() for pop

C) Use append() for push and pop() for pop

D) Use add() for push and delete() for pop

Answer: C) Use append() for push and pop() for pop

8. **What is the time complexity of push and pop operations in a stack implemented using a list?**

A) O(1)

B) O(n)

C) O(log n)

D) O(n²)

Answer: A) O(1)

9. **What is the purpose of the peek operation in a stack?**

A) To remove the top element

B) To add an element to the stack

C) To view the top element without removing it

D) To clear the stack

Answer: C) To view the top element without removing it

10. **Given the stack implementation in Python:**

```
stack = []
stack.append(5)
stack.append(10)
stack.append(15)
print(stack.pop())
```

What will be the output of the program?

A) 5

B) 10

C) 15

D) None

Answer: C) 15

11. **Which of the following is a characteristic of a stack?**

A) Elements can be accessed in any order

B) Only the top element can be accessed

C) Elements are stored in sorted order

D) Stack allows duplicate elements only

Answer: B) Only the top element can be accessed

12. **Which of the following statements about stack implementation is true?**

A) A stack can only be implemented using an array.

B) A stack can be implemented using a linked list.

C) Both A and B are true.

D) None of the above.

Answer: B) A stack can be implemented using a linked list.

13. **In Python, which built-in function would you use to determine if a list (acting as a stack) is empty?**

A) isEmpty()

B) len()

C) count()

D) size()

Answer: B) len()

14. **If the following operations are performed on a stack:**

○ Push(1)

○ Push(2)

○ Push(3)

○ Pop()

○ Push(4)

What is the final state of the stack?

A) [1, 2]

B) [1, 4]

C) [2, 3, 4]

D) [3, 4]

Answer: B) [1, 4]

15. **Which of the following implementations of stack is more memory efficient?**

A) Using an array

B) Using a linked list

C) Both have the same efficiency

D) None of the above

Answer: B) Using a linked list

16. **If a stack has a maximum size of 5, what happens when you try to push the sixth element?**

A) It successfully adds the element.

- B) It raises an overflow error.
- C) It replaces the bottom element.
- D) It does nothing.

Answer: B) It raises an overflow error.

17. **Which of the following is NOT a common application of stacks?**

- A) Expression evaluation
- B) Backtracking algorithms
- C) Queue implementation
- D) Undo mechanisms in applications

Answer: C) Queue implementation

18. **What is the primary difference between a stack and a queue?**

- A) Stack follows LIFO, while queue follows FIFO
- B) Stack follows FIFO, while queue follows LIFO
- C) Both follow the same principle
- D) There is no difference

Answer: A) Stack follows LIFO, while queue follows FIFO

19. **Which of the following is a valid Python syntax for popping an element from a stack implemented as a list?**

- A) stack.pop()
- B) stack.remove()
- C) stack.delete()
- D) stack.pop(1)

Answer: A) stack.pop()

20. **If you have the following stack operations, what will the stack contain after these operations?**

- o Push(2)
- o Push(5)
- o Push(8)
- o Pop()
- o Push(10)

A) [2, 5]

B) [2, 10]

C) [5, 10]

D) [2, 5, 10]

Answer: B) [2, 10]

Very Short Questions on Stack Data Structure

1. **Q: What does LIFO stand for in the context of a stack?**

A: Last In, First Out.

2. **Q: Define the term 'stack' in data structures.**

A: A stack is a linear data structure that allows adding and removing elements only from one end, called the top.

3. **Q: What operation is performed to add an element to a stack?**

A: Push.

4. **Q: What is the result of performing a pop operation on an empty stack?**

A: It raises an error (typically an Underflow in Python).

5. **Q: Explain the push operation in a stack.**

A: The push operation adds an element to the top of the stack.

6. **Q: What is the time complexity of push and pop operations in a stack implemented using a list?**

A: O(1).

7. **Q: In Python, which method is used to remove the top element of a list (acting as a stack)?**

A: pop().

8. **Q: What will be the top element after performing the following operations: push(1), push(2), push(3), pop()?**

A: 2.

9. **Q: Can a stack contain duplicate elements? Why or why not?**
A: Yes, a stack can contain duplicate elements because it does not impose any restrictions on value uniqueness.
10. **Q: What is the difference between a stack and a queue?**
A: A stack follows LIFO (Last In, First Out) while a queue follows FIFO (First In, First Out).
11. **Q: How can you implement a stack using a Python list?**
A: By using the list's `append()` method for push and `pop()` method for pop operations.
12. **Q: What is the function of the peek operation in a stack?**
A: The peek operation returns the top element of the stack without removing it.
13. **Q: What built-in Python function can check if a list is empty?**
A: `len()` (e.g., `len(stack) == 0`).
14. **Q: Describe a real-life application where a stack might be used.**
A: In web browsers, stacks are used to manage the back button functionality.
15. **Q: What happens if you try to push an element onto a full stack?**
A: It raises an overflow error (in a fixed-size implementation); otherwise, it simply adds the element in dynamic implementations.
16. **Q: Is it possible to implement a stack using an array? Explain briefly.**
A: Yes, a stack can be implemented using an array, but it requires careful management of the size and overflow conditions.
17. **Q: How would you check if a stack is empty in Python?**
A: By checking if the list's length is 0, e.g., if `len(stack) == 0`.
18. **Q: Write a Python statement to push an element `x` onto a stack named `stack`.**
A: `stack.append(x)`.
19. **Q: What does the pop method return if the stack is empty?**
A: It raises an Underflow Error in Python.
20. **Q: Explain how the stack is used in function call management in programming.**
A: When a function is called, its information (return address, local variables) is pushed onto the call stack, and when the function returns, this information is popped from the stack.

Short Questions and Answers

1. **Q: What is a stack?**
A: A stack is a linear data structure that follows the Last In, First Out (LIFO) principle, meaning the last element added is the first to be removed.
2. **Q: What operation is used to add an element to a stack?**
A: The operation used to add an element to a stack is called push.
3. **Q: What operation is used to remove the top element from a stack?**
A: The operation used to remove the top element from a stack is called pop.
4. **Q: What will happen if you try to pop an element from an empty stack?**
A: An error will be raised (typically an `IndexError` in Python).
5. **Q: Describe the peek operation in a stack.**
A: The peek operation returns the top element of the stack without removing it.
6. **Q: How do you implement a stack using a list in Python?**
A: A stack can be implemented using a list by using the `append()` method for push and `pop()` method for pop operations.
7. **Q: Can a stack store duplicate values?**
A: Yes, a stack can store duplicate values, as it does not enforce uniqueness among its elements.
8. **Q: What is the time complexity of push and pop operations in a stack implemented using a list?**
A: The time complexity of both push and pop operations is $O(1)$.
9. **Q: What is the main use of stacks in programming?**
A: Stacks are commonly used for function call management, expression evaluation, and backtracking algorithms.
10. **Q: Explain the difference between a stack and a queue.**
A: A stack follows the Last In, First Out (LIFO) principle, while a queue follows the First In, First Out (FIFO) principle.

Long Questions and Answers

1. **Q: Explain the push and pop operations in a stack with an example.**

A: The push operation adds an element to the top of the stack, while the pop operation removes and returns the top element. For example, consider the stack represented as a list:

```
stack = []
stack.append(5) # Push 5
stack.append(10) # Push 10
stack.append(15) # Push 15
print(stack.pop()) # Pop returns 15, stack is now [5, 10]
```

After the pop operation, the top element (15) is removed, leaving the stack with [5, 10].

2. **Q: Describe how to implement a stack using a Python list and provide code for basic stack operations.**

A: A stack can be implemented using a Python list with the following operations:

```
stack = []

# Push operation
def push(element):
    stack.append(element)

# Pop operation
def pop():
    if len(stack) == 0:
        return "Stack is empty"
    return stack.pop()

# Peek operation
def peek():
    if len(stack) == 0:
        return "Stack is empty"
    return stack[-1]

# Example usage
push(1)
push(2)
print(pop()) # Output: 2
print(peek()) # Output: 1
```

3. **Q: What are the applications of stacks in computer science?**

A: Stacks have several applications in computer science, including:

- **Function Call Management:** Stacks are used to manage function calls in programming languages, where each call is pushed onto the stack and popped when the function returns.
- **Expression Evaluation:** Stacks are used to evaluate postfix expressions and to convert infix expressions to postfix.
- **Backtracking Algorithms:** Stacks are used to keep track of choices in backtracking algorithms, such as in maze solving or puzzle games.

4. **Q: Explain the role of the peek operation in stack management and provide an example.**

A: The peek operation allows access to the top element of the stack without modifying it, which is useful for checking the next item to be processed. For example:

```
stack = []
```



```
stack.append(10)
stack.append(20)
top_element = stack[-1] # Peek operation
print(top_element) # Output: 20
```

The peek operation here returns 20 without removing it from the stack.

5. **Q: Describe how a stack can be used in recursion and provide a relevant example.**

A: Stacks are fundamental in managing recursive function calls. When a function calls itself, its context is stored on the call stack. For example:

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)
```

Each call to factorial(n) pushes a new context onto the stack, and as the base case is reached, the contexts are popped off, resolving the recursion.

6. **Q: Explain how to check if a stack is empty and why it is important.**

A: You can check if a stack is empty by verifying the length of the underlying list:

```
if len(stack) == 0:
    print("Stack is empty")
```

This is important to prevent errors during pop operations and to ensure that the program logic correctly handles cases where no elements are available to be removed.

7. **Q: Provide a detailed explanation of stack overflow and underflow.**

A:

- **Stack Overflow:** This occurs when a stack reaches its maximum capacity and an attempt is made to push another element onto it. This can lead to program crashes or errors if not handled properly.
- **Stack Underflow:** This happens when a pop operation is attempted on an empty stack. In this case, the program may raise an error indicating that there are no elements to remove.

Scoring Tips on Stacks

1. **Understand the Basic Concepts:**

- **Know the Definition:** Be clear about what a stack is (a linear data structure following LIFO).
- **LIFO Principle:** Remember that in stacks, the last element added is the first to be removed.

2. **Memorize Operations:**

- **Push Operation:** Know how to add elements to the stack and its implications.
- **Pop Operation:** Understand how to remove elements and handle errors when popping from an empty stack.

3. **Master the Implementation:**

- **Using Lists in Python:** Be familiar with using Python lists for stack implementation. Know the methods:
 - `append()` for push
 - `pop()` for pop
- **Practice Coding:** Write and test simple code snippets to implement a stack. Example:

```
stack = []
stack.append(1) # Push
stack.append(2)
print(stack.pop()) # Pop
```

4. **Error Handling:**

- **Handle Empty Stack Conditions:** Be able to explain or write code that checks for an empty stack before popping. This demonstrates good programming practices.
 - Example:


```
if len(stack) == 0:
    print("Stack is empty!")
else:
    print(stack.pop())
```
5. **Use Real-World Examples:**
 - **Explain with Examples:** Be ready to explain stack concepts using real-world analogies (e.g., a stack of plates, function calls in programming).
 - **Applications:** Mention applications like undo mechanisms in text editors, expression evaluations, and backtracking algorithms.
 6. **Practice Problems:**
 - **Work on Problems:** Solve problems related to stack operations, such as evaluating postfix expressions, checking for balanced parentheses, or converting infix to postfix expressions.
 7. **Be Clear with Definitions:**
 - **Define Terms:** Be clear about related terms like underflow (trying to pop from an empty stack) and overflow (trying to push onto a full stack).
 8. **Visualize the Stack:**
 - **Draw Diagrams:** When explaining stack operations, use diagrams to show the stack's state before and after operations. Visual aids can enhance understanding and presentation.
 9. **Review Previous Year Questions:**
 - **Practice Past Papers:** Go through previous years' question papers to familiarize yourself with the types of questions asked related to stacks. This will help you understand the exam pattern.

Syllabus for 2024-25

Unit 2: Computer Networks

- Evolution of networking: introduction to computer networks, evolution of networking (ARPANET, NSFNET, INTERNET)
- Data communication terminologies: concept of communication, components of data communication (sender, receiver, message, communication media, protocols), measuring capacity of communication media (bandwidth, data transfer rate), IP address, switching techniques (Circuit switching, Packet switching)
- Transmission media: Wired communication media (Twisted pair cable, Co-axial cable, Fiber-optic cable), Wireless media (Radio waves, Micro waves, Infrared waves)
- Network devices (Modem, Ethernet card, RJ45, Repeater, Hub, Switch, Router, Gateway, WIFI card)
- Network topologies and Network types: types of networks (PAN, LAN, MAN, WAN), networking topologies (Bus, Star, Tree)
- Network protocol: HTTP, FTP, PPP, SMTP, TCP/IP, POP3, HTTPS, TELNET, VoIP
- Introduction to web services: WWW, Hyper Text Markup Language (HTML), Extensible Markup Language (XML), domain names, URL, website, web browser, web servers, web hosting

Unit 2: Computer Networks

2.1 Evolution of Networking

2.1.1 Introduction to Computer Networks:

- A computer network is a group of interconnected devices capable of sharing data and resources.
- Devices in a network can include computers, servers, printers, and other peripherals.
- Networks enable efficient communication and resource sharing.

2.1.2 Evolution of Networking:

- ARPANET: The Advanced Research Projects Agency Network, an early packet-switching network and the first network to implement the TCP/IP protocol suite.
- NSFNET: The National Science Foundation Network, a backbone network that supported research and educational institutions.
- INTERNET: The global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide.

2.2 Data Communication Terminologies

2.2.1 Concept of Communication:

- Data communication involves the transfer of data between devices over a communication medium.
- Essential components include sender, receiver, message, communication medium, and protocols.

2.2.2 Components of Data Communication:

- Sender: The device that sends the data.
- Receiver: The device that receives the data.
- Message: The data being communicated.
- Communication Medium: The channel through which the data travels (wired or wireless).
- Protocols: Rules and conventions for data communication (e.g., HTTP, TCP/IP).

2.2.3 Measuring Capacity of Communication Media:

- Bandwidth: The range of frequencies that can be transmitted, measured in Hertz (Hz).
- Data Transfer Rate: The amount of data transmitted per second, measured in bits per second (bps).

2.2.4 IP Address:

- An IP (Internet Protocol) address is a unique identifier assigned to each device connected to a network that uses the Internet Protocol for communication.
- It allows devices to locate and communicate with each other over an IP-based network, such as the internet or a local network.
- IPv4: Consists of four sets of numbers separated by periods (e.g., 192.168.1.1), each ranging from 0 to 255, providing approximately 4.3 billion unique addresses. It is a 32-bit address.
- IPv6: Developed to address the exhaustion of IPv4 addresses, it consists of eight groups of four hexadecimal digits separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334), allowing for a vastly larger number of unique addresses. It is a 128-bit address.

2.2.5 Switching Techniques:

- Circuit Switching: It is a method where a dedicated communication path is established between two devices for the duration of the communication session. This path remains reserved and exclusive for the communicating devices until the session ends.
- Packet Switching: In packet switching, data is broken into small packets that are transmitted independently over the network. Each packet can take different paths to reach the destination, where they are reassembled in the correct order.

2.3 Transmission Media

Transmission media refers to the physical pathways through which data is transmitted from one device to another in a network. It serves as the conduit that carries signals representing data between networked devices. Transmission media can be broadly categorized into two types: wired (guided) and wireless (unguided).

2.3.1 Wired Communication Media:

- Twisted Pair Cable: Consists of pairs of insulated copper wires twisted together to reduce electromagnetic interference.

- Co-axial Cable: Contains a central conductor, insulating layer, metallic shield, and outer insulating layer, used for cable TV and internet.
- Fiber-optic Cable: Transmits data as light pulses through a glass or plastic fiber, offering high speed and low signal loss.

2.3.2 Wireless Communication Media:

- Radio Waves: Used for short to long-distance communication, including broadcast radio, Wi-Fi, and mobile networks.
- Micro Waves: Used for point-to-point communication links and satellite communication.
- Infrared Waves: Used for short-range communication like remote controls and some wireless peripherals.

2.4 Network Devices

- Modem: Modulates and demodulates signals for data transmission over telephone lines.
- Ethernet Card (NIC): Network Interface Card used for connecting a computer to a wired network.
- RJ45: A standard type of connector for network cables.
- Repeater: Regenerates signals to extend the range of a network.
- Hub: Connects multiple Ethernet devices, making them act as a single network segment.
- Switch: Connects devices within a network and uses MAC addresses to forward data to the correct destination.
- Router: Connects different networks and routes data between them.
- Gateway: A node that connects two different networks, often translating different protocols.
- Wi-Fi Card: Allows devices to connect to wireless networks.

2.5 Network Topologies and Types

2.5.1 Types of Networks:

- PAN (Personal Area Network): A small network for personal devices, typically within a range of 10 meters.
- LAN (Local Area Network): Covers a small geographic area like a home, office, or building.
- MAN (Metropolitan Area Network): Covers a larger geographic area such as a city.
- WAN (Wide Area Network): Covers large geographic areas, often a country or continent, connecting multiple LANs and MANs.

2.5.2 Networking Topologies:

Network topology refers to the arrangement of various elements (links, nodes, etc.) in a computer network. It is a key factor in determining the network's performance, scalability, and reliability. Here are some common types of network topologies:

- Bus Topology: All devices are connected to a single central cable.
- Star Topology: All devices are connected to a central hub or switch.
- Ring Topology: In a ring topology, each device is connected to two other devices, forming a circular path for signals to travel.
- Tree Topology: A combination of bus and star topologies, with groups of star-configured devices connected to a central bus.

2.6 Network Protocols

A network protocol is a set of established rules and conventions that dictate how data is transmitted, received, and processed across a network. These protocols define the procedures and formats that devices must follow to ensure successful communication and interoperability between different hardware and software systems. Some examples of Network Protocols are:

- HTTP (HyperText Transfer Protocol): Used for transferring web pages on the internet.
- FTP (File Transfer Protocol): Used for transferring files between computers on a network.

- PPP (Point-to-Point Protocol): Used for direct communication between two nodes.
- SMTP (Simple Mail Transfer Protocol): Used for sending emails.
- TCP/IP (Transmission Control Protocol/Internet Protocol): Fundamental protocol suite for internet and other networks.
- POP3 (Post Office Protocol version 3): Used for retrieving emails from a server.
- HTTPS (HyperText Transfer Protocol Secure): Secure version of HTTP, used for secure communication over a computer network.
- TELNET (Teletype Network): Used for remote communication with another computer.
- VoIP (Voice over Internet Protocol): Used for delivering voice communications over IP networks.

2.7 Introduction to Web Services

A web service is a standardized way of enabling communication and data exchange between different software applications over a network, typically the internet. It uses protocols and technologies such as HTTP, XML, SOAP (Simple Object Access Protocol), and REST (Representational State Transfer) to allow different systems to interact with each other regardless of their underlying platforms or technologies.

- WWW (World Wide Web): A system of interlinked hypertext documents and multimedia content accessible over the internet.
- HTML (HyperText Markup Language): The standard language for creating web pages and web applications.
- XML (Extensible Markup Language): A language for defining and transporting data.
- Domain Names: Human-readable addresses for web servers (e.g., www.example.com).
- URL (Uniform Resource Locator): The address used to access resources on the web.
- Website: A collection of related web pages hosted on a web server.
- Web Browser: Software application used to access and display web pages (e.g., Google Chrome, Firefox).
- Web Servers: Computers that host websites and deliver web pages to users.
- Web Hosting: Service that allows individuals and organizations to make their websites accessible on the internet.

Multiple Choice Questions

1. Which protocol provides the mechanism for dynamic IP address allocation?
a) FTP b) HTTP c) **DHCP** d) SNMP
2. Which protocol is used to send large files over the internet?
a) SMTP b) HTTP c) **FTP** d) POP3
3. Which of the following is a valid IPv4 address?
a) 256.100.50.25 b) **192.168.1.1** c) 300.1.1.1 d) 172.500.0.1
4. What does the term "bandwidth" refer to in networking?
a) **The amount of data a connection can handle in a given time**
b) The distance between two network devices
c) The security level of a network d) The speed of data processing
5. What type of network topology connects all devices to a central hub or switch?
a) Mesh b) **Star** c) Ring d) Bus
6. Which device in a network segment regenerate and amplifies signals to extend the distance of data transmission?
a) Switch b) Hub c) **Repeater** d) Router
7. What is the purpose of a DNS server in a network?
a) **Translate domain names into IP addresses** b) Manage email routing

Answer: A domain name is a human-readable address used to identify a website or resource. It is mapped to an IP address through DNS, enabling browsers to locate the resource.

9. Define network topology and name a common type.

Answer: Network topology refers to the arrangement of network devices and connections. A common type is the star topology, where all devices are connected to a central hub or switch.

10. What is the function of a repeater in a network?

Answer: A repeater amplifies and regenerates signals to extend the distance over which data can travel in a network.

11. What is the difference between IPv4 and IPv6 addresses?

Answer: IPv4 addresses are 32-bit and written in decimal format (e.g., 192.168.1.1), while IPv6 addresses are 128-bit and written in hexadecimal format (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).

12. Describe the purpose of a switch in a network.

Answer: A switch connects multiple devices within a local area network (LAN) and forwards data only to the device for which it is intended, based on MAC addresses.

13. What is the purpose of the DHCP protocol?

Answer: DHCP (Dynamic Host Configuration Protocol) automatically assigns IP addresses and other network configuration parameters to devices on a network.

14. What does the acronym HTTPS stand for, and what is its advantage over HTTP?

Answer: HTTPS stands for Hypertext Transfer Protocol Secure. It provides encrypted communication over the internet, enhancing security and protecting data from eavesdropping.

3 Marks Questions

1. What are the advantages of using a switch over a hub in a network?

Answer: A switch is more efficient than a hub because it forwards data only to the intended recipient based on MAC addresses, reducing network collisions and improving performance. In contrast, a hub broadcasts data to all connected devices, which can lead to network congestion.

2. Explain the difference between a local area network (LAN) and a wide area network (WAN).

Answer: A LAN (Local Area Network) covers a small geographical area, such as a single building or campus, and typically offers high-speed connections. A WAN (Wide Area Network) spans a large geographical area, connecting multiple LANs over long distances, often using public or leased communication lines.

3. Describe the process of data transmission in a bus topology.

Answer: In a bus topology, all devices are connected to a single central cable (the bus). Data transmitted by any device travels along the bus and is received by all other devices. Each device checks the data's destination address and processes it if it matches its own.

4. What is the purpose of the HTTP protocol, and how does it work?

Answer: HTTP (Hypertext Transfer Protocol) is used for transmitting hypertext (web pages) between a client (browser) and a server. It works by sending requests from the client to the server and receiving responses containing the requested resources.

5. Describe how a network switch improves network performance compared to a hub.

Answer: A network switch reduces network collisions and improves performance by forwarding data only to the specific device intended, based on MAC addresses. In contrast, a hub broadcasts data to all devices, leading to potential collisions and reduced efficiency.

6. What is the purpose of a router in a network, and how does it differ from a switch?

Answer: A router connects different networks and routes data packets between them based on IP addresses. It manages traffic between networks, while a switch connects devices within the same network and forwards data based on MAC addresses.

4 Marks Questions

1. Explain the concept of network topology. Compare and contrast bus, star, and ring topologies in terms of their advantages and disadvantages.

Answer: Network topology refers to the arrangement of devices and connections in a network.

- Bus Topology: All devices share a single communication line. It's simple and cost-effective but can suffer from collisions and network failures affecting all devices.
- Star Topology: Devices are connected to a central hub or switch. It's reliable and easy to manage but requires more cable and can be affected if the central device fails.
- Ring Topology: Devices are connected in a circular fashion. It provides predictable performance but can be disrupted if any single connection fails.

Note : Do practice more and more case based questions from Sumita Arora's book "Computer Science with Python" class XII. It will give you confidence to grasp more concepts.

Success Tips: There is no alternative to self study. Study with yourself and discuss with your peers. By this activity your concept will be more clear.

UNIT III

DATABASE MANAGEMENT SYSTEM

Introduction to database concepts

Data: Data is simply a collection of raw facts, figures, numbers, or observations.

Database: Its collection of interrelated data store to serve multiple application.

DBMS (DATABASE MANAGEMENT SYSTEM): A database management system (DBMS) is a software that is responsible for storing, maintaining and utilizing databases. Examples- MySQL, Oracle, PostgreSQL, SQL Server, Microsoft Access, MongoDB.

Need of DBMS:

Keeping organizational information in a file- processing system has a number of major disadvantages like:

- a) Data redundancy and inconsistency
- b) Difficulty in accessing data
- c) Data isolation
- d) Integrity problems
- e) Atomicity problems
- f) Concurrent access anomalies
- g) Security problems

DBMS lets users to create a database, store, manage, update/modify and retrieve data from that database by users or application programs.

Some of the advantages are given below –

- a) Reducing Data Redundancy
- b) Sharing of Data
- c) Data Integrity
- d) Data Security
- e) Privacy
- f) Backup and Recovery
- g) Data Consistency

Database model: It defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. There are four data models in DBMS:

- a) Relational Data Model
- b) Hierarchical Data Model
- c) Network Model
- d) Object Oriented Data Model

Relational model: In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field. This model was introduced by E.F Codd in 1970. Each relation(table) is a collection of columns and rows.

- **Relation:** A relation is a table with rows and columns. Each row represents a unique record, and each column represents a field or attribute of the record.
- **Attributes:** The columns of the table are called attributes. Each attribute represents a property of relation.
- **Tuples:** The rows of the table are called tuples. Each tuple represents a single record.
- **Domain:** It's a set of pre-defined values that an attribute can take i.e. it describes the allowable values that an attribute can take.
- **Degree:** It refers to the number of attributes (columns) in a relation (table).
- **Cardinality:** It specifies the number of entities involved in the relation i.e., it is the total number of rows present in the relation.
- **Primary Key –** A primary is an attribute or set of attributes in a relation that uniquely identifies tuples (rows) in that relation.

- **Candidate Key** –It is an attribute or a set of attributes or keys participating for Primary Key, to uniquely identify each tuple in that relation.
- **Alternate Key** – A candidate key that is not the primary key is called alternate key or secondary key.

Student_Id	Name	F_Name	Dob	Class	Phone_No
101	Anu	SK		X	1234567890
102	Sumit	DK		IX	0123456789
103	Disha	AK		X	1122334455

Row/Tuple

Attribute/Column

Primary Key: Student_Id

Candidate Key: {Name, F_Name}

- **Foreign Key** – Foreign keys are the attributes of a relation that points to the primary key of another relation

Primary Key TABLE: CUSTOMER (Parent Table)

Cust_ID	First_Name	Last_Name

Primary Key TABLE: ORDERS (Child Table) Foreign Key

Order_No.	Order_Date	Cust_ID	Amount

SQL (Structured Query Language): SQL is a non-procedural language that enables us to create and operate on relational databases, which are sets of related information stored in tables.

MySQL: It is freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

Features of MySQL:

- a) Cost: Freely available
- b) Performance: MySQL is fast.
- c) Simplicity: MySQL is easy to install.
- d) High Performance
- e) Data Security: In has powerful mechanism for ensuring only authorized users have access the data.

Difference between SQL and MySQL:

SQL	MySQL
This is the language that used in database.	This is the open source database.
This language is used in MySQL to write the commands in queries.	This is the database. It uses the SQL to write the queries.

Classification of SQL commands:

1. **Data Definition Language (DDL):** It consist the commands to create objects such as tables, views, indexes etc. in the database.

COMMANDS: CREATE, ALTER, DROP, RENAME, TRUNCATE

2. **Data Manipulation Language (DML):** It is used for queries. It allows you to perform data manipulation e.g. retrieval, insertion, deletion, modification of data stored in database.

COMMANDS: SELECT, INSERT, UPDATE, DELETE

3. **Transaction Control Language (TCL):** This language allows you to manage and control the transaction.

COMMANDS: COMMIT, ROLLBACK

4. Data Control Language (DCL): This language is used to control data and access to the databases. It is used for protecting the data from unauthorized access.

COMMANDS: GRANT, REVOKE

DATA TYPES

Data types define the kind of values that can be stored in each column of a table. MySQL supports a wide range of data types, which can be broadly classified into three categories: Text Data Type, Numeric types, Date and time types.

1. Text Data Types

Char(size) – fixed length of size bytes

Varchar(size)-variable length of size bytes

Char DataType	Varchar DataType
It specifies a fixed length character String.	It specifies a variable length character string.
When a column is given datatype as CHAR(n), then MySQL ensures that all values stored in that column have this length i.e. n bytes. If a value is shorter than this length n then blanks are added, but the size of value remains n bytes.	When a column is given datatype as VARCHAR(n), then the maximum size a value in this column can have is n bytes. Each value that is stored in this column store exactly as you specify it i.e. no blanks are added if the length is shorter than maximum length n.

2. Number Data Types

Integer(size)or Int- It represents a number without decimal point

Float (Size)-It represents a floating-point number

3. Date Data Type

Date

SOME IMPORTANT POINTS:-

- SQL statements are not case sensitive.
- To end the SQL command, we write the semicolon (;) at the end of a line followed by <ENTER>

- **NULL Values:** If a column in a row has no value, then column is said to be null. NULLs can appear in a column if the column is not restricted by NOT NULL or Primary Key. You should use a null value when the actual value is not known. Null and Zero is not equivalent. Any arithmetic expression containing a null always evaluates to null.

- Example: $7 + \text{null} = \text{null}$

- $7 + 0 = 7$

} Difference between null and zero.

- **Comments:** Comment is a text that is not executed. There are two types of comments that we use in MySQL.

- Single line Comment: Beginning with two hyphens (--) OR beginning with # symbol.

- Multi Line Comment: /*.....*

.....*/

CONSTRAINT The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table. They ensure data accuracy and integrity inside the table.

Types of Constraints:

- a) **Unique:** - The UNIQUE constraint in MySQL ensures that all values in a column or a set of columns are unique. This prevents duplicate values from being inserted into the column(s) on which the constraint is applied. There can be many unique constraints defined on a table.

Example: CREATE TABLE Employees (EmployeeID INT , Email VARCHAR(100) UNIQUE, FirstName VARCHAR(100), LastName VARCHAR(100));

- b) **Primary Key:** The Primary Key constraint uniquely identifies each record in a table. A primary key column must contain unique values, and it cannot contain NULL values. Each table can have only one primary key, which can consist of single or multiple columns (composite key)

Example 1: Primary Key on a Single Column

CREATE TABLE Employees (EmployeeID INT NOT NULL PRIMARY KEY, FirstName VARCHAR(100), LastName VARCHAR(100), Email VARCHAR(100));

Example 2: Composite Primary Key

CREATE TABLE OrderDetails (OrderID INT, ProductID INT, Quantity INT, PRIMARY KEY (OrderID, ProductID));

c) **Not null:** This constraint ensures column should not contain NULL

Example: CREATE TABLE Employees (EmployeeID INT NOT NULL UNIQUE, FirstName VARCHAR(100) NOT NULL, LastName VARCHAR(100), Email VARCHAR(100) UNIQUE);1.8
DATABASE COMMANDS IN MYSQL

d) **FOREIGN KEY:** The constraint that joins two tables together is known as a foreign key.

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

e) **DEFAULT** – This constraint sets a default value for a column if no value is specified.

Example: CREATE TABLE Orders (ID int NOT NULL, OrderNumber int NOT NULL, OrderDate date DEFAULT CURRENT_DATE ());

CREATE DATABASE: Used for creating a database in MySQL.

Syntax: CREATE DATABASE database_name;

Example: mysql> CREATE DATABASE School;

SHOW DATABASES: Display list of existing databases

Syntax: SHOW DATABASES;

USE: Select a particular database.

Syntax: USE database_name;

Example: mysql> USE School;

DROP DATABASE: Used to drop an existing SQL database. It is used to delete a table definition and all data from a table.

Syntax :DROP DATABASE database_name;

Example: mysql> DROP DATABASE School;

SHOW TABLES: Used to retrieve the names of tables that are present in a specific database.

Syntax : SHOW TABLES;

CREATE TABLE: Used to create a new table in a database.

Syntax:

```
CREATE TABLE table_name (column1 datatype constraints , column2 datatype constraints,
```

.....

.....

);

```
CREATE TABLE EMPLOYEE(Ecode int(6), Ename varchar(30), Dept varchar(30), city varchar(25), sex char(1), DOB Date, salary float(12,2) );
```

Ecode	Ename	Dept	City	Sex	Dob	Salary

Fig. : EMPLOYEE

DESCRIBE TABLE: Shows the structure of the table, such as column names, constraints on column names etc. The DESC command is a short form of the DESCRIBE command.

Note:Both DESCRIBE and DESC commands are equivalent.

Syntax: DESCRIBE table_name;

Example: mysql> DESCRIBE Students;

ALTER TABLE: Used to add, delete, or modify columns/constraints in an existing table.

1. ALTER TABLE - ADD Column/Attribute

syntax: ALTER TABLE table_name ADD column_name datatype;

Example ALTER TABLE Customers ADD Email varchar(255);

2. ALTER TABLE - DROP COLUMN

Syntax: ALTER TABLE table_name DROP COLUMN column_name;

Example ALTER TABLE Customers DROP COLUMN Email;

3. ALTER TABLE – ADD PRIMARY KEY

Syntax: ALTER TABLE table_name ADD PRIMARY KEY (Column_name);

Example ALTER TABLE Persons ADD PRIMARY KEY (ID);

4. ALTER TABLE – DROP PRIMARY KEY

Syntax: ALTER TABLE table_name DROP PRIMARY KEY;

Example: ALTER TABLE Persons DROP PRIMARY KEY;

DROP TABLE: Used to drop an existing table in a database.

Syntax: DROP TABLE table_name;

Example: mysql> DROP TABLE Students;

MANIPULATING DATA OF A TABLE: -

- a) Inserting data : INSERT command
- b) Retrieving data : SELECT command
- c) Deleting data : DELETE command
- d) Modification : UPDATE command

a) **INSERT INTO:** Used to insert new records in a table.

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

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

Syntax 2: Add all column data without specifying column names

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

Example: mysql> INSERT INTO Students (StudentID, FirstName, LastName, Age, Email) VALUES (1, 'John', 'Doe', 20, 'john.doe@example.com');

b) **SELECT**: A SELECT command is used to retrieve information from a table.

In SQL queries, we use three clauses mostly:-

SELECT:- What to select (columns)

FROM:- Which Table (specifies the table name that contains the columns.)

WHERE:- Specifies Conditions to satisfy

- **Selecting All Columns:-** To select all the columns, we use asterisk (*) in SELECT statement.

Example:- SELECT * FROM EMPLOYEE;

SQL allows us to use the keyword ALL to specify explicitly that duplicates are not removed.

- **Selecting Specific Columns:-** To display the specific columns of the table, write columns name, separated by commas.

Example:- SELECT Ecode, Ename, salary FROM EMPLOYEE;

- **Eliminating redundant data:-** The DISTINCT keyword eliminates duplicate rows from the results. DISTINCT is used in SELECT statement.

Example:- SELECT DISTINCT(Dept) FROM EMPLOYEE;

Example: SELECT ALL Dept FROM EMPLOYEE;

- **COLUMN ALIAS:-** We can change a column heading by using a column alias.

An alias only exists for the duration of that query. An alias is created with the AS keyword.

Example:- SELECT Ename as Name FROM EMPLOYEE;

Alias Table

Syntax: SELECT column_name(s) FROM table_name AS alias_name;

Operators

1. Mathematical Operators

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

- Modulo (%)

2. **Relational Operators** - Relational operators are used to compare values in queries. They return a boolean result (TRUE, FALSE, or UNKNOWN).

- **Equal to (=):** Checks if two values are equal.
- **Not equal to (!= or <>)**
- **Greater than (>)**
- **Less than (<)**
- **Greater than or equal to (>=)**
- **Less than or equal to (<=)**

3. **Logical Operators**

SQL AND, OR and NOT Operators

The AND operator displays a record if all the conditions separated by AND are TRUE.

Syntax:

```
SELECT column1, column2, ...FROM table_name WHERE condition1 AND condition2 AND condition3 ...;
```

The NOT operator displays a record if the condition(s) is NOT TRUE

Syntax:

```
SELECT column1, column2, FROM table_name WHERE NOT condition;
```

The OR operator displays a record if any of the conditions separated by OR is TRUE.

Syntax:

```
SELECT column1, column2, ...FROM table_name WHERE condition1 OR condition2 OR condition3 ...;
```

4. **The SQL IN Operator**

The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Syntax

```
SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);
```

or:

```
SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT STATEMENT);
```

Example:- Find the name of those employees who live in guwahati, surat or jaipur city.

Solution:-

```
SELECT Ename, city FROM EMPLOYEEWHERE city IN('Guwahati','Surat','Jaipur');
```

5. The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

Syntax

```
SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;
```

Example:- Find the name and salary of those employees whose salary is between 35000 and 40000.

Solution:-

```
SELECT Ename, salary FROM EMPLOYEE WHERE salary BETWEEN 35000 and 40000;
```

6. The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign () represents one, single character

The percent sign and the underscore can also be used in combinations!

Syntax

```
SELECT column1, column2, ...FROM table_name WHERE columnN LIKE pattern;
```

Example:-Display the name of those employees whose department name ends with 'a'.

Solution:-

```
SELECT Ename FROM EMPLOYEE WHERE Dept LIKE '%a';
```

Example:- List the name of employees whose name having 'e' as the second character.

Solution:- SELECT Ename FROM EMPLOYEE WHERE Ename LIKE '_e%';

IS NULL

It is not possible to test for NULL values with comparison operators, such as =, <, or >. We will have to use the IS NULL and IS NOT NULL operators instead.

Syntax:

```
SELECT column_names from table_name WHERE column_name IS NULL;
```

```
SELECT column_names from table_name WHERE column_name IS NOT NULL;
```

Example:- List the name of employees not assigned to any department.

Solution:-

```
SELECT Ename FROM EMPLOYEE WHERE Dept IS NULL;
```

The SQL ORDER BY Keyword

The ORDERY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax

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

Example: - Display the list of employees in descending order of employee code.

Solution: - SELECT * FROM EMPLOYEE ORDER BY ecode DESC;

Example: - Display the employee code, name in ascending order of salary.

Solution: - SELECT Ecode, Ename, salary FROM EMPLOYEE ORDER BY salary asc;

UPDATE Table

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city. Example:

UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

The SQL DELETE Statement:

The DELETE statement is used to delete existing records in a table.

DELETE Syntax:

DELETE FROM table_name WHERE condition;

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

The following SQL statement deletes all rows in the "Customers" table, without deleting the table: Example:

DELETE FROM Customers;

Aggregate (Group) function

Aggregate functions are the functions that operate on a set of rows to give one result per group.

These sets of rows on which group function is applied may be the whole table or the table split into groups.

Types of Group Functions

Function	Description
sum()	Find the sum of numeric values
avg()	Find the average of numeric values

count()	Counts the number of rows in a table
max()	Find the maximum among all the values in a column
min()	Find the minimum among all the values in a column

Remember the following points about group functions:

- All group functions, except count(*) ignore NULL values.
- Functions -sum(), avg() are used with NUMERIC data..
- Functions -min() and max() can be used with any data type.
- Count() function Count () has got three formats:
 1. Count(*) This function returns the number of rows in the table that satisfy the criteria of select statement. In its counting, it includes duplicate rows and rows with NULL values in any of the column Example: Q: Count the number of employees in the employee table.
 2. Count(<col name>) This function returns the number of not null values in the specified column, but includes duplicate values in counting
 3. Count(DISTINCT <col name>) This function returns the number of unique, not null values in the specified column.

NOTE: - The input to sum () and avg() must be a collection of numbers, but the other functions can operate on non-numeric data types e.g.string.

Example: Find the average salary of the employees in employee table.

```
SELECT avg(salary) FROM EMPLOYEE;
```

Grouping Records (Group by clause)

- To divide the rows in a table into smaller groups of information, group by clause is used.
- It combines all identical rows in a group of fields.
- A column name is used for grouping

Syntax:-

```
SELECT [DISTINCT] <COL LIST> FROM <TABLE NAME> [WHERE <CONDITION>]
[GROUP BY < GROUP BY EXPR>] [HAVING <CONDITION>]
[ORDER BY <COL NAME>/<EXPR> ASC/DESC];
```

NOTE -

- Group by expression specifies the columns whose values determine the basics for grouping rows
- WHERE clause is always before GROUP BY if required. Example

Example: Consider the following table employee

```
mysql> select * from employee;
```

No	Name	Salary	Zone	Age	Grade	Dept
1	Mukul	30000	West	28	A	10
2	Kritika	35000	Centre	30	A	10
3	Naveen	32000	West	40	NULL	20
4	Uday	38000	North	38	C	30
5	Nupur	32000	East	26	NULL	20
6	Moksh	37000	South	28	B	10
7	Shelly	36000	North	26	A	30

7 rows in set (0.00 sec)

Q: Find the sum, average, minimum, maximum value of salaries of employees in the

Q. Display the no of employees in each zone.

```
mysql> select zone,count(zone) from employee group by zone;
```

zone	count(zone)
Centre	1
East	1
North	2
South	1
West	2

5 rows in set (0.01 sec)

Q. Display the no of employees in each zone whose salary is greater than 32000

```
mysql> select zone,count(zone) from employee where salary>32000 group by zone;
```

zone	count(zone)
Centre	1
North	2
South	1

3 rows in set (0.00 sec)

Having clause

- This clause is used to restrict rows resulting after grouping.
- Steps followed in execution of select with group by and having clause-
 1. Rows are grouped according to the columns in the group by clause.
 2. Then the group function is applied.
 3. Groups matching with Having clauses are displayed.

Example:

Q. Display only those departments with sum of salaries whose total salary is greater than 70000

```
mysql> select dept,sum(salary) from employee group by dept having sum(salary)>70000;
+-----+
| dept | sum(salary) |
+-----+
| 10   | 102000     |
| 30   | 74000      |
+-----+
2 rows in set (0.00 sec)
```

Cartesian Product (Cross Join or unrestricted join)

- Returns all the rows in the two tables listed in the query.
- Each row of the first table is paired with all the rows in the second table.
- This happens when there is no relationship between two tables.
- It is rarely useful.

Example- Consider the following tables

```
mysql> SELECT * FROM EMPLOYEE;
+-----+-----+-----+-----+-----+-----+-----+
| No  | Name      | Salary | Zone   | Age  | Grade | Dept |
+-----+-----+-----+-----+-----+-----+-----+
| 1   | Mukul     | 30000  | West  | 28   | A     | 10   |
| 2   | Kritika   | 35000  | Centre| 30   | A     | 10   |
| 3   | Naveen    | 32000  | West  | 40   | NULL  | 20   |
| 4   | Uday      | 38000  | North | 38   | C     | 30   |
| 5   | Nupur     | 32000  | East  | 26   | NULL  | 20   |
| 6   | Moksh     | 37000  | South | 28   | B     | 10   |
| 7   | Shelly    | 36000  | North | 26   | A     | 30   |
| 8   | Mukul     | 40000  | West  | 50   | C     | 10   |
+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> SELECT * FROM DEPARTMENT;
+-----+-----+
| Dept | Dname      |
+-----+-----+
| 10   | Mechanical |
| 20   | Electrical  |
| 30   | Computer Sci |
+-----+-----+
3 rows in set (0.00 sec)
```

Q: To display the name of the employees and their department name

```
mysql> select Name,Dname from employee,department;
+-----+-----+
| Name      | Dname      |
+-----+-----+
| Mukul     | Mechanical |
| Mukul     | Electrical  |
| Mukul     | Computer Sci |
| Kritika   | Mechanical |
| Kritika   | Electrical  |
| Kritika   | Computer Sci |
| Naveen    | Mechanical |
| Naveen    | Electrical  |
| Naveen    | Computer Sci |
| Uday      | Mechanical |
| Uday      | Electrical  |
| Uday      | Computer Sci |
| Nupur     | Mechanical |
| Nupur     | Electrical  |
| Nupur     | Computer Sci |
| Moksh     | Mechanical |
| Moksh     | Electrical  |
| Moksh     | Computer Sci |
| Shelly    | Mechanical |
| Shelly    | Electrical  |
| Shelly    | Computer Sci |
| Mukul     | Mechanical |
| Mukul     | Electrical  |
| Mukul     | Computer Sci |
+-----+-----+
24 rows in set (0.00 sec)
```

Joins in MySQL

- A join is used when data from two or more tables is required.
- Rows in one table can be joined to the rows in another table based on the common values existing in corresponding columns of two tables.
- Joins are used to retrieve data from tables related to each other with primary- foreign key relationships.
- There are many types of join

Equi join

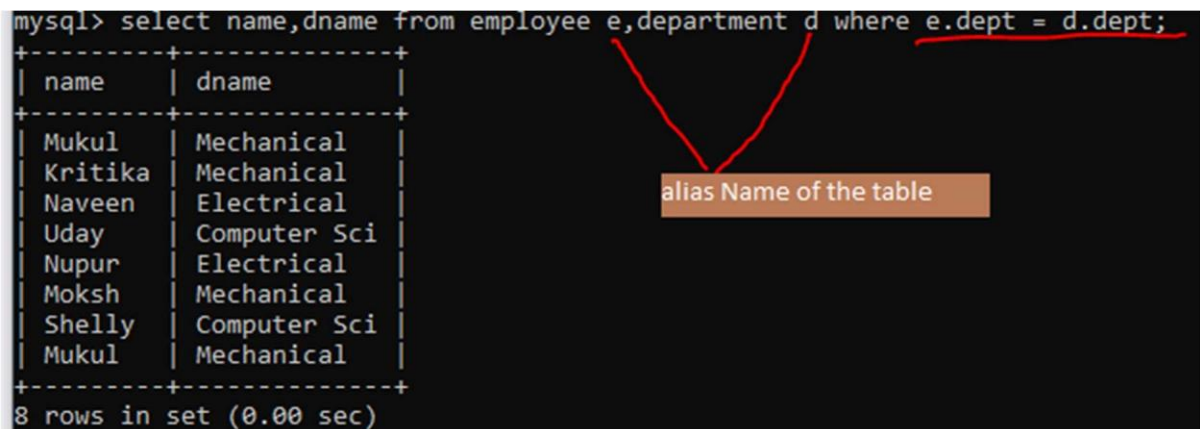
- Specified columns from the joining tables are checked for equality.
- Values from joining tables are retrieved only if the condition in where clause is satisfied.

SYNTAX:-

```
SELECT <column_name (s)> FROM <table_name1>, <table_name2>,....., <table_nameN>
```

WHERE <table_name1>.<column_name> = <table_name2>.<column_name>; Q: to display the name of the employee and their department

```
mysql> select name,dname from employee e,department d where e.dept = d.dept;
+-----+-----+
| name  | dname |
+-----+-----+
| Mukul | Mechanical |
| Kritika | Mechanical |
| Naveen | Electrical |
| Uday  | Computer Sci |
| Nupur | Electrical |
| Moksh | Mechanical |
| Shelly | Computer Sci |
| Mukul | Mechanical |
+-----+-----+
8 rows in set (0.00 sec)
```



Note-

- You should always qualify the columns when joining tables having the same name as corresponding columns. To qualify the columns we use “.” (dot) operator.

Natural Join

This clause is based on all the columns in the two tables that have the same name. It selects the rows from two tables that have equal values in the matched columns. SYNTAX:-

```
SELECT [column_names | *] FROM table_name1 NATURAL JOIN table_name2
```

Example- consider the same tables employee and department.

Q : To display the name of employee and department of all employee


```
mysql> select Name,dept,Dname from employee natural join department;
```

Name	dept	Dname
Mukul	10	Mechanical
Kritika	10	Mechanical
Naveen	20	Electrical
Uday	30	Computer Sci
Nupur	20	Electrical
Moksh	10	Mechanical
Shelly	30	Computer Sci
Mukul	10	Mechanical

8 rows in set (0.00 sec)

Appears only once

Multiple choice Questions(MCQ):

Q1. What is not true in respect of DBMS?

- (a) Database enforces standards
- (b) Database increases redundancy
- (c) Database facilitates sharing of data
- (d) Database helps to maintain integrity

Q2. A data _____ is a set of rules that define valid data.

- a) Query b)Constraint c)Dictionary d)All of the above

Q3. A relational database consists of a collection of

- a) Fields b)Records c)Keys d)Tables

Q4. A row in a database.

- a) Field b)Record c)Key d)Table

Q5. The term _____ is used to refer to a field in a table.

- a) Attribute b)Row c)Tuple d)Instance

Q6. Which of the following statements is not true about relational database?

- a) Relational data model is the most widely used data model.
- b) The data is arranged as a collection of tables in relational database.
- c) Relational database increases data redundancy and inconsistency.
- d) None of the above.

Q7. In a table in MYSQL database with two tuples, an attribute DOB of type date has the value '1988-07-24'. The same attribute DOB has value '1978-01-05' for another tuple. The oldest person can be queried by using?

- a. MAX b. MIN c. ORDER BY d. Sub queries

Q8. In a table in the MYSQL database, there are 10 attributes, how many of these attributes can act as primary key(s)?

- a. 1 b. 9 c. All the 10 attributes d. Can't be predicted

Q9. Which clause is used with aggregate functions?

- a. GROUP BY b. ORDER BY c. BETWEEN d. IN

Q10. Choose the correct statement that better explains the term, Alternate keys in a relational database

- a. The set of all attributes which are eligible to be chosen as the Primary key
b. The set of all attributes which are eligible to be chosen as the Foreign key
c. The set of all attributes in a relation, which were eligible to have become a primary key, but were not chosen as primary key
d. The set of all attributes in a relation, which were eligible to have become a primary key, but were not chosen as foreign key

Q11. Which type of values will not be considered by SQL while executing the following statement?

SELECT COUNT (column name) FROM STOCK;

- a. Null value b. Text value c. Numeric value d. Date value

Q12. Which of the following constraint is used to prevent a duplicate value in a record?

- (a) Empty (b) check (c) foreign key (d) unique

Q13. The structure of the table/relation can be displayed using _____ command.

- (a) view (b) describe (c) show (d) select.

Q14. In MySQL, which function is used to count the number of rows in a table?

- (a) COUNT(*) (b) SUM(*) (c) TOTAL(*) (d) AVG(*)

Q15. The term _____ is used to refer to a field in a table.

- (a) Attribute (b) Row (c) Tuple (d) Instance

Q16. Table: Employee

StudentId	Name	Class
101	Rohit	X
102	Rehansh	X
103	Ayushi	X
104	Rani	X

Answer the questions based on the table Employee.

If two columns are added to the table Student, then the cardinality and degree of the table is and respectively.

- (a) 4 , 5 (b) 7, 4 (c) 6,5 (d) 5,6

Q17. An attribute in a relation is a foreign key if it is the _____ key in any other relation.

- (a) Candidate (b) Primary (c) Unique Key (d) None of the above

Q18. If there are 200 students enrolled in the school, what is the cardinality of the Students table?

- (a) 3 (b) 200 (c) 4 (c) 600

Q19. Which of the following SQL statements will select all records where the column name starts with the letter 'A'?

- (a) SELECT * FROM table WHERE name LIKE '%A';
(b) SELECT * FROM table WHERE name LIKE 'A%';
(c) SELECT * FROM table WHERE name LIKE '%A%';
(d) SELECT * FROM table WHERE name = 'A';

Q20. How can you modify the SQL statement SELECT * FROM books WHERE title LIKE 'A%'; to also include titles that end with 'B'?

- (a) SELECT * FROM books WHERE title LIKE 'A%' OR title LIKE '%B';

(b) SELECT * FROM books WHERE title LIKE 'A%B';

(c) SELECT * FROM books WHERE title LIKE 'A%' AND title LIKE '%B';

(d) SELECT * FROM books WHERE title = 'A%B';

Q21 to Q24 are ASSERTION AND REASONING based questions.

Mark the correct choice as

- a. Both A and R are true and R is the correct explanation for A
- b. Both A and R are true and R is not the correct explanation for A
- c. A is True but R is False
- d. A is false but R is True

Q21. Assertion (A): In SQL, the aggregate function Avg() calculates the average value on a set of values and produce a single result.

Reason (R): The aggregate functions are used to perform some fundamental arithmetic tasks such as Min(),Max(), Sum()

Q22. Assertion (A): In SQL, NULL values can be compared using the equality operator (=).

Reason (R): NULL represents an unknown value, and comparing it with any other value is not meaningful.

Q23. Assertion (A): The JOIN clause in SQL is used to combine rows from two or more tables.

Reason (R): The JOIN clause requires at least one column from each table to be the same in both tables.

Q24. Assertion (A): The GROUP BY clause in SQL is used to arrange identical data into groups.

Reason (R): The GROUP BY clause can be used with aggregate functions to produce summary reports.

ANSWERS: 1. Multiple Choice Questions (MCQ)

1	B	2	B	3	D	4	B
5	A	6	C	7	B	8	D
9	A	10	C	11	A	12	D
13	B	14	A	15	A	16	A
17	B	18	B	19	B	20	A
21	A	22	D	23	B	24	A

Very Short Answer Questions:

1. What is meant by a database?

Ans. A database is an organized collection of structured information, or inter-related data, typically stored in a computer system.

2. What is primary key?

Ans. A primary key is a column or set of columns that contain values that uniquely identify each row in a table.

3. What do you mean by candidate key?

Ans. It is an attribute or a set of attributes or keys participating for Primary Key, to uniquely identify each record in that table.

4. What is meant by degree and cardinality of a table?

Ans. Degree refers to the number of attributes/columns in a relation. Cardinality refers to the number of tuples/rows in a relation

5. What is meant by RDBMS?

Ans. RDBMS (relational database management system) is the software used to store, manage, query, and retrieve data stored in a relational database. The RDBMS provides an interface between users and applications and the database, as well as administrative functions for managing data storage, access and performance.

6. What is meant by database schema?

Ans. Database schema is also called the visual or logical architecture as it tells us how the data are organized in a database.

7. In MySQL, specify the order of execution and the order of writing for the following SQL clauses:

SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING.

i. Execution Order:

ii. Writing Order:

Ans. i. Execution Order:

FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY

ii. Writing Order:

SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY

8. Differentiate between having and where clause data types with respect to databases.

WHERE	HAVING
Filter data From the table based on the specified condition.	Filter data From the groups based on the specified condition.
This clause is used with a single row function.	This clause is used with a multiple-row function.

9. Differentiate between count() and count(*) functions in SQL with appropriate example.

Count()	Count(*)
Counts all rows in the result set, including rows with NULL values.	Counts non-NULL values in a specified column.

Used when you want the total number of rows in a table or result set.	Used when you want to count only the non-NULL entries in a specific column..
---	--

10. Differentiate between DELETE and DROP table commands ?

Provide an example SQL query to illustrate both orders.

Ans.

DELETE	DROP
Removes specific rows from a table	Removes the entire table and its structure.
Data is removed based on a condition.	All data in the table is removed.
Ex: DELETE FROM employees WHERE age > 60;	Ex: DROP TABLE employees;

11. Vishal is using a table Employee which has columns **Code,Name,Salary,Deptcode**. He wants to display maximum salary departments wise but he did not get the desired result. Rewrite the query given below with necessary change to help him get the desired result.

SELECT Deptcode , MAX(Salary) FROM Employee;

Ans: SELECT Deptcode, MAX(Salary) AS MaxSalary FROM Employee GROUP BY Deptcode;

12. .Give difference between primary key and foreign key.

Ans.

PRIMARY KEY	FOREIGN KEY
Uniquely identifies each record in a table.	Establishes a link between records in two tables.
Must be unique for each record in the table.	Can have duplicate values; its uniqueness is not enforced.
Cannot accept NULL values.	Can accept NULL values, depending on the relationship.

Short Answer Questions

Q1. Student Table

RollNo	Name	Class	DOB	Gender	City	Marks
1	Nanda	X	1995-06-06	M	Agra	551
2	Saurabh	XII	1993-05-07	M	Mumbai	462
3	Sonal	XI	1994-05-06	F	Delhi	400
4	Trisla	XII	1995-08-08	F	Mumbai	450
5	Store	XII	1995-10-08	M	Delhi	369
6	Marisla	XI	1994-12-12	F	Dubai	250
7	Neha	X	1995-12-08	F	Moscow	377
8	Nishant	X	1995-06-12	M	Moscow	489

- (i) Display the student record in alphabetical order as per the name of student
- (ii) Display Class, DOB and City whose marks is between 450-551
- (iii) Display Total number of record

Ans.

- (i) Select * from Student Order by name;
- (ii) Select Name, Class, DOB, City from Students where marks between 450 and 551;
- (iii) Select COUNT(*) AS total_records FROM student;

Q2. Write the SQL queries (i) to (iii) based on the table given below:

Table: Salesman

SNO	SNAME	SALARY	BONUS	DATE_OF_JOIN	Storetype
A01	ANIL	30000	45.23	29-10-2019	Grocery
A02	DEEPAK	50000	25.34	13-03-2018	Grocery
B03	NIRBHAY	30000	35.00	18-03-2017	Hardware
B04	UBESH	80000	NULL	31-12-2018	Hardware
C05	AATIF	20000	NULL	23-01-1989	Health
C06	KRITIKA	50000	25.34	13-03-2018	Health
E07	ANGEL	30000	35.00	18-03-2017	Software

E08	PAYAL	80000	NULL	31-12-2018	Software
-----	-------	-------	------	------------	----------

- i. Calculate the total salary of salesman whose salary is greater than 20000 and not getting bonus
- ii. Calculate the total of maximum salary and minimum bonus for salesmen working in Grocery store and adds these values together.
- iii. Display all salesman who is not getting bonus.

Ans.

i. `SELECT SUM(SALARY) AS total_salary FROM Salesman WHERE BONUS IS NULL AND SALARY > 20000;`

ii. `SELECT MAX(SALARY) + MIN(BONUS) AS combined_value FROM Salesman WHERE storetype = 'Grocery';`

iii. `SELECT SNAME AS Name FROM Salesman WHERE BONUS IS NULL`

Q3.

SetCode	SetName	TouchScreen	PhoneCost
N1	Nokia 2G	N	5000
N2	Nokia 3G	Y	8000
B1	BlackBerry	N	14000

CustNo	SetNo	CustAddress
1	N2	Delhi
2	B1	Mumbai
3	N2	Mumbai
4	N1	Kolkata
5	B1	Delhi

With reference to these tables, Write commands in SQL for (i) and (ii) and output for (iii)

below:

(i) Display the CustNo, CustAddress and corresponding SetName for each customer.

(ii) Display the Customer Details for each customer who uses a Nokia handset.

(iii) select SetNo, SetName from Handsets, customer where SetNo = SetCode and CustAddress = 'Delhi';

Ans

(i) `SELECT CustNo, CustAddress, SetName FROM Customer JOIN Handsets ON Customer.SetCode = Handsets.SetNo;`

(ii) `SELECT CustNo, CustName, CustAddress, SetName FROM Customer JOIN Handsets ON Customer.SetCode = Handsets.SetNo WHERE SetName LIKE '%Nokia%';`

(iii) `SELECT SetNo, SetName FROM Handsets JOIN Customer ON Handsets.SetNo = Customer.SetCode WHERE CustAddress = 'Delhi';`

Long Answer Questions:

Q1. COMPUTER

PROD_ID	PROD_NAME	PRICE
1	Laptop	75000.00
2	Desktop	45000.00
3	Tablet	30000.00
4	Smartphone	25000.00
5	Monitor	15000.00

SALES

SALE_ID	PROD_ID	QTY_SOLD	QUARTER
1	1	50	1
2	2	30	1
3	1	20	2
4	3	40	2
5	4	10	3
6	5	25	3
7	2	15	4
8	3	35	4
9	4	20	4
10	5	30	4

Write SQL queries for the following:

- (i) Display PROD_NAME and QTY_SOLD from the tables COMPUTER and SALES.
- (ii) Display the structure of the table COMPUTER.
- (iii) Delete last 3 data of table SALES.
- (iv) Display the total QTY_SOLD in each QUARTER for only Laptop and Tablet

Ans.

- (i)

```
SELECT COMPUTER.PROD_NAME, SALES.QTY_SOLD FROM COMPUTER, SALES
WHERE COMPUTER.PROD_ID = SALES.PROD_ID;
```
- (ii)

```
DESCRIBE COMPUTER;
```
- (iii)

```
SELECT SALES_ID FROM SALES s1
WHERE 3 >= ( SELECT COUNT(*) FROM SALES s2 WHERE s2.SALES_ID > s1.SALES_ID
);
```
- (iv)

```
SELECT QUARTER, SUM(QTY_SOLD) AS total_qty_sold FROM SALES WHERE PROD_ID
IN (SELECT PROD_ID FROM COMPUTER WHERE PROD_NAME LIKE '%Laptop%' OR
PROD_NAME LIKE '%Tablet%') GROUP BY QUARTER;
```

Q2. Consider the tables SHOPPE and ACCESSORIES given below:

Table : SHOPPE

Id	SName	Area
S001	ABC Computeronics	CP
S002	All Infotech Media	GK II
S003	Tech Shoppe	CP
S004	Geeks Tecno Soft	Nehru Place
S005	Hitech Tech Store	Nehru Place

Table : ACCESSORIES

No	Name	Price	Id
A01	MotherBoard	12000	S01
A02	Hard Disk	5000	S01
A03	Keyboard	500	S02
A04	Mouse	300	S01
A05	MotherBoard	13000	S02
A06	Keyboard	400	S03
A07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard Disk	4500	S03

Write SQL queries for the following:

- (i) Display shop name and Product name from the tables SHOPPE and ACCESSORIES.
- (ii) Display the schema of the table SHOPPE.
- (iii) Display the maximum and minimum price quoted for Motherboard.
- (iv) Display the Accessory name, price, and number of all the products which are available in Nehru Place.

Ans.

- (i) `SELECT SHOPPE.ShopName, ACCESSORIES.ProductName FROM SHOPPE, ACCESSORIES WHERE SHOPPE.ShopID = ACCESSORIES.ShopID;`
- (ii) `DESCRIBE SHOPPE;`
- (iii) `SELECT MAX(Price) AS MaxPrice, MIN(Price) AS MinPrice FROM ACCESSORIES WHERE ProductType = 'Motherboard';`
- (iv) `SELECT ProductName AS AccessoryName, Price FROM ACCESSORIES`

`WHERE ShopID IN (SELECT ShopID FROM SHOPPE WHERE Location = 'Nehru Place');`

Interface Python with SQL database

Database connectivity

Database connectivity refers to connection and communication between an application and a database system. The term “front-end” refers to the user interface, while “back-end” means the server, application and database that work behind the scenes to deliver information to the user.

MySQL.connector- Library or package to connect from python to MySQL.

Before we connect the program with mysql , we need to install connectivity package named mysql-connector-python

- Command to install connectivity package:- pip install mysql-connector-python
- Command to import connector:- import mysql.connector
- Steps for python MySQL connectivity
 1. Install Python
 2. Install MySQL
 3. Open Command prompt
 4. Switch on internet connection
 5. Type pip install mysql-connector-python and execute
 6. Open python IDLE
 7. import mysql.connector

Steps for creating database connectivity applications:

1. Start Python- start python editor to create our own python script.
2. Import mysql.connector package

```
import mysql.connector      or      import mysql.connector as Con
```

Establishing a connection to Mysql DATABASE.

We need to establish a connection to a mysql database using connect () function of mysql.connector package. The connect statement creates a connection to the mysql server and returns a MySQL connection object.

Syntax:

```
<Connection object> = mysql.connector.connect (host=<hostname>, user=<username>, passwd:<password>, database=<dbname>)
```

Example:

```
import mysql.connector con=mysql.connector.connect(host="localhost",user="root", passwd=""
```

Creating a cursor Object

It is a useful control structure of database connectivity. It will let us execute all the queries we need. Cursor stores all the data as a temporary container of returned data and allows traversal so that we can fetch data one row at a time from cursor. Cursors are created by the connection.cursor() method.

Syntax:

```
<cursor object>=<connectionobject>.cursor()
```

Example: cursor=con.cursor()

Execute SQL query:

We can execute SQL query using execute() function .

Syntax: <cursor object>.execute(SQL QUERY)

Example: cursor.execute("select * from data")

The above code will execute the sql query and store the retrieved records (resultset) in the cursor object(cursor).

Result set refers to a logical set of records that are fetched from the database by executing an sql query and made available in the program.

Extract data from Result set:

The records retrieved from the database using SQL select query has to be extracted as record from the result set. We can extract data from the result set using the following fetch () function.

1. fetchall()- Fetches all (remaining) rows of a query result, returning them as a sequence of sequences (e.g. a list of tuples) .
2. fetchone()-Fetches the next row of a query result set, returning a single sequence or None when no more data is available
3. fetchmany (size)-Fetches the next set of rows of a query result, returning a sequence of sequences. It will return number of rows that matches to the size argument.

TO CREATE DATABASE SCHOOL USING PYTHON INTERFACE

```
import mysql.connector mydb=mysql.connector.connect(host="localhost",user="root",passwd="system ")  
mycursor=mydb.cursor()  
mycursor.execute("CREATE DATABASE SCHOOL")
```

SHOW DATABASE

```
import mysql.connector
```

```
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system")
mycursor=mydb.cursor()
mycursor.execute("SHOW DATABASES")
for x in mycursor:
print (x)
```

TO CREATE A TABLE IN MYSQL USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system",database="student")
mycursor=mydb.cursor()
mycursor.execute("CREATE TABLE FEES (ROLLNO INT,NAME VARCHAR(20),AMOUNT INT);")
```

TO SHOW THE TABLES IN MYSQL USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system",database="student")
mycursor=mydb.cursor()
mycursor.execute("SHOW TABLES")
for x in mycursor:
print(x)
```

TO DESCRIBE TABLE STRUCTURE USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system",database="student")
mycursor=mydb.cursor()
mycursor.execute("DESC STUDENT")
for x in mycursor:
print(x)
```

TO EXECUTE SELECT QUERY USING A PYTHON INTERFACE

```
import mysql.connector
conn=mysql.connector.connect(host="localhost",user="root",passwd="12345",database="student")
```

```

c=conn.cursor()
c.execute("select * from student")
r=c.fetchone()
while r is not None:
print(r) r=c.fetchone()

```

TO EXECUTE SELECT QUERY WITH WHERE CLAUSE USING A PYTHON INTERFACE

```

import mysql.connector
conn=mysql.connector.connect(host="localhost",user="root",passwd="12345", database="student")
if conn.is_connected()==False:
print("Error connecting to MYSQL DATABASE")
c=conn.cursor()
c.execute("select * from student where marks>90")
r=c.fetchall()
count=c.rowcount
print("total no of rows:",count)
for row in r:
print(row)

```

TO INSERT A RECORD (ROLLNO, NAME AND MARKS) IN MYSQL TABLE student USING PYTHON INTERFACE

```

import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system ",database="student")
mycursor=mydb.cursor()

r=int(input("enter the rollno"))
n=input("enter name")
m=int(input("enter marks"))
mycursor.execute("INSERT INTO student(rollno,name,marks) VALUES( {}, '{}', {} )".format(r,n,m))
mydb.commit()
print(mycursor.rowcount," RECORD  INSERTED")

```

TO UPDATE A DATA IN A TABLE USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system",database="student")
mycursor=mydb.cursor() mycursor.execute("UPDATE STUDENT SET MARKS=100 WHERE MARKS=40")
mydb.commit() print(mycursor.rowcount,"RECORD UPDATED")
```

TO DELETE A RECORD FROM THE TABLE USING PYTHON INTERFACE

```
import mysql.connector mydb=mysql.connector.connect(host="localhost",user="root",passwd="system
",database="student")
mycursor=mydb.cursor()
mycursor.execute("DELETE FROM STUDENT WHERE MARKS<50") mydb.commit()
print(mycursor.rowcount,"RECORD DELETED")
```

TO DROP AN ENTIRE TABLE FROM MYSQL DATABASE USING PYTHON INTERFACE

```
import mysql.connector
```

```
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system ",database="student")
mycursor=mydb.cursor()
mycursor.execute("DROP TABLE STUDENT")
```

TO ADD A COLUMN IN THE EXISTING TABLE USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system ",database="student")
mycursor=mydb.cursor()
mycursor.execute("ALTER TABLE STUDENT ADD AGE INT")
mydb.commit()
```

TO DROP A COLUMN FROM THE TABLE USING PYTHON INTERFACE

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="system ",database="student")
mycursor=mydb.cursor()
mycursor.execute("ALTER TABLE DROP AGE ")
mydb.commit()
```

TO ALTER THE DATATYPE OF A COLUMN IN A TABLE USING PYTHON INTERFACE

```
import mysql.connector
mydb = mysql.connector.connect(host="localhost",user="root",passwd="syste m",database="student")
```

```
mycursor=mydb.cursor()
```

```
mycursor.execute("ALTER TABLE STUDENT MODIFY GRADE CHAR(3)")
```

TO DELETE A RECORD FROM TABLE USING PYTHON INTERFACE

```
import mysql.connector
```

```
demodb = mysql.connector.connect(host="localhost", user="root", passwd="computer",  
database="EDUCATION")
```

```
democursor=demodb.cursor()
```

```
democursor.execute("delete from student where admn_no=1356")
```

```
demodb.commit()
```

Multiple choice Questions(MCQ):

Q1. In order to open a connection with MySQL database from within Python using mysql.connector package, function is used.

- a) open() b) database() c) connect() d) connectdb()

Q2. A database controls the connection to an actual database, established from within a Python program.

- a) database object b) connection object c) fetch object d) query object

Q3. The set of records retrieved after executing an SQL query over an established database connection is called

- a) table b) sqlresult c) result d) resultset

Q4. A database is a special control structure that facilitates the row by row processing of records in the resultset.

- a) Fetch b) table c) cursor d) query

Q5. Which of the following is not a legal method for fetching records from database from within Python?

- a) fetchone() b) fetchtwo() c) fetchall() d) fetchmany()

Q6. To obtain all the records retrieved, you may use <cursor>. method.

- a) fetch() b) fetchmany() c) fetchall() d) fetchmultiple()

Q7. To fetch one record from the resultset, you may use <cursor>. method.

- a) fetch() b) fetchone() c) fetchtuple() d) none of these

Q8. To fetch multiple records from the resultset, you may use <cursor>. method.

- a) fetch() b) fetchmany() c) fetchmultiple() d) fetchmore()

Q9. To run an SQL query from within Python, you may use <cursor>. method().

- a) query()
- b) execute()
- c) run()
- d) all of these

Q10. To reflect the changes made in the database permanently, you need to run <connection>. method.

- a) done()
- b) reflect()
- c) commit()
- d) final()

Q11 to Q15 are ASSERTION AND REASONING based questions.

Mark the correct choice as

- a. Both A and R are true and R is the correct explanation for A
- b. Both A and R are true and R is not the correct explanation for A
- c. A is True but R is False
- d. A is false but R is True

Q11.Assertion. A database connection object controls the connection to a database.

Reason. A connection object represents a unique session with a database, connected from within a script/program.

Q12. Assertion. A database cursor receives all the records retrieved as per the query.

Reason. A resultset refers to the records in the database cursor and allows processing of individual records in it.

Q13. Assertion. The database cursor and resultset have the same data yet they are different.

Reason. The database cursor is a control structure and the resultset is a logical set of records.

Q14. Assertion. One by one the records can be fetched from the database directly through the database connection.

Reason. The database query results into a set of records known as the resultset.

Q15. Assertion. The cursor rowcount returns how many rows have been retrieved so far through fetch...() methods.

Reason. The number of rows in a resultset and the rowcount are always equal.

Ans.

1	C	2	B	3	D	4	C	5	B
6	C	7	B	8	B	9	B	10	C
11	A	12	A	13	A	14	D	15	C

Very Short Answer Questions

Q1. What is the package used for creating a Python database connectivity application.

Ans. mysql.connector is the package used for creating a Python database connectivity application.

Q2. Which function/method do you use for establishing connection to database ?

Ans. The connect() function of mysql.connector is used for establishing connection to a MYSQL database.

Q3. Which function/method do you use for executing an SQL query ?

Ans. The execute() function with cursor object is used for executing an SQL query.

Q4. Which method do you use to fetch records from the result set ?

Ans. The fetchall() method, fetchmany() method, or fetchone() method can be used to fetch records from the result set.

Long Answer Questions:

Q1. Write code to connect to a MySQL database namely **School** and then fetch all those records from table **Student** where grade is 'A' .

Ans. import mysql.connector as mysql

```
db_con = mysql.connect( host = "localhost", user = "root", password = "tiger", database = "School")
```

```
cursor = db_con.cursor()
```

```
cursor.execute("SELECT * FROM Student WHERE grade = 'A'")
```

```
student_records = cursor.fetchall()
```

```
for student in student_records:
```

```
    print(student)
```

```
db_con.close()
```

Scoring Tips:

Scoring well in the DBMS (Database Management System) involves understanding key concepts and practicing problem-solving. Here are some tips to help students maximize their scores:

1. Understand Core Concepts
2. Practice SQL Queries
3. Solve Past Papers and Sample Questions