TOUCHPAD
Computer Textbook Series

12

Computer Science with PYTHON

Beta*
*Updated Copy coming soon

ORANGE

Answer Key

Class XII

Computer Science with PYTHON

Unit I: Computational Thinking and Programming-2

2. Local and Global Scope

Assessment

A.	1. b	2. a	3. b	4. b	5. a
B.	1. True	2. False	3. True	4. True	5. True
C.	1. scope	2. local	3. global	4. error	5. accessible

D. 1. Part of the program where a named object is accessible is called its scope. The time period during which a variable is active (during execution of the program) is known as its lifetime.

2. Local Variable is a variable that is defined inside a function and isaccessible only within the body of a function (also called the scope of a function). Global variable is a variable defined outside the functions and can be accessed inside as well as outside of functions. A global variable can be modified inside any function by preceding it with the keyword global.

3. A global variable can be accessed and modified inside any function by using the keyword global. Example:

```
a = 90
def func():
 global a
 a += 10
 print('Locally in function func(), a =', a)
func()
print('In global frame, a =', a)
```

4. The scope of a named object (variable) refers to the part of a program in which it is accessible. Local variable is accessible only within the body of a function while Global variables can be accessed inside as well as outside of functions.

Example:

```
a = 10
```

```
def func():
    b = 20
    print('Locally in function func(), a =', a)
    print('Locally in function func(), b =', b)
func()
print('In global frame, a =', a)
print('In global frame, b =', b)
# Above statement will yield an error as the variable b is not visible
outside
# func()
```

In the above example, variable a refers to a global variable, however, b is a variable local to function `func()`, not visible outside the function.

5. An imported name can only be accessed within its scope. For example, let us examine the following Program:

```
def areaCircle(radius):
    """
    Objective: To compute the area of the circle
    Input Parameters:
    radius- numeric value denoting radius of circle
    Return value: area of circle
    """
    from math import pi
    area = pi * radius ** 2
    return area
radius = 5
print('radius of circle = ', radius, 'area of circle = ',
round(areaCircle(5),2) )
print('value of pi = ', pi)
```

In the above program, as pi has been imported within the function areaCircle(), it is not accessible in the global frame. Hence, we get an error when an attempt is made to print the value of pi in the global frame.

6.
```
local = 10        #Global Variable
global_1 = 20     #Global Variable
def myFunc(num):
    local = 3    #Local Variable
    local = local * 5
    global global_1       #Global Variable
    global_1 = global_1 + 100
    print(local, global_1, sep='@@')
myFunc(local)
print(local, global_1, sep='@@')
```

Output:

```
15@@120
10@@120
```

7.
```
count = 0
def countSpaces(str1):
    global count    # Statement added
    for d in str1:
        if d == ' ':
            count += 1
    return count
sentence1 = 'How are you?'
print('No of spaces in "' + sentence1 + '" =', countSpaces(sentence1))
sentence2 = ' I am good, Thank you'
print('No of spaces in "' + sentence1 + sentence2 + '" =',
countSpaces(sentence2))
```

Output after correcting program:

No of spaces in "How are you?" = 2

No of spaces in "How are you? I am good, Thank you" = 7

8. No of spaces in "How are you?" = 2

   No of spaces in "How are you? I am good, Thank you" = 7

9. [3, 9, -2, 0, 8, 6]

   [9, -2, 0, 8, 6, 3]

   [0, 8, 6, 3, 9, -2]

Assertion and Reasoning Based Questions

1. d   2. b

Case-based Questions

1.
```
maxMarks = 100
def grade(name, marks):
    '''
    Objective: To print a student's grade, based on his/her marks
    Input: name:string
           marks: marks obtained by the student
    Return Value: None
    '''

    global maxMarks
    print('Name:', name)
    print('Marks:', marks)
    percentage = (marks/maxMarks) * 100
    if percentage >= 90:
        print('Grade: A')
    elif percentage >= 60 and percentage <90:
        print('Grade: B')
    elif percentage >= 40 and percentage < 60:
        print('Grade: C')
    else :
        print('Grade: C')

name = input("Enter student's name: ")
marks = int(input("Enter student's marks: "))
grade(name, marks)
```

2.
```
discount = 10
def discountedPrice(category, price):
    '''
    Objective: To return discounted price
    Input: category
           price
    Return Value: None
    '''
    global discount
    if category == 'apparels':
        return 12,(price - 0.12*price)
    elif category == 'stationery':
        return 5,(price - 0.05*price)
    else:
        return discount,(price- (discount/100)*price)

name = input("Enter item name: ")
category = input("Enter category of item: ")
price = int(input("Enter price of item: "))
print('Item Name:', name)
print('Category:', category)
print('Sale Price:', price)
discount, updatedPrice = discountedPrice(category, price)
print('Discount', str(discount)+'%')
print('Discounted Sale Price', updatedPrice)
```

3. Handling Files in Python

Assessment

A.	1. c	2. c	3. b	4. c	5. d
	6. b	7. a	8. a	9. b	10. b
	11. b	12. a	13. d	14. a	15. c
	16. b	17. c			
B.	1. True	2. False	3. False	4. True	5. False
C.	1. text	2. binary	3. a (append mode)		4. r (read mode)
	5. text	6. text, csv	7. csv	8. close()	9. newline or \n
	10. csv, commas				

D. 1. • Function `readline()` is used to read line by line from a file whereas function `readlines()` returns the entire content of the file from the current position in the form of a list of lines

- Function `write()` is used to write the data to the file. The function returns the number of bytes written to the file. However, the function `writelines()` allows us to write a sequence of lines to a file in one go.
- Write mode (w) is used for writing to a file. If the file being opened already exists in the folder, opening the file in write mode will destroy the existing data of the file. Append mode (a) is used for writing the new data at the end of an already existing file. If the named file does not exist in the folder, a new file is created.
- The methods `load()` and `dump()` of the pickle module are used for reading and writing the objects from and to the binary files respectively. Syntax for using the `load()` and `dump()` methods of the pickle module are mentioned as follows:

  ```
  pickle.dump(sourceObject, fileObject)
  pickle.load(fileObject)
  ```
- The try-except blocks are used to handle the exceptions. A try block contains statements that may lead to an exception, the except block specifies what action should be performed if an exception is raised.
- To facilitate writing the data to CSV files, Python module csv provides the writer class that takes the file object as an input argument and returns the writer object `csv.writer` that is used for writing to a CSV file. Python module csv provides the reader class that takes file object as an input argument and returns a reader object `(csv.reader)` that is used for reading data from the CSV file.
- Pickling is the process of transforming a Python object into a byte stream for writing the object to a file. While reading a binary file, reverse process of unpickling is applied to obtain the data back in the object form.
- Function `seek()` is used to point the file handler to a particular position in the file while the function `tell()` returns the current position of the file handler .

2. 0

   2

3. (i)

```
Traceback (most recent call last):
  File "C:/Users/SheetalRajpal/AppData/Local/Programs/Python/
  Python312/prog1.py", line 1, in <module>
    stud = open('student.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'student.
txt'
```

(ii)

```
Meditation is the food to the soul!It takes care of the health of
mind.
```

(iii)

```
Meditation is the food to the soul!It takes care of the heal
```

(iv) `Meditation is the food to the soul!It takes care of the health of mind.`

`Meditation has been practiced for thousands of years to help`

4.
```
empDetails = [   ['E01', 'Aasmeen', 'Khan', '01-10-1989', 1000000,
'Admin'],
['E02', 'Pooja', 'Gupta', '13-02-1985', 2000000, 'Research'],
['E03', 'Rachit', 'Mittal', '01-01-1990', 900000, 'Admin'],
['E04', 'Ayushi', 'Sinha', '23-05-1985', 2500000, 'Research'],
['E05', 'Sumit', 'Talwar', '29-03-1988', 1400000, 'Admin'],
['E06', 'Rahul', 'Roy', '22-05-1986', 1200000, 'Admin'],
['E07', 'Diksha', 'Dewan', '12-12-1979', 3200000, 'Research']]
employee = open('employees.csv', 'w', newline='')
writeEmployee = csv.writer(employee)
writeEmployee.writerows(empDetails) #Use writerows to write multiple
rows
```

5.
```
EmpID
************************************************************
FName
************************************************************
LName
************************************************************
DateOfBirth
************************************************************
Salary
************************************************************
Department
************************************************************
```
D e t

6. The new content of the file will be the following:

```
Welcome to the class.
```

7.
```
def copy(file1, file2):
    '''
    Objective: To copy contents from file1 to file2 with a newline at
    the end of each line
```

```
     Input Parameters: file1, file2 - string values representing file
     names
     Return Value: None
     '''
     f1 = open(file1, 'r')
 f2 = open(file2, 'w')
 for line in f1:
         f2.write(line.rstrip() + '\n')
 f1.close()
 f2.close()
```

8.
```
def capitalizeInput():
     '''
     Objective: To take user input line by line, capitalize it, and
     write to a file named data.txt
     Input Parameters: None
     Return Value: None
     '''
     file = open('data.txt', 'w')
     while True:
             line = input("Enter a line (or press Enter to finish): ")
             if line == "":
                 break
             file.write(line.upper() + '\n')
```

9.
```
def copyOddLines(file1, file2):
     '''
    Objective: To copy alternate odd-numbered lines from file1 to file2
     Input Parameters: file1, file2 - string values representing file
     names
     Return Value: None
     '''
     f1 = open(file1, 'r')
     f2 = open(file2, 'w')

     lineNumber = 1
    while True:
        line = f1.readline()
        if line == '':
            break
        if lineNumber % 2 != 0:
            f2.write(line)
        lineNumber += 1

    f1.close()
    f2.close()
```
10.
```
def countWords():
    '''
    Objective: To count the occurrences of "my" and "rainbow" in Poem.
    txt
    Input Parameters: None
    Return Value: None
    '''
    file = open('Poem.txt', 'r')

    myCount = 0
    rainbowCount = 0

    while True:
        line = file.readline()
        if line == '':
            break
        words = line.split()
        for word in words:
            if word.lower() == 'my':
                myCount += 1
            elif word.lower() == 'rainbow':
                rainbowCount += 1

    file.close()
```

11.
```
def wordFrequency():
    '''
    Objective: To list the frequency count of every word and find the
    most frequently occurring word in swap
    Input Parameters: None
    Return Value: None
    '''
    file = open('swap', 'r')
    wordCounts = {}
    while True:
        line = file.readline()
        if line == '':
            break
        words = line.split()
        for word in words:
            word = word.lower()
            if word in wordCounts:
                wordCounts[word] += 1
            else:
                wordCounts[word] = 1
    file.close()
    mostCommonWord = None
    mostCommonCount = 0

    for word, count in wordCounts.items():
        if count > mostCommonCount:
            mostCommonWord = word
            mostCommonCount = count

    print("The most frequently occurring word is", mostCommonWord,
    "which occurs", mostCommonCount, "times")
```
12.
```
def inputStory():
    '''
    Objective: To take the contents of file Story.txt as input from
    the user
    Input Parameters: None
    Return Value: None
    '''
    file = open('Story.txt', 'w')

    while True:
        line = input("Enter a line (or press Enter to finish): ")
        if line == "":
            break
        file.write(line + '\n')
    file.close()

def displayLinesStartingWithA():
    '''
    Objective: To read contents from Story.txt and display lines starting
    with the article 'A'
    Input Parameters: None
    Return Value: None
    '''
    file = open('Story.txt', 'r')

    while True:
        line = file.readline()
        if line == '':
            break
        if line.startswith('A'):
            print(line, end='')
    file.close()
```

13.
```
def saveText():
    '''
    Objective: To take text as input from the user and store it in
    Story.txt
    Input Parameters: None
    Return Value: None
    '''
    file = open('Story.txt', 'w')

    while True:
        line = input("Enter a line (or press Enter to finish): ")
        if line == "":
            break
        file.write(line + '\n')
    file.close()
def descFile(filename):
    '''
    Objective: To read contents from a file and compute descriptive
    statistics
    Input Parameters: filename - string value representing the file
    name
    Return Value: A tuple containing the number of characters, words,
and sentences
    '''
    file = open(filename, 'r')
    text = file.read()
    file.close()
    numChars = len(text)
    numWords = len(text.split())
    numSentences = text.count('.') + text.count('!') + text.count('?')
    return numChars, numWords, numSentences
```

14.

```
def analyzeText(filename):
    '''
    Objective: To compute the number of uppercase and lowercase
    characters, and words starting with a vowel
    Input Parameters: filename - string value representing the file
    name
    Return Value: A list containing the number of uppercase characters,
    lowercase characters, and words starting with a vowel
```

```
        '''
        file = open(filename, 'r')
        text = file.read()
        file.close()

        numUppercase = 0
        numLowercase = 0
        numVowelWords = 0

        for char in text:
            if char.isupper():
                numUppercase += 1
            elif char.islower():
                numLowercase += 1

        words = text.split()
        for word in words:
            if word[0].lower() in 'aeiou':
                numVowelWords += 1

        return [numUppercase, numLowercase, numVowelWords]
```
15.
```
    import csv
    def calculateTotalPrice(file1, file2, outputFile):
        '''
        Objective: To compute total price for each item and write it to a
        third CSV file
        Input Parameters: file1, file2 - string values representing the
        file names
                          outputFile - string value representing the name
                          of the output file
        Return Value: None
        '''
        f1 = open(file1, 'r')
        f2 = open(file2, 'r')
    f3 = open(outputFile, 'w', newline='')

    file1Reader = csv.reader(f1)
    file2Reader = csv.reader(f2)
    outputWriter = csv.writer(f3)

    while True:
        itemRow = next(file1Reader)
        quantityRow = next(file2Reader)
        if itemRow and quantityRow:
            itemNumber = itemRow[0]
            pricePerUnit = float(itemRow[1])
            quantityPurchased = int(quantityRow[1])

            totalPrice = pricePerUnit * quantityPurchased
            outputWriter.writerow([itemNumber, totalPrice])
        else:
            break

    f1.close()
    f2.close()
    f3.close()
```

16.
```
import csv
def calculateTotalPriceWithWeight(file1, file2, outputFile):
    '''
    Objective: To compute total price for each item and price per unit
    weight, then write it to a third CSV file
    Input Parameters: file1, file2 - string values representing the
    file names
                      outputFile - string value representing the name
                      of the output file
    Return Value: None
    '''
    f1 = open(file1, 'r')
    f2 = open(file2, 'r')
    f3 = open(outputFile, 'w', newline='')

    file1Reader = csv.reader(f1)
    file2Reader = csv.reader(f2)
    outputWriter = csv.writer(f3)

    while True:
        itemRow = next(file1Reader)
        quantityRow = next(file2Reader)
        if itemRow and quantityRow:
            itemNumber = itemRow[0]
            pricePerUnit = float(itemRow[1])
            weight = float(itemRow[2])
            quantityPurchased = int(quantityRow[1])

            totalPrice = pricePerUnit * quantityPurchased
            pricePerUnitWeight = pricePerUnit / weight

           outputWriter.writerow([itemNumber, totalPrice,
           pricePerUnitWeight])
        else:
            break

    f1.close()
    f2.close()
    f3.close()
```

17.
```
import pickle
def writeStudentDictionaries(dictA, dictB):
    '''
    Objective: To write two student dictionaries to a file
    'classXIIstudents.dat'
    Input Parameters: dictA, dictB - dictionaries containing student
    details
```

```
        Return Value: None
        '''
        f = open('classXIIstudents.dat', 'wb')

        data = {
            "Class XII A": dictA,
            "Class XII B": dictB
        }

        pickle.dump(data, f)
        f.close()

    def readStudentDictionaries():
        '''
        Objective: To read student dictionaries from the file 'classXIIstudents.
        dat' and display student details
        Input Parameters: None
        Return Value: None
        '''
        f = open('classXIIstudents.dat', 'rb')
        data = pickle.load(f)
        f.close()
           for classKey in data:
             print(classKey)
             for rollNumber in data[classKey]:
                 name = data[classKey][rollNumber]
                 print("Roll Number:", rollNumber, "Name:", name)
```

18.
```
    import pickle
    def updateMarks(fileName, n):
        '''
        Objective: To update marks in dictionaries and save them to
        'classXIIstudents.dat'
        Input Parameters: fileName - string value representing the file
        name
```

```
                        n - number of dictionaries
    Return Value: None
    '''
    dictionaries = []
    for _ in range(n):
        dictionary = {}
        rollNum = int(input("Enter Roll Number: "))
        name = input("Enter Name: ")
        section = input("Enter Section: ")
        marks = int(input("Enter Marks: "))
        dictionary['RollNum'] = rollNum
        dictionary['Name'] = name
        dictionary['Section'] = section
        dictionary['Marks'] = min(marks + 5, 100)
        dictionaries.append(dictionary)

    f = open(fileName, 'wb')
    pickle.dump(dictionaries, f)
    f.close()

def readAndUpdateMarks(fileName):
    '''
    Objective: To read dictionaries from a file, update the marks, and
    save them
    Input Parameters: fileName - string value representing the file
    name
    Return Value: None
    '''
    f = open(fileName, 'rb')
    dictionaries = pickle.load(f)
    f.close()

    for dictionary in dictionaries:
            dictionary['Marks'] = min(dictionary['Marks'] + 5, 100)

        f = open(fileName, 'wb')
        pickle.dump(dictionaries, f)
        f.close()
```

19.
```
    import csv
    def splitEmailIds(inputFile, outputFile):
        '''
        Objective: To split email ids into username and domain_name and
        write to a new CSV file with headers
        Input Parameters: inputFile - string value representing the input
        file name
                          outputFile - string value representing the output
                          file name
        Return Value: None
        '''
        f1 = open(inputFile, 'r')
        f2 = open(outputFile, 'w', newline='')

        writer = csv.writer(f2)
        writer.writerow(['username', 'domainname'])  # Write the header

        line = f1.readline()
        while line:
            email = line.strip()
            username, domainName = email.split('@')
            writer.writerow([username, domainName])
            line = f1.readline()

        f1.close()
        f2.close()
```

20.
```
    import pickle
    def updateRecord(fileName):
```

```
'''
Objective: To moderate student marks based on specific criteria
and save the results
Input Parameters: fileName - string value representing the file
name
Return Value: None
'''
studFile = open(fileName, 'rb')
students = pickle.load(studFile)
f.close()

for student in students:
    marks = student['Marks']
    if marks <= 27:
        moderation = 7
    elif marks <= 40:
        moderation = 6
    elif marks <= 50:
        moderation = 5
    elif marks <= 60:
        moderation = 4
    elif marks <= 70:
        moderation = 3
    elif marks <= 80:
        moderation = 2
    elif marks <= 90:
        moderation = 1
    else:
        moderation = 0

    student['Marks'] = min(marks + moderation, 100)

f = open(fileName, 'wb')
```

```
        pickle.dump(students, f)
        f.close()
```

Assertion and Reasoning Based Questions

1. b 2. b 3. c 4. b

Case-based Questions

1. (i) c. csv
   (ii) b. "Student.csv","w"
   (iii) c. writer(fh)
   (iv) d. roll_no, name, Class, section
   (v) c. writerows()
2. (i) a. F= open("STUDENT.DAT",'wb')
   (ii) c. pickle.dump(L,F)
   (iii) a. R = pickle.load(F)
   (iv) a. 'r+' opens a file for both reading and writing. File object points to its beginning.
   (v) d. moves the current file position to a given specified position

4. Exception Handling

Assessment

A. 1. c 2. c 3. b 4. c 5. b 6. b
7. b 8. c 9. c 10. c

B. 1. True 2. False 3. True 4. True 5. True 6. True
7. False 8. True 9. True 10. True

C. 1. incorrect indentation 2. TypeError 3. ZeroDivisionError
4. IndexError 5. EOFError 6. else 7. finally

D. 1. Exceptions are errors or exceptional conditions that occur during the execution of a Python program, disrupting the normal flow of code. Exception handling is a feature of Python that allows developers to manage and respond to errors or exceptional situations gracefully.

2. Exception handling is a feature of Python that allows developers to manage and respond to errors or exceptional situations gracefully. Exception handling in Python is done using

the try...except block. The try block contains the code that may raise an exception, and the except block specifies how to handle specific exception types.

3. Built-in exceptions are commonly encountered exceptions that are pre-defined by the compiler or interpreter. Examples of built-in exceptions include NameError, TypeError, ValueError, ZeroDivisionError, IndexError, EOFError, FileNotFoundError, and PermissionError. To avoid the sudden termination of a program when an exception is raised, it is necessary to handle the exception by catching it and implementing the appropriate actions using the try...except clause.

4. **`NameError:`** This exception occurs whenever a name that appears in a statement is not found globally.

   `TypeError:` This exception occurs when an operation or function is applied to an object of inappropriate type.

   `ValueError:` This exception occurs whenever an inappropriate argument value, even though of the correct type, is used in a function call.

   `ZeroDivisionError:` This exception occurswhen we try to perform a numeric division in whichthe denominator happens to be zero.

   `IndexError:` This exception occurs whenever we try to access an index that is out of a valid range.

5. The try block consists of statements that can potentially raise an exception. These statements are enclosed within the try block because they are likely to raise an exception during execution. The except block defines the actions to be performed when an exception is raised within the try block.

   For example,

   ```
   try:
     file = open("example.txt", "r")
     # Perform operations on the file
     file.close()
   except FileNotFoundError:
             print("File missing in working directory")
   ```

   In the above code, when we try to open a file named example.txt in read mode, and the file doesn't exist in the working directory, a **`FileNotFoundError`** is still raised, but it would be managed smoothly. If the exception occurs, the program jumps to the except block that contains the handling mechanism for the exception **`FileNotFoundError`**. The user now gets a suitable message.

6. Clause **`finally`** is used to define a block of code that will always be executed, regardless of whether an exception occurred or not. The **`finally`** block is placed after all the except blocks (if any) and is optional.

7.
```
try:
    length = LEN('Enter your marks')
except NameError as e:
    print('Undefined Name:', e)
```
8.
```
try:
    marks = 99
    message = 'Marks Scored:' + marks
except TypeError as e:
    print('TypeError: ', e)
```
9.
```
try:
    num1 = int(input ("Enter the first number"))
    num2 = int(input("Enter the second number"))
    quotient = (num1 / num2)
    print ("Both the numbers entered were correct")
except ValueError: # to enter only integers
    print (" Please enter only numbers")
except ZeroDivisionError: # Denominator should not be zero
  print(" Number 2 should not be zero")
else:
  print(" Great")
```
10.
```
try:
    num1 = int(input ("Enter the first number"))
    num2 = int(input("Enter the second number"))
    quotient = (num1 / num2)
    print ("Both the numbers entered were correct")
    except ValueError: # to enter only integers
    print (" Please enter only numbers")
    except ZeroDivisionError: # Denominator should not be zero
    print(" Number 2 should not be zero")
else:
    print(" Great")
```

11.
```
import sys
try:
    # Open the source file in read mode
    source_file = open('source.txt', 'r')
    # Read the contents of the source file
    content = source_file.read()
    # Close the source file
    source_file.close()
try:
    # Open the destination file in write mode
    destination_file = open('destination.txt', 'w')
    # Write the contents to the destination file
    destination_file.write(content)
    # Close the destination file
    destination_file.close()
    print('File copied successfully.')
except PermissionError:
    print('Permission denied to write to the destination file.')
except Exception as e:
    print('An error occurred while copying the file:', str(e))
except FileNotFoundError:
    print('Source file not found.')
except Exception as e:
   print('An error occurred while accessing the source file:', str(e))
```

12.
```
try:
    num1 = int(input ("Enter the first number"))
    num2 = int(input("Enter the second number"))
    quotient = (num1 / num2)
    print ("Both the numbers entered were correct")
except ValueError: # to enter only integers
    print (" Please enter only numbers")
except ZeroDivisionError: # Denominator should not be zero
    print(" Number 2 should not be zero")
else:
    print(" Great")
```

Assertion and Reasoning Based Questions

1. a   2. b   3. a

5. Data Structures: Stacks

Assessment

A.	1. b	2. a	3. c	4. c	5. b
B.	1. False	2. False	3. False	4. True	5. True
C.	1. data structure		2. push	3. pop	4. underflow

D. 1. The phrase data structures comprises two words, data and structures. The term data refers to facts and statistics relating to something of interest. Similarly, the word structure refers to a formation, an organisation, an arrangement, a collection, an entity, etc. Different organisations of data enable different operations on it. Thus, a data structure is an arrangement of data along with the associated operations.

2. Stack is a Last In First Out (LIFO) arrangement. Push operation inserts an element on the top of the stack. Pop operation removes the top element from the stack.

3. An attempt to pop an element from the empty stack results in a stack underflow condition.

4. Given a stack, we may perform the following operations:

- **Push:** Push operation places an element on the top of the stack. Sometimes, push operation is also called the insertion of an element on top of a stack.
- **Pop:** Pop operation removes the top element from the stack. Pop operation is also called the deletion of the top element of the stack.

5. (i) (a b c d + * *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (d + c)	(2 + 3)	[12, 6, 5]
Perform (b * 5)	(6 * 5)	[12, 30]
Perform (a * 30)	(12 * 30)	[360]

```
Result: 360
```

(ii) (a b c d * + *)

Step	Operation	Stack
Initial		

Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (d * c)	(2 * 3)	[12, 6, 6]
Perform (b + 6)	(6 + 6)	[12, 12]
Perform (a * 12)	(12 * 12)	[144]

Result: 144

(iii) (a b c d * * +)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (d * c)	(2 * 3)	[12, 6, 6]
Perform (b * 6)	(6 * 6)	[12, 36]
Perform (a + 36)	(12 + 36)	[48]

Result: 48

(iv) (a b c d + + *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (d + c)	(2 + 3)	[12, 6, 5]
Perform (b + 5)	(6 + 5)	[12, 11]
Perform (a * 11)	(12 * 11)	[132]

Result: 132

(v) (a b c d + * +)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]

Perform (d + c)	(2 + 3)	[12, 6, 5]
Perform (b * 5)	(6 * 5)	[12, 30]
Perform (a + 30)	(12 + 30)	[42]

Result: 42

(vi) (a b c d * + +)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (d * c)	(2 * 3)	[12, 6, 6]
Perform (b + 6)	(6 + 6)	[12, 12]
Perform (a + 12)	(12 + 12)	[24]

Result: 24

(vii) (a b + c * d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a + b)	(12 + 6)	[18]
Push c		[18, 3]
Perform (18 * c)	(18 * 3)	[54]
Push d		[54, 2]
Perform (54 * d)	(54 * 2)	[108]

Result: 108

(viii) (a b c + * d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Perform (b + c)	(6 + 3)	[12, 9]
Perform (a * 9)	(12 * 9)	[108]
Push d		[108, 2]
Perform (108 * d)	(108 * 2)	[216]

Result: 216

(ix) (a b - c * d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a - b)	(12 - 6)	[6]
Push c		[6, 3]
Perform (6 * c)	(6 * 3)	[18]
Push d		[18, 2]
Perform (18 * d)	(18 * 2)	[36]

Result: 36

(x) (a b c - * d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Perform (b - c)	(6 - 3)	[12, 3]
Perform (a * 3)	(12 * 3)	[36]
Push d		[36, 2]
Perform (36 * d)	(36 * 2)	[72]

Result: 72

(xi) (a b - c d * /)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a - b)	(12 - 6)	[6]
Push c		[6, 3]
Push d		[6, 3, 2]
Perform (c * d)	(3 * 2)	[6, 6]
Perform (6 / 6)	(6 / 6)	[1]

Result: 1

(xii) (a b c d / + +)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Perform (c / d)	(3 / 2)	[12, 6, 1]
Perform (b + 1)	(6 + 1)	[12, 7]
Perform (a + 7)	(12 + 7)	[19]

```
Result: 19
```

(xiii) (a b + c * d /)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a + b)	(12 + 6)	[18]
Push c		[18, 3]
Perform (18 * c)	(18 * 3)	[54]
Push d		[54, 2]
Perform (54 / d)	(54 / 2)	[27]

```
Result: 27
```

(xiv) (a b + c / d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a + b)	(12 + 6)	[18]
Push c		[18, 3]
Perform (18 / c)	(18 / 3)	[6]
Push d		[6, 2]
Perform (6 * d)	(6 * 2)	[12]

```
Result: 12
```

(xv) (a b c / * d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Perform (b / c)	(6 / 3)	[12, 2]
Perform (a * 2)	(12 * 2)	[24]
Push d		[24, 2]
Perform (24 * d)	(24 * 2)	[48]

Result: 48

(xvi) (a b + c d / *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a + b)	(12 + 6)	[18]
Push c		[18, 3]
Push d		[18, 3, 2]
Perform (c / d)	(3 / 2)	[18, 1]
Perform (18 * 1)	(18 * 1)	[18]

Result: 18

(xvii) (a b - c / d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a - b)	(12 - 6)	[6]
Push c		[6, 3]
Perform (6 / c)	(6 / 3)	[2]
Push d		[2, 2]
Perform (2 * d)	(2 * 2)	[4]

Result: 4

(xviii) (a b - c * d /)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a - b)	(12 - 6)	[6]
Push c		[6, 3]
Perform (6 * c)	(6 * 3)	[18]
Push d		[18, 2]
Perform (18 / d)	(18 / 2)	[9]

Result: 9

(xix) (a b c - / d *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Perform (b - c)	(6 - 3)	[12, 3]
Perform (a / 3)	(12 / 3)	[4]
Push d		[4, 2]
Perform (4 * d)	(4 * 2)	[8]

Result: 8

(xx) (a b - c d / *)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Perform (a - b)	(12 - 6)	[6]
Push c		[6, 3]
Push d		[6, 3, 2]
Perform (c / d)	(3 / 2)	[6, 1]
Perform (6 * 1)	(6 * 1)	[6]

Result: 6

(xxi) (a b c d e f + * + + +)

Step	Operation	Stack
Initial		
Push a		[12]
Push b		[12, 6]
Push c		[12, 6, 3]
Push d		[12, 6, 3, 2]
Push e		[12, 6, 3, 2, 6]
Push f		[12, 6, 3, 2, 6, 9]
Perform (f + e)	(9 + 6)	[12, 6, 3, 2, 15]
Perform (d * 15)	(2 * 15)	[12, 6, 3, 30]
Perform (c + 30)	(3 + 30)	[12, 6, 33]
Perform (b + 33)	(6 + 33)	[12, 39]
Perform (a + 39)	(12 + 39)	[51]

```
Result: 51
```

6. (i) (a - b - c - d)

Step	Operation	Stack	Output
1	Read a		a
2	Read -	[-]	a
3	Read b	[-]	a b
4	Read -	[-, -]	a b -
5	Read c	[-]	a b - c
6	Read -	[-, -]	a b - c-
7	Read d	[-]	a b - c - d
8	End		a b - c - d -

```
Postfix: a b - c - d -
```

(ii) ((a - b) - (c - d))

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)		a b -
6	Read -	[-]	a b -
7	Read (	[-, (]	a b -
8	Read c	[-, (]	a b - c
9	Read -	[-, (, -]	a b - c
10	Read d	[-, (, -]	a b - c d
11	Read)	[-]	a b - c d -
12	End		a b - c d - -

```
Postfix: a b - c d - -
```

(iii) ((a - b) - c - d)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)		a b -
6	Read -	[-]	a b -
7	Read c	[-]	a b - c
8	Read -	[-, -]	a b - c -
9	Read d	[-]	a b - c - d
10	End		a b - c - d -

`Postfix: a b - c - d -`

(iv) (a / b + c - d)

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read +	[+]	a b /
5	Read c	[+]	a b / c
6	Read -	[-]	a b / c +
7	Read d	[-]	a b / c + d
8	End		a b / c + d -

`Postfix: a b / c + d -**`

(v) (a / b / c / d)

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read /	[/]	a b /
5	Read c	[/]	a b / c
6	Read /	[/]	a b / c /
7	Read d	[/]	a b / c / d
8	End		a b / c / d /

`Postfix: a b / c / d /`

(vi) ((a / b) / (c / d))

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a

3	Read /	[(, /]	a
4	Read b	[(, /]	a b
5	Read)		a b /
6	Read /	[/]	a b /
7	Read (	[/, (]	a b /
8	Read c	[/, (]	a b / c
9	Read /	[/, (, /]	a b / c
10	Read d	[/, (, /]	a b / c d
11	Read)	[/]	a b / c d /
12	End		a b / c d / /

```
Postfix: a b / c d / /
```

(vii) ((a / b) / c / d)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read /	[(, /]	a
4	Read b	[(, /]	a b
5	Read)		a b /
6	Read /	[/]	a b /
7	Read c	[/]	a b / c
8	Read /	[/]	a b / c /
9	Read d	[/]	a b / c / d
10	End		a b / c / d /

```
Postfix: a b / c / d /
```

(viii) (a / (b / c) / d)

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read (	[/, (]	a
4	Read b	[/, (]	a b
5	Read /	[/, (, /]	a b
6	Read c	[/, (, /]	a b c
7	Read)	[/]	a b c /
8	Read /	[/]	a b c /
9	Read d	[/]	a b c / d
10	End		a b c / d /

```
Postfix: a b c / d /
```

(ix) (a / ((b / c) / d))

Step	Operation	Stack	Output
1	Read a		a

2	Read /	[/]	a
3	Read (	[/, (]	a
4	Read (	[/, (, (]	a
5	Read b	[/, (, (]	a b
6	Read /	[/, (, (, /]	a b
7	Read c	[/, (, (, /]	a b c
8	Read)	[/, (, /]	a b c /
9	Read /	[/, (, /]	a b c /
10	Read d	[/, (, /]	a b c / d
11	Read)	[/]	a b c / d /
12	Read)	[/]	a b c / d /
13	End		a b c / d / /

Postfix: a b c / d / /

(x) (a / b / (c / d))

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read /	[/]	a b /
5	Read (	[/, (]	a b /
6	Read c	[/, (]	a b / c
7	Read /	[/, (, /]	a b / c
8	Read d	[/, (, /]	a b / c d
9	Read)	[/]	a b / c d /
10	Read /	[/]	a b / c d /
11	End		a b / c d / /

Postfix: a b / c d / /

(xi) (a - b * c / d)

Step	Operation	Stack	Output
1	Read a		a
2	Read -	[-]	a
3	Read b	[-]	a b
4	Read *	[-, *]	a b
5	Read c	[-, *]	a b c
6	Read /	[-, /]	a b c *
7	Read d	[-, /]	a b c * d
8	End		a b c * d / -

Postfix: a b c * d / -

(xii) (a / b - c / d)

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read -	[-]	a b /
5	Read c	[-]	a b / c
6	Read /	[-, /]	a b / c
7	Read d	[-, /]	a b / c d
8	End		a b / c d / -

```
Postfix: a b / c d / -
```

(xiii) (a / b - (c / d))

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read -	[-]	a b /
5	Read (	[-, (]	a b /
6	Read c	[-, (]	a b / c
7	Read /	[-, (, /]	a b / c
8	Read d	[-, (, /]	a b / c d
9	Read)	[-]	a b / c d /
10	End		a b / c d / -

```
Postfix: a b / c d / -
```

(xiv) ((a - b) / (c / d))

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)		a b -
6	Read /	[/]	a b -
7	Read (	[/, (]	a b -
8	Read c	[/, (]	a b - c
9	Read /	[/, (, /]	a b - c
10	Read d	[/, (, /]	a b - c d
11	Read)	[/]	a b - c d /
12	End		a b - c d / /

```
Postfix: a b - c d / /
```

(xv) ((a - b) * c / d)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)		a b -
6	Read *	[*]	a b -
7	Read c	[*]	a b - c
8	Read /	[/]	a b - c *
9	Read d	[/]	a b - c * d
10	End		a b - c * d /

`Postfix: a b - c * d /`

(xvi) (a / (b - c) / d)

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read (	[/, (]	a
4	Read b	[/, (]	a b
5	Read -	[/, (, -]	a b
6	Read c	[/, (, -]	a b c
7	Read)	[/]	a b c -
8	Read /	[/]	a b c - /
9	Read d	[/]	a b c - / d
10	End		a b c - / d /

`Postfix: a b c - / d /`

(xvii) (a / (b + c / d))

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read (	[/, (]	a
4	Read b	[/, (]	a b
5	Read +	[/, (, +]	a b
6	Read c	[/, (, +]	a b c
7	Read /	[/, (, +, /]	a b c
8	Read d	[/, (, +, /]	a b c d
9	Read)	[/]	a b c d / +
10	End		a b c d / + /

`Postfix: a b c d / + /`

(xviii) (a / b / (c - d))

Step	Operation	Stack	Output
1	Read a		a
2	Read /	[/]	a
3	Read b	[/]	a b
4	Read /	[/]	a b /
5	Read (	[/, (]	a b /
6	Read c	[/, (]	a b / c
7	Read -	[/, (, -]	a b / c
8	Read d	[/, (, -]	a b / c d
9	Read)	[/]	a b / c d -
10	End		a b / c d - /

Postfix: a b / c d - /

(xix) (a + b / (c - d))

Step	Operation	Stack	Output
1	Read a		a
2	Read +	[+]	a
3	Read b	[+]	a b
4	Read /	[+, /]	a b
5	Read (	[+, /, (]	a b
6	Read c	[+, /, (]	a b c
7	Read -	[+, /, (, -]	a b c
8	Read d	[+, /, (, -]	a b c d
9	Read)	[+, /]	a b c d -
10	End		a b c d - / +

Postfix: a b c d - / +

(xx) ((a - b) * (c - d) * e)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)	[*]	a b -
6	Read *	[*]	a b -
7	Read (	[*, (]	a b -
8	Read c	[*, (]	a b - c
9	Read -	[*, (, -]	a b - c
10	Read d	[*, (, -]	a b - c d
11	Read)	[*]	a b - c d -
12	Read *	[*]	a b - c d - *

13	Read e	[*]	a b - c d - * e
14	End		a b - c d - * e *

Postfix: a b - c d - * e *

(xxi) ((a - b) * c + (d + e) * f)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)	[*]	a b -
6	Read *	[*]	a b -
7	Read c	[*]	a b - c
8	Read +	[+]	a b - c *
9	Read (	[+, (]	a b - c *
10	Read d	[+, (]	a b - c * d
11	Read +	[+, (, +]	a b - c * d
12	Read e	[+, (, +]	a b - c * d e
13	Read)	[+]	a b - c * d e +
14	Read *	[+, *]	a b - c * d e +
15	Read f	[+, *]	a b - c * d e + f
16	End		a b - c * d e + f * +

Postfix: a b - c * d e + f * +

(xxii) ((a - b) * c * d + e)

Step	Operation	Stack	Output
1	Read (	[(]	
2	Read a	[(]	a
3	Read -	[(, -]	a
4	Read b	[(, -]	a b
5	Read)	[*]	a b -
6	Read *	[*]	a b -
7	Read c	[*]	a b - c
8	Read *	[*]	a b - c *
9	Read d	[*]	a b - c * d
10	Read +	[+]	a b - c * d *
11	Read e	[+]	a b - c * d * e
12	End		a b - c * d * e +

Postfix: a b - c * d * e +

(xxiii) (a * (b + (c + d) * e))

Step	Operation	Stack	Output
1	Read a		a

2	Read *	[*]	a
3	Read (	[*, (]	a
4	Read b	[*, (]	a b
5	Read +	[*, (, +]	a b
6	Read (	[*, (, +, (]	a b
7	Read c	[*, (, +, (]	a b c
8	Read +	[*, (, +, (, +]	a b c
9	Read d	[*, (, +, (, +]	a b c d
10	Read)	[*, (, +]	a b c d +
11	Read *	[*, (, +, *]	a b c d +
12	Read e	[*, (, +, *]	a b c d + e
13	Read)	[*, (]	a b c d + e *
14	Read)	[*]	a b c d + e * +
15	End		a b c d + e * + *

```
Postfix: a b c d + e * + *
```

7. Sequence (i): 1 2 3

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 1 exits	[]	1
3	Train 2 arrives	[2]	1
4	Train 2 exits	[]	1 2
5	Train 3 arrives	[3]	1 2
6	Train 3 exits	[]	1 2 3

```
Sequence (i) is possible.
```

Sequence (ii): 1 3 2

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 1 exits	[]	1
3	Train 2 arrives	[2]	1
4	Train 3 arrives	[2, 3]	1
5	Train 3 exits	[2]	1 3
6	Train 2 exits	[]	1 3 2

```
Sequence (ii) is possible.
```

Sequence (iii): 2 1 3

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 2 arrives	[1, 2]	
3	Train 2 exits	[1]	2

4	Train 1 exits	[]	2 1
5	Train 3 arrives	[3]	2 1
6	Train 3 exits	[]	2 1 3

Sequence (iii) is not possible. (Train 2 cannot exit before Train 1 without additional tracks)

Sequence (iv): 2 3 1

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 2 arrives	[1, 2]	
3	Train 3 arrives	[1, 2, 3]	
4	Train 3 exits	[1, 2]	3
5	Train 2 exits	[1]	3 2
6	Train 1 exits	[]	3 2 1

Sequence (iv) is not possible. (Train 2 cannot exit before Train 1 without additional tracks)

Sequence (v): 3 1 2

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 2 arrives	[1, 2]	
3	Train 3 arrives	[1, 2, 3]	
4	Train 3 exits	[1, 2]	3
5	Train 1 exits	[2]	3 1
6	Train 2 exits	[]	3 1 2

Sequence (v) is not possible. (Train 3 cannot exit before Train 1 and Train 2 without additional tracks)

Sequence (vi): 3 2 1

Step	Action	Stack	Output
1	Train 1 arrives	[1]	
2	Train 2 arrives	[1, 2]	
3	Train 3 arrives	[1, 2, 3]	
4	Train 3 exits	[1, 2]	3
5	Train 2 exits	[1]	3 2
6	Train 1 exits	[]	3 2 1

Sequence (vi) is possible.

8. A similar approach as used in previous question can be used to attempt this question.

9.
```
def PUSH(stack, stackEvens ):
     '''
      Objective: To push the even number at the top of the stack,
      Input Parameter:
          stack - list of even multiples of 5
          stackEvens - list
      Return Value: None
    '''
    for number in N:
          if number%5==0 and number%2==0:
                stack.append(num)
    print(stackEvens)
    print(len(stackEvens))
```

10.
```
def pushToStack(charStack, character):
    '''
    Objective: To push a character onto the stack.
    Input Parameter:
        charStack - list acting as the stack
        character - character to push onto the stack
    Return Value: None
    '''
    charStack.append(character)


def printReversedStack(charStack):
    '''
    Objective: To print characters from the stack in reverse order,
    each character printed twice.
    Input Parameter:
        charStack - list acting as the stack
    Return Value: None
    '''
    while charStack:
        character = charStack.pop()
```

```
            print(character * 2, end='')

def main():
    '''
    Objective: To accept characters from the user and print them in
    reverse order with each character printed twice.
    Input Parameter: None
    Return Value: None
    '''
    charStack = []
    userInput = input("Enter up to 30 characters: ")

    for char in userInput[:30]:
        pushToStack(charStack, char)

    printReversedStack(charStack)
    print()  # For a new line after the output

# Run the main function
main()
```

11.
```
def pushToStack(stack, number):
    '''
    Objective: To push a number onto the stack.
    Input Parameter:
        stack - list acting as the stack
        number - number to push onto the stack
    Return Value: None
    '''
    stack.append(number)

def findMaxInStack(stack):
    '''
    Objective: To find the maximum number in the stack.
    Input Parameter:
        stack - list acting as the stack
    Return Value:
        maxValue - maximum value in the stack
    '''
    maxValue = None
    if stack:
        maxValue = stack[0]
        for number in stack:
            if number > maxValue:
                maxValue = number
    return maxValue

def main():
    '''
    Objective: To accept 10 numbers from the user, store even numbers
    in a stack, and display the largest element from the stack.
    Input Parameter: None
    Return Value: None
    '''
    numList = []
    stack = []

    print("Enter 10 numbers:")
    for _ in range(10):
        num = int(input())
        numList.append(num)

    for num in numList:
        if num % 2 == 0:
            pushToStack(stack, num)

    if stack:
        maxValue = findMaxInStack(stack)
        print("The maximum value of stack is", maxValue)
    else:
        print("The stack is empty, no even numbers were entered.")

# Run the main function
main()
```

12.
```
def push(stack, travelId, travelDate, destination):
    '''
    Objective: To push a travel detail onto the stack.
    Input Parameters:
        stack - list acting as the stack
        travelId - integer representing the travel ID
        travelDate - string representing the travel date
        destination - string representing the travel destination
    Return Value: None
    '''
    travelDetails = [travelId, travelDate, destination]
    stack.append(travelDetails)
    print("Travel details pushed to stack.")


def pop(stack):
    '''
    Objective: To pop the top travel detail from the stack.
    Input Parameters:
        stack - list acting as the stack
    Return Value:
        travelDetails - list containing travel details popped from the
        stack
    '''
    if not stack:
        print("Stack is empty. Cannot pop.")
        return None
        travelDetails = stack.pop()
        print("Travel details popped from stack.")
        return travelDetails

    def display(stack):
        '''
        Objective: To display all travel details in the stack.
        Input Parameters:
            stack - list acting as the stack
        Return Value: None
        '''
        if not stack:
            print("Stack is empty.")
            return
        print("Travel details in stack:")
        for details in reversed(stack):
            print("Travel ID:", details[0], "Travel Date:", details[1],
            "Destination:", details[2])

    def main():
        '''
       Objective: To provide a menu-driven interface for stack operations.
        Input Parameters: None
        Return Value: None
        '''
        stack = []

        while True:
            print("\nMenu:")
            print("1. PUSH")
            print("2. POP")
            print("3. DISPLAY")
            print("4. EXIT")
        choice = int(input("Enter your choice (1-4): "))

        if choice == 1:
            travelId = int(input("Enter Travel ID: "))
            travelDate = input("Enter Travel Date (YYYY-MM-DD): ")
            destination = input("Enter Destination: ")
            push(stack, travelId, travelDate, destination)
        elif choice == 2:
            poppedDetails = pop(stack)
            if poppedDetails:
                print("Popped Details - Travel ID:", poppedDetails[0],
                "Travel  Date:",  poppedDetails[1],  "Destination:",
                poppedDetails[2])
        elif choice == 3:
            display(stack)
        elif choice == 4:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and
            4.")

# Run the main function
if __name__ == "__main__":
    main()
```

13.
```
def push(stack, num):
    '''
    Objective: To push the prime number at the top of the stack
    Input Parameter:
        stack - list of prime numbers
        num - number to be pushed
    Return Value: None
    '''
```

```
        stack.append(num)

def pop(stack):
    '''
    Objective: To pop an element from the stack
    Input Parameter:
        stack - list of prime numbers
    Return Value: Top element of the stack
    '''
    if stack != []:
        return stack.pop()
    else:
        return None

def isPrime(num):
    '''
    Objective: To check if a number is prime
    Input Parameter:
        num - number to be checked
    Return Value: True if num is prime, else False
    '''
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# Main program
stackPrimes = []
numList = []  # List to store the 10 numbers
```

```
# Accept 10 numbers from the user
for i in range(10):
    num = int(input("Enter number: "))
    numList.append(num)

# Push only prime numbers onto the stack
for number in numList:
    if isPrime(number):
        push(stackPrimes, number)

print("Stack with prime numbers:", stackPrimes)
```

14.
```
def push(stack, char):
    '''
    Objective: To push a character onto the stack
    Input Parameter:
        stack - list of characters
        char - character to be pushed
    Return Value: None
    '''
    stack.append(char)

def pop(stack):
    '''
    Objective: To pop a character from the stack
    Input Parameter:
        stack - list of characters
    Return Value: Top character of the stack
    '''
    if stack != []:
        return stack.pop()
    else:
        return None
```

```
def isPalindrome(str1):
    '''
     Objective: To check if the given string is a palindrome using a
stack
    Input Parameter:
        str1 - string to be checked
    Return Value: True if str1 is a palindrome, else False
    '''
    stack = []
    length = len(str1)

    # The middle character should be 'b'
    if str1[length // 2] != 'b':
        return False

    # Push first half of the string onto the stack
    for i in range(length // 2):
        push(stack, str1[i])

    # Compare second half of the string with the stack
    for i in range(length // 2 + 1, length):
        if str1[i] != pop(stack):
            return False

    return True

# Main program
str1 = input("Enter the string: ")

if isPalindrome(str1):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

15.
```
def push(stack, char):
    '''
    Objective: To push a character onto the stack
    Input Parameter:
        stack - list of characters
        char - character to be pushed
    Return Value: None
    '''
    stack.append(char)

def pop(stack):
    '''
    Objective: To pop a character from the stack
    Input Parameter:
        stack - list of characters
    Return Value: Top character of the stack
          '''
    if stack:
        return stack.pop()
    else:
        return None

def isBalanced(expression):
    '''
   Objective: To check if the given expression has balanced parentheses
    Input Parameter:
        expression - string containing the expression
    Return Value: True if parentheses are balanced, else False
    '''
    stack = []
    for char in expression:
        if char in '([{':
            push(stack, char)
        elif char in ')]}':
            if not stack:
                return False
            topChar = pop(stack)
            if not matches(topChar, char):
                return False
    return not stack

def matches(opening, closing):
    '''
    Objective: To check if the opening and closing parentheses match
    Input Parameter:
        opening - opening parenthesis character
        closing - closing parenthesis character
    Return Value: True if they match, else False
    '''
    opens = '([{'
    closes = ')]}'
    return opens.index(opening) == closes.index(closing)

# Main program
expression = input("Enter the expression: ")
if isBalanced(expression):
    print("The expression has balanced parentheses.")
else:
    print("The expression does not have balanced parentheses.")
```

16.
```
def push(stack, url):
    '''
    Objective: To push a URL onto the stack
    Input Parameter:
        stack - list of URLs
        url - URL to be pushed
    Return Value: None
    '''
    stack.append(url)

def pop(stack):
    '''
    Objective: To pop a URL from the stack
    Input Parameter:
        stack - list of URLs
    Return Value: Top URL of the stack
    '''
    if stack:
        return stack.pop()
    else:
        return None

def displayStack(stack):
    '''
    Objective: To display the URLs in the stack
    Input Parameter:
        stack - list of URLs
    Return Value: None
    '''
    for url in reversed(stack):
        print(url)

# Main program
backStack = []
forwardStack = []

while True:
    print("\nMenu Options:")
    print("1. BACK")
    print("2. FORWARD")
print("3. DISPLAY BACK PAGES")
print("4. DISPLAY FORWARD PAGES")
print("5. EXIT")

choice = int(input("Enter your choice: "))

if choice == 1:
    if backStack:
        url = pop(backStack)
        push(forwardStack, url)
        print(f"Navigated back to: {url}")
    else:
        print("No pages in back stack.")

elif choice == 2:
    if forwardStack:
        url = pop(forwardStack)
        push(backStack, url)
        print(f"Navigated forward to: {url}")
    else:
        print("No pages in forward stack.")

elif choice == 3:
    print("Back Pages:")
    if backStack:
        displayStack(backStack)
    else:
        print("No pages in back stack.")

elif choice == 4:
    print("Forward Pages:")
    if forwardStack:
        displayStack(forwardStack)
```

```
            else:
                print("No pages in forward stack.")

        elif choice == 5:
            print("Exiting the program.")
            break

        else:
            print("Invalid choice. Please try again.")
```

17.
```
def precedence(op):
    '''
    Objective: To return the precedence of the given operator
    Input Parameter:
        op - operator character
    Return Value:
        precedence value as integer
    '''
    if op == '+' or op == '-':
        return 1
    if op == '*' or op == '/':
        return 2
    return 0

def applyOp(op, a, b):
    '''
    Objective: To apply the operator on two operands
    Input Parameter:
        op - operator character
        a - first operand
        b - second operand
    Return Value:
        result of the operation
    '''
    if op == '+':
        return a + b
    if op == '-':
        return a - b
    if op == '*':
        return a * b
    if op == '/':
        return a / b

def evaluate(expression):
    '''
    Objective: To evaluate the infix expression using two stacks
    Input Parameter:
        expression - infix expression string
    Return Value:
        result of the expression
    '''
    operands = []
    operators = []

    i = 0
    while i < len(expression):
        if expression[i] == ' ':
            i += 1
            continue
        elif expression[i] == '(':
            operators.append(expression[i])
        elif expression[i].isdigit():
            val = 0
            while (i < len(expression) and expression[i].isdigit()):
                val = (val * 10) + int(expression[i])
                i += 1
            operands.append(val)
            i -= 1
        elif expression[i] == ')':
            while len(operators) != 0 and operators[-1] != '(':
                op = operators.pop()
                val2 = operands.pop()
                val1 = operands.pop()
                operands.append(applyOp(op, val1, val2))
            operators.pop()
        else:
            while (len(operators) != 0 and precedence(operators[-1])
            >= precedence(expression[i])):
                op = operators.pop()
                val2 = operands.pop()
                val1 = operands.pop()
                operands.append(applyOp(op, val1, val2))
            operators.append(expression[i])
        i += 1

    while len(operators) != 0:
        op = operators.pop()
        val2 = operands.pop()
        val1 = operands.pop()
        operands.append(applyOp(op, val1, val2))
    return operands[-1]

# Main program
expression = input("Enter an infix expression: ")
result = evaluate(expression)
print(f"The result of the expression is: {result}")
```

Assertion and Reasoning Based Questions

1. c 2. c

Case-based Questions

1.
```
def pushOn(bookStack, bookName):
    '''
    Objective: To push a book name onto the stack
    Input Parameters:
        bookStack - list of book names
        bookName - name of the book to be added
    Return Value: None
    '''
    bookStack.append(bookName)

def popFrom(bookStack):
    '''
    Objective: To pop a book name from the stack
    Input Parameters:
        bookStack - list of book names
    Return Value:
       name of the book that was removed or None if the stack is empty
    '''
    if len(bookStack) == 0:
        print("The book stack is empty. Cannot pop.")
        return None
    else:
        return bookStack.pop()
```

2.
```
def push15(studentDict, ageStack):
    '''
    Objective: To push names of students onto the stack whose age is
    greater than 15
    Input Parameters:
        studentDict - dictionary containing student names as keys and
        their ages as values
        ageStack - list used as a stack to store names of students with
        age greater than 15
    Return Value: None
    '''
    for name, age in studentDict.items():
        if age > 15:
            ageStack.append(name)


def pop15(ageStack):
    '''
    Objective: To display and delete the elements of the stack in LIFO
    order
    Input Parameters:
        ageStack - list used as a stack to store names of students with
        age greater than 15
    Return Value: None
    '''
    while ageStack:
        print(ageStack.pop())

# Main program
D = {"SHANTI": 14, "Joe": 20, "Prabhjot": 12, "Tehzeeb": 16}
ageStack = []

# Push names of students whose age is greater than 15
push15(D, ageStack)

# Display and delete elements from the stack in LIFO order
pop15(ageStack)
```

6. Computer Networks

Assessment

A.	1. a	2. c	3. d	4. b	5. b	6. c
	7. b	8. d	9. b	10. a	11. b	12. c
	13. c	14. b				
B.	1. **False**	2. **True**	3. **True**	4. **False**	5. **True**	
	6. **True**	7. **False**	8. **True**	9. **True**		
C.	1. **Wi-Fi**	2. **packet**	3. **data rate**	4. **light**	5. **circuit**	
	6. **star**	7. **Bit rate**	8. **Bandwidth**			

D. 1. Give full form of the following:

MAC: Media Access Control

NIC: Network Interface Card

ARPANET: Advanced Research Projects Agency Network

LAN: Local Area Network

MAN: Metropolitan Area Network

WAN: Wide Area Network

2. Name four types of networks.

LAN (Local Area Network)

MAN (Metropolitan Area Network)

WAN (Wide Area Network)

PAN (Personal Area Network)

3. Give two examples of MAN.

City-wide Wi-Fi network

Cable TV network within a city

4. Define the following terms:

(i) **Transmission medium:** The physical path between the transmitter and receiver in a communication system.

(ii) **Topology:** The arrangement of various elements (links, nodes, etc.) of a computer network.

(iii) **Bandwidth:** The maximum rate at which data can be transmitted over a network or internet connection, typically measured in bits per second (bps). It indicates the capacity of the connection.

(iv) **Communication channel:** A medium used to transport information from a sender to a receiver.

5. Differentiate between the followings:

(i) **Packet Switching and Circuit Switching**

- **Packet Switching:** Data is divided into packets, each taking an independent path to the destination. Efficient for variable data rates and robust against line failures.
- **Circuit Switching:** A dedicated communication path is established between the sender and receiver for the duration of the transmission. Provides a constant connection with fixed bandwidth.

(ii) **Star topology and Tree topology:**

- **Star Topology:** All nodes are connected to a central hub.
- **Tree Topology:** A hierarchy of nodes where each node is connected to a parent node, forming a tree-like structure.

(iii) **Bus topology and Ring topology:**

- **Bus Topology:** All nodes share a common communication line (bus).
- **Ring Topology:** Each node is connected to exactly two other nodes, forming a ring.

(iv) **LAN and PAN:**

- **LAN:** Network covering a small geographical area like a single building.
- **PAN:** Network covering a very small area, typically within a range of a few meters (e.g., Bluetooth devices).

(v) **MAN and WAN:**

- **MAN:** Network covering a city.
- **WAN:** Network covering a large geographical area, possibly worldwide.

(vi) **Hub and Switch:**

- **Hub:** A basic networking device that connects multiple devices in a LAN.
- **Switch:** A more advanced device that connects multiple devices and can filter and forward data to the correct destination.

(vii) **Modem and Router:**

- **Modem:** Device that modulates and demodulates signals for data transmission.

- **Router:** Device that forwards data packets between computer networks.

(viii) **Guided and unguided transmission media:**

- **Guided:** Uses physical wires or cables (e.g., twisted pair, coaxial cable).
- **Unguided:** Uses wireless methods (e.g., radio waves, infrared).

6. **Network used by Ramya:**

   PAN (Personal Area Network)

7. **Data Transfer Rate:** The speed at which data is transmitted from one device to another, usually measured in bits per second (bps).

8. **Bit Rate vs. Baud Rate:**

   - **Bit Rate:** The number of bits transmitted per second.
   - **Baud Rate:** The number of signal changes (symbols) transmitted per second.

   Example: If each symbol represents 2 bits, a baud rate of 1000 symbols per second results in a bit rate of 2000 bits per second.

9. **Minimum required data rate:**

   Calculation:

   - 1 page = 600 characters
   - 12 pages = 12 * 600 = 7200 characters
   - 7200 characters * 8 bits/character = 57600 bits
   - Data rate = 57600 bits / 15 seconds = 3840 bits per second (bps)

10. **Three Types of Cables in Computer Networks:**

    - Twisted Pair Cable
    - Coaxial Cable
    - Optical Fiber Cable

11. **Different Ways Devices May Be Connected in a LAN:**

    - Star Topology
    - Bus Topology
    - Ring Topology
    - Mesh Topology
    - Tree Topology

12. **Example of Wired PAN:**

    - USB (Universal Serial Bus) connections between a computer and peripherals like a printer or external hard drive.

13. **Two Characteristics of a Hub:**
    - It broadcasts data to all devices in the network segment.
    - Operates at the physical layer (Layer 1) of the OSI model.
14. **Modem**: A device that modulates and demodulates signals for data transmission over telephone lines or cable.

    Categorization: Internal modem (inside a computer) and external modem (connected externally to a computer).
15. **Short Note on Modems:**
    - A modem (modulator-demodulator) converts digital data from a computer into analog signals for transmission over telephone or cable lines and vice versa. Modems enable internet access by connecting to ISPs and are essential for dial-up and broadband connections. Types include DSL, cable, and fiber modems, which differ based on the technology and medium they use.
16. **Device for Wireless Internet Access:** A wireless router or Wi-Fi access point.
17. **Ascending Order of Frequencies:**
    - Radio Waves
    - Infrared
    - Microwaves
18. **Best Transmission Media for Hilly Areas:** Satellite communication, as it provides wide coverage and is not impeded by physical terrain.
19. **Differentiation between Sender and Receiver:**
    - **Sender:** The device or person that initiates and transmits data.
    - **Receiver:** The device or person that accepts and processes the received data.
20. **Use of MAC Address:** Identifies devices on a local network and ensures that data is delivered to the correct hardware.
21. **Short Note on Microwaves:**
    - Microwaves are electromagnetic waves with frequencies ranging from 1 GHz to 300 GHz. They are used in communication systems like satellite and mobile networks due to their ability to carry large amounts of data over long distances. Microwaves require line-of-sight transmission and are also used in radar and cooking appliances.

Assertion and Reasoning Based Questions

1. a. Both A and R are true and R is the correct explanation of A

2. d. `A is false but R is true`
3. a. `Both A and R are true and R is the correct explanation of A`

Case-based Questions

1. a. Mumbai HO — Gurugram Operational Centre — Kolkata Branch Office

```
         |                              |
   Madurai Branch Office    Mumbai HO
```

   b. Gurugram Operational Centre
   c. (A)
   d. (A)
2. a. Administrative Unit
   b. Administrative Unit - Oncology Unit - Nephrology Unit - Neurology Unit - Orthopaedic Unit
   c. Switch
   d. **Topology:** Star
   Network cable: Ethernet cables

7. Computer Networks: Protocols, Technologies, and Web Services

Assessment

A.	1. b	2. b	3. a	4. a	5. a	6. c
	7. a.	8. d	9. a	10. a	11. b	12. d
	13. c	14. a	15. d			
B.	1. True	2. False	3. True	4. True	5. False	
	6. True	7. True	8. True	9. False	10. True	
C.	1. VoIP (Voice over Internet Protocol)			2. TELNET	3. 3rd	
	4. World Wide Web Consortium (W3C)			5. IP address	6. static	
	7. script	8. Wireless	9. IP address	10. communication protocols		

D. 1.
- **DNS:** Domain Name System
- **POP3:** Post Office Protocol 3
- **TELNET:** Terminal Network

- **CDMA:** Code Division Multiple Access
- **MMS:** Multimedia Messaging Service

2. TCP (Transmission Control Protocol) operates in the transport layer of the OSI model. It provides reliable, ordered, and error-checked delivery of a stream of data between applications. TCP segments the data from the application layer, attaches headers, and ensures that data packets are delivered to the correct application on the receiving end, working seamlessly with IP in the network layer.
3. The Transmission Control Protocol (TCP) ensures the reliable transmission of data across a network. It establishes a connection between sender and receiver, manages data segmentation, controls flow to prevent congestion, and handles error correction by retransmitting lost packets.
4. An FTP (File Transfer Protocol) session involves two main components: the client and the server. The client initiates a connection to the FTP server using the FTP protocol. Once connected, the client can upload, download, and manage files on the server. The session typically involves user authentication (username and password), navigating directories, and performing file operations using commands like GET (download), PUT (upload), LIST (list files), and DELETE (remove files). The communication can occur in active or passive modes, depending on the way the data connection is established between the client and server.
5. The protocol used is **PPP (Point-to-Point Protocol)**.
6. Do It Yourself
7. Do It Yourself
8. Do It Yourself
9. Examples include **XMPP (Extensible Messaging and Presence Protocol)** and **IRC (Internet Relay Chat)**.
10. **VoIP (Voice over Internet Protocol)** allows users to make voice calls using a broadband Internet connection instead of a regular (or analog) phone line.
11. **Radio waves** are used to connect a device to the Internet via a WiFi hotspot.
12. **WiFi:** Typically covers a shorter range (up to 100 meters) and is used for local area networks (LANs) such as in homes, cafes, and offices.

    WiMax: Covers a much larger range (up to 50 kilometers) and is designed for metropolitan area networks (MANs), providing broadband wireless access over a wider area.
13. Fifth Generation (5G) mobile technology has made businesses more efficient by giving consumers faster access to information. Like 4G mobile technology, 5G technology is also based on packet switching. 5G supports the data transfer speed of approximately 10 Gbps. Thus, it enables low latency, i.e., significantly reduced data transfer time. As of now, 5G Ultra-Wideband is available in several cities of India.
14. (i) **Web Server:** A web server is a software or hardware that serves web pages to users in response to their requests, which are made via web browsers.

(ii) **Web Browser:** A web browser is a software application used to access information on the World Wide Web.

(iii) **URL (Uniform Resource Locator):** A URL is the address used to access web pages and resources on the Internet.

(iv) **Web Hosting:** Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet.

(v) **Web Scripting:** The process of writing code to automate tasks and enhance interactivity on web pages. This is usually done using languages like JavaScript, PHP, or Python, allowing dynamic content generation, form validation, and user interaction.

15. The **World Wide Web Consortium (W3C)** is an international community that develops open standards to ensure the long-term growth of the Web.
16. The **Domain Name System (DNS)** translates domain names into IP addresses, allowing browsers to load Internet resources.
17. A web browser works by sending a request to a web server, receiving the HTML document in response, and rendering it into a web page that users can interact with.
18. **HTML (HyperText Markup Language):**
    - **Purpose:** Used to create and design the structure of web pages.
    - **Functionality:** Defines the layout and appearance of web content, including text, images, links, and other multimedia elements.
    - **Static:** Primarily used for displaying static content.
    - **Predefined Tags:** Uses predefined tags such as <html>, <body>, <div>, etc.

    XML (eXtensible Markup Language):
    - **Purpose:** Used to store and transport data.
    - **Functionality:** Focuses on the data itself and its structure, not its presentation.
    - **Dynamic:** Can be used to represent dynamic data that can be manipulated and transferred across different systems.
    - **Custom Tags:** Allows users to define their own tags to suit the data structure.
19. - **Social Networking Sites** (e.g., Facebook, Twitter)
    - **Content Sharing Platforms** (e.g., YouTube, Flickr)
    - **Collaborative Tools** (e.g., Google Docs, Wikipedia)
20. **Client-Side Scripting:**
    - **Execution:** Runs on the user's browser.
    - **Purpose:** Used to create interactive web pages that respond to user input without needing to reload the page.

      Examples: JavaScript, HTML, CSS.

- **Interaction:** Can manipulate the DOM (Document Object Model) and enhance user experience through dynamic content updates.

Server-Side Scripting:

- **Execution:** Runs on the web server.
- **Purpose:** Used to generate dynamic content, manage databases, and handle server logic before sending the page to the client.

  Examples: PHP, ASP.NET, Python, Ruby.
- **Interaction:** Processes user input, interacts with databases, and generates HTML to be sent to the client.

Assertion and Reasoning Based Questions

1. b. Both A and R are true and R is not the correct explanation of A.
2. d. A is false but R is true.
3. a. Both A and R are true and R is the correct explanation of A.

Case-based Questions

Protocol: https

Domain Name: www.incredibleindia.org

Path: /content/incredible-india-v2/en/experiences/

Name of the HTML Document: adventure.html

8. Network Security

Assessment

A.	1. d	2. c	3. d	4. a	5. b	6. d
	7. b	8. a	9. c	10. a	11. c	12. d
B.	1. True	2. False	3. True	4. False	5. False	
	6. True	7. True	8. False			
C.	1. **risk**	2. **macro**	3. **attachment**		4. **white hat hacker**	
	5. **firewall**	6. **copyright**				

D. 1. Confidentiality, integrity, availability, authentication, and non-repudiation.

2. An attack is any attempt to destroy, expose, alter, disable, steal, or gain unauthorized access to or make unauthorized use of an asset. **Attack and vulnerability** are related because an attack exploits vulnerabilities.
3. - **Attack:** An intentional attempt to compromise system security.
   - **Vulnerability:** A weakness in the system that can be exploited by an attacker. Example: SQL injection (attack) exploiting a lack of input validation (vulnerability).
4. Both replicate and spread, infecting their hosts and causing harm.
5. A virus can corrupt or delete data, use up system resources, log keystrokes, and provide unauthorized access to attackers.
6. (i) **Virus and Worm:** A virus needs a host file to spread, while a worm is self-replicating and spreads without a host.

   (ii) **HTTP and HTTPS:** HTTP is unsecured, while HTTPS uses SSL/TLS to secure data transmission.

   (iii) **Trademark and Trade Secret:** A trademark is a recognizable sign/design representing a product or service, while a trade secret is confidential information providing a business advantage.

   (iv) **Hacker and Cracker:** A hacker explores systems for educational purposes, while a cracker breaks into systems with malicious intent.
7. (i) **Virus:** Code attaches to a target program and replicates.

   (ii) **Trojan Horse:** "Happy New Year.exe" appears benign but performs a malicious action.

   (iii) **Spam:** Unsolicited commercial emails.

   (iv) **Worm:** Malicious code replicates through email and overwhelms networks.
8. (i) **Section 66A:** Penalty for sending offensive messages through communication service.

   (ii) **Section 66F:** Cyber terrorism (includes inciting communal riots or disharmony).
9. The type of software attack is a Trojan Horse because:
   - The user received an email attachment named "Happy2021" which seemed benign.
   - Upon clicking the attachment, a video played, which is the expected behavior, masking the malicious activity.
   - While the video was playing, a malicious code was downloaded and attached itself to all outgoing emails from the user's computer.

   Trojan Horses are characterized by their ability to disguise themselves as legitimate software or files while performing hidden malicious activities, such as downloading additional malware or spreading itself through emails, as described in the scenario.

Assertion and Reasoning Based Questions

1. c. A is true but R is false

2. a. Both A and R are true and R is the correct explanation of A
3. b. Both A and R are true and R is not the correct explanation of A

Case-based Questions

1. Copyright infringement. Legal protection: Roy can seek legal action under copyright laws, and the producer-director may be liable for damages and an injunction to stop the use of the script.
2. Surabhi should file for a patent to legally protect her drug invention from being used or sold by others without her permission.

9. Database Management System

Assessment

A.	1. c	2. c	3. b	4. a	5. d	6. b
	7. c	8. a	9. d	10. b	11. a	12. c
	13. d	14. c	15. b			
B.	1. True	2. True	3. False	4. True	5. True	
	6. True	7. False				
C.	1. Attribute	2. Schema	3. Degree	4. Relation	5. Composite key	
	6. Domain					

D. 1.

(i) `True`	(ii) `True`	(iii) `True`	(iv) `True`	(v) `False`
(vi) `True`	(vii) `True`	(viii) `False`	(ix) `False`	(x) `True`
(xi) `True`				

2. **`ItemCode: INT`**
   - `This will store integer values.`

   `ItemName: VARCHAR`
   - `You can specify the maximum length for the item name. For example, if the maximum length of the item name is expected to be 255 characters, you can use` **`VARCHAR(255)`**`.`

   `Price: DECIMAL`
   - `The` **`DECIMAL`** `type is used for storing precise numeric values, especially when dealing with financial data. You can specify the precision and`

scale. For example, **DECIMAL(10, 2)** allows up to 10 digits in total, with 2 digits after the decimal point.

3. No, a relation cannot have two identical tuples because it would violate the principle of a set in relational databases, where all tuples must be unique.

4. (i) **Domain:** A domain is the set of permissible values that an attribute can have.

   (ii) **Constraint:** A constraint is a rule that restricts the values that can be stored in a database to ensure data integrity.

   (iii) **Candidate Key:** A candidate key is an attribute, or a set of attributes, that can uniquely identify a tuple in a relation.

   (iv) **Alternate Key:** An alternate key is any candidate key that is not chosen as the primary key.

5. **Primary Keys:**

   - **Employee Table:** Emp_id
   - **Department Table:** Dept_no

   Insert Operations:

   a. An insert operation that will be consistent with the current state of the Employee table:

   INSERT INTO Employee (Emp_id, Name, Dept_no) VALUES ('E005', 'Anjali Gupta', 20);

   b. An insert operation that will be consistent with the current state of the Department table:

   INSERT INTO Department (Dept_no, Dept_name) VALUES (40, 'Human Resources');

   c. An insert operation that will be inconsistent with the current state of the Employee table, but would be fine if the Employee table were empty:

   INSERT INTO Employee (Emp_id, Name, Dept_no) VALUES ('E001', 'Rajesh Kumar', 10);

   (**Note:** This will cause a duplicate primary key error because E001 already exists.)

   d. An insert operation that will be inconsistent with the current state of the Department table, but would be fine if the Department table were empty:

   INSERT INTO Department (Dept_no, Dept_name) VALUES (10, 'Finance');

   (**Note:** This will cause a duplicate primary key error because 10 already exists.)

   e. An insert operation that will be invalid on the empty table Employee:

   INSERT INTO Employee (Emp_id, Name, Dept_no) VALUES (NULL, 'Ravi Shastri', 10);

   (**Note:** Emp_id cannot be NULL as it is a primary key.)

   f. An insert operation that will be invalid on the empty table Department:

   INSERT INTO Department (Dept_no, Dept_name) VALUES (NULL, 'Operations');

   (**Note:** Dept_no cannot be NULL as it is a primary key.)

   g. Will an operation to delete the tuple having Dept_no 20 be consistent with the current state of the Employee and Department tables? Justify your answer:

No, it will be inconsistent because there are employees (Emp_id E003) associated with Dept_no 20 in the Employee table. Deleting the department will violate the foreign key constraint.

6. **Primary Keys:**

   - **Employee Table:** E_id
   - **Project Table:** P_no
   - **WorksOn Table:** (P_no, E_id) (Composite primary key)

   Foreign Keys:

   - WorksOn Table: P_no (references Project.P_no), E_id (references Employee.E_id)

   Insert Operations:

   a. A tuple insertion that will be invalid on the empty table Employee:

      INSERT INTO Employee (E_id, Ename, City, Salary, Department, YearofJoining) VALUES (NULL, 'Alok Sharma', 'Delhi', 50000, 'HR', 2018);

      (**Note:** E_id cannot be NULL as it is a primary key.)

   b. A tuple insertion that will be invalid on the empty table Project:

      INSERT INTO Project (P_no, PName, City, DeptName, StartYear) VALUES (NULL, 'New Project', 'Mumbai', 'IT', 2024);

      (**Note:** P_no cannot be NULL as it is a primary key.)

   c. A tuple insertion that will be invalid on the empty table WorksOn:

      INSERT INTO WorksOn (P_no, E_id) VALUES (1, 1);

      (**Note:** Foreign keys P_no and E_id do not exist in the Project and Employee tables respectively.)

   d. A tuple insertion that will be valid on the empty table Employee:

      INSERT INTO Employee (E_id, Ename, City, Salary, Department, YearofJoining) VALUES (1, 'Alok Sharma', 'Delhi', 50000, 'HR', 2018);

   e. A tuple insertion that will be valid on the empty table Project:

      INSERT INTO Project (P_no, PName, City, DeptName, StartYear) VALUES (1, 'New Project', 'Mumbai', 'IT', 2024);

   f. A tuple insertion that will be valid on the empty table WorksOn:

      INSERT INTO WorksOn (P_no, E_id) VALUES (1, 1);

      (**Note:** This would require the Employee and Project tables to have the corresponding entries.)

7. **Primary Keys:**

   - **Suppliers Table:** SNo
   - **Parts Table:** PNo
   - **Project Table:** JNo

- **Shipment Table:** (SNo, PNo, JNo) (Composite primary key)

Foreign Keys:

- **Shipment Table:** SNo (references Suppliers.SNo), PNo (references Parts.PNo), JNo (references Project.JNo)

8. **Insert Operations:**

   (i) An insertion on the empty table Suppliers that will generate an error:

   INSERT INTO Suppliers (SNo, SName, Status, SCity) VALUES (NULL, 'ABC Supplies', 'Active', 'Mumbai');

   (**Note:** SNo cannot be NULL as it is a primary key.)

   (ii) An insertion on the empty table Parts that will generate an error:

   INSERT INTO Parts (PNo, PName, Colour, Weight, City) VALUES (NULL, 'Bolt', 'Silver', 50, 'Delhi');

   (**Note:** PNo cannot be NULL as it is a primary key.)

   (iii) An insertion on the empty table Suppliers that will be successful:

   INSERT INTO Suppliers (SNo, SName, Status, SCity) VALUES (1, 'ABC Supplies', 'Active', 'Mumbai');

   (iv) An insertion on the empty table Parts that will be successful:

   INSERT INTO Parts (PNo, PName, Colour, Weight, City) VALUES (1, 'Bolt', 'Silver', 50, 'Delhi');

   (v) An insertion on the empty table Project that will be successful:

   INSERT INTO Project (JNo, JName, JCity) VALUES (1, 'Project Alpha', 'Bangalore');

   (vi) An insertion on the empty table Shipment that will be successful:

   INSERT INTO Shipment (SNo, PNo, JNo, Quantity) VALUES (1, 1, 1, 100);

   (**Note:** This would require the Suppliers, Parts, and Project tables to have the corresponding entries.)

9. **Primary and Foreign Keys:**

   - **Employee Table:**
     - ◊ Primary Key: employee_id
     - ◊ Foreign Key: department_id (references Department.department_id)
   - **Department Table:**
     - ◊ Primary Key: department_id
   - **Resources Table:**
     - ◊ Primary Key: resource_id
     - ◊ Foreign Key: department_id (references Department.department_id)

Table Creation Order:

1. Department (since other tables reference department_id)
2. Employee
3. Resources

10. **Entity Integrity:**
    - Ensures that each table has a primary key and that the column or columns chosen to be the primary key are unique and not null.
    - Example: In the Employee table, employee_id must be unique and not null.

    Referential Integrity:
    - Ensures that a foreign key value always points to an existing row.
    - Example: In the Employee table, department_id must match a department_id in the Department table.

11. **Insert Operations:**

    (i) <18, "Mukesh Agrawal", 11, "C", 88>
    - Executed successfully.

    (ii) <21, "Sanjay", 11, "A", 76>
    - Executed successfully.

    (iii) <NULL, "Phule Bai", 11, "A", 88>
    - Not executed successfully. Violates the primary key constraint (S_ID cannot be NULL).

    (iv) <20, NULL, 11, "A", 88>
    - Executed successfully.

    (v) <20, NULL, 11, NULL, NULL>
    - Executed successfully.

    (vi) <20, NULL, NULL, NULL, NULL>
    - Executed successfully.

12. **Insert Operations:**

    (i) <799575933699, NULL, "Pooja", "1990-10-01", 9776626565>
    - Not executed successfully. Violates the NOT NULL constraint on last_name.

    (ii) <712349049911, "Singh", "Gagan", "1990-11-11", 9812476543>
    - Not executed successfully. Violates the uniqueness constraint on AadharNo.

Assertion and Reasoning Based Questions

1. **Correct Answer:** (a) Both A and R are true and R is the correct explanation of A.
   - **Explanation:** This statement is true because constraints ensure data integrity by enforcing rules on attribute values. Constraints such as NOT NULL, UNIQUE, PRIMARY KEY, etc., are specified for attributes to maintain data accuracy and consistency.
2. **Correct Answer:** (b) Both A and R are true and R is not the correct explanation of A.
   - **Explanation:** The relational schema defines the structure of a relation (table), including attributes and their types. Adding rows or columns does not change the schema; it changes the instance (data) of the relation, not its structure.
3. **Correct Answer:** (c) A is true but R is false.
   - **Explanation:** The entity set refers to the set of all entities (rows/tuples) in a relation. Deleting a record removes a tuple from the entity set, but it does not modify the entity set itself; it reduces its cardinality.
4. **Correct Answer:** (a) Both A and R are true and R is the correct explanation of A.
   - **Explanation:** This statement is true because in database systems, NULL represents unknown or missing data. If an attribute like "spouse name" is not applicable (as in the case of an unmarried candidate), it would typically be represented as NULL.

Case-based Questions

When inserting data into tables that have foreign key constraints, it's crucial to ensure that the referenced records exist. Here's how you can handle the scenario described:

1. **Inserting a New Department and Employee:** To insert a new department (Dept_No = 6) and a new employee (with manager E0011):
   - Step 1: Insert the employee who will be the manager (E0011) into the Employee table first.
   - Step 2: Once the employee record (E0011) exists, then insert the department (Dept_No = 6) into the Department table, referencing the manager (E0011).

2. This sequence ensures that when you insert the department record, the foreign key reference to the manager (E0011) exists in the Employee table, hence preserving referential integrity

1. **Relational Schema:**

```
CREATE TABLE myClass (
    Name VARCHAR(50),
    Father_Name VARCHAR(50),
    Contact_Number VARCHAR(15),
    Birth_Date DATE,
    Hobbies TEXT,
    Class VARCHAR(10),
    PRIMARY KEY (Name)
);
```

   - **Primary Key:** Name (assuming names are unique within the class).

2. **Table1: Events**

```
CREATE TABLE Events (
    Event_Id INT,
    Event_Name VARCHAR(50),
    Event_Date DATE,
    PRIMARY KEY (Event_Id)
);
```

   - **Primary Key:** Event_Id

   Table2: Players

```
CREATE TABLE Players (
    Player_Id INT,
    Player_Name VARCHAR(50),
    Age INT,
    Event_Id INT,
    Result VARCHAR(50),
    PRIMARY KEY (Player_Id),
    FOREIGN KEY (Event_Id) REFERENCES Events(Event_Id)
);
```

   - **Primary Key:** Player_Id
   - **Foreign Key:** Event_Id references Events(Event_Id)

3. **Trains Table:**
   - **Primary Key:** TrainID
   - **Foreign Key:** No explicit foreign key constraint shown, assuming TrainID may reference other tables like Stations or Schedules.

   Passengers Table:
   - **Primary Key:** RefNo
   - **Foreign Key:** TrainID references Trains(TrainID)

   Degree and Cardinality:
   - **Trains Table:**
     ◊ Degree: 4 (TrainID, TrainName, Source, Destination)
     ◊ Cardinality: Number of rows (5 in this case)
   - **Passengers Table:**
     ◊ Degree: 4 (RefNo, TrainID, PassengerName, DOJ)
     ◊ Cardinality: Number of rows (4 in this case)

   Can TrainID in Passengers Table have the value 9999?
   - It depends on the constraints defined. Normally, if 9999 is not an existing TrainID in the Trains table (assuming it's a foreign key), inserting 9999 into Passengers(TrainID) would violate referential integrity unless NULLs are allowed.

10. Structured Query Language (SQL)

Assessment

A.	1. a	2. b	3. a	4. c	5. a	6. b
	7. b	8. b	9. d	10. d	11. b	12. c
	13. a	14. d	15. b	16. c		
B.	1. False	2. False	3. True	4. True	5. True	
	6. False					
C.	1. **DISTINCT**	2. **primary**	3. **CHECK constraint**		4. **UNIQUE**	
	5. **GROUP BY**	6. **outer, inner**		7. **ORDER BY**		

D. 1. (i) To view the list of databases:

SHOW DATABASES;

(ii) To start using the database named TEST:

```
USE TEST;
```

(iii) To view the structure of table PRACTICE:

```
DESCRIBE PRACTICE;
```

(iv) To display all the records from table PRACTICE:

```
SELECT * FROM PRACTICE;
```

2. ORDER BY: This clause is used to sort the result set in either ascending or descending order. It does not affect the grouping of rows.

   Example:

   ```
   SELECT * FROM Employee ORDER BY Salary DESC;
   ```

   GROUP BY: This clause is used to arrange identical data into groups. It is often used with aggregate functions like COUNT, SUM, AVG, etc.

   Example:

   ```
   SELECT Dept_no, COUNT(*) FROM Employee GROUP BY Dept_no;
   ```

3. **Wildcard Characters for Pattern Matching with LIKE**
   - %: Matches any sequence of characters (including zero characters).
   - _: Matches any single character.

4. (i) **CREATE TABLE statement for table TEACHER:**

```
CREATE TABLE TEACHER (
    ID CHAR(5) PRIMARY KEY,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50),
    Dept VARCHAR(50),
    Contact_Num VARCHAR(15),
    Salary INT,
    Email_ID VARCHAR(100)
);
```

(ii) **SELECT Statements:**

a. **List the salary of those teachers whose name starts with 'S':**

```
SELECT Salary FROM TEACHER WHERE First_Name LIKE 'S%';
```

b. **List the first name and last name of the teachers who have salary more than 70000:**

```
SELECT First_Name, Last_Name FROM TEACHER WHERE Salary > 70000;
```

c. List the count of number of teachers of each department:

```
SELECT Dept, COUNT(*) FROM TEACHER GROUP BY Dept;
```

d. List the first name and contact number of teachers whose mail ID is not known:

```
SELECT First_Name, Contact_Num FROM TEACHER WHERE Email_ID IS
NULL;
```

e. Display the rows from the table TEACHER in descending order of salary:

```
SELECT * FROM TEACHER ORDER BY Salary DESC;
```

f. Add an attribute Subject to the table TEACHER:

```
ALTER TABLE TEACHER ADD Subject VARCHAR(50);
```

g. Drop attribute Email_ID from the table TEACHER:

```
ALTER TABLE TEACHER DROP COLUMN Email_ID;
```

(iii) Output of SQL Queries:

a. `SELECT AVG(Salary) FROM TEACHER WHERE Dept = 'Economics';`

Output:

```
67500
```

b.
```
SELECT First_Name, Last_Name, Contact_Num FROM TEACHER WHERE
Salary BETWEEN 40000 AND 80000;
```

Output:

```
Naishadh Kumar 9965789799
Tenzin Wangdi 8023456780
Krishan Kumar 9977885566
```

c. `SELECT DISTINCT Dept FROM TEACHER;`

Output:

```
Political Science
English
History
Computer Science
Economics
```

d.
```
SELECT MAX(Salary) FROM TEACHER GROUP BY Dept HAVING Dept =
'Political Science';
```

Output:

```
85000
```

5. (i) CREATE TABLE statement for table STUDENT:

```
CREATE TABLE STUDENT (
    Roll_Num INT PRIMARY KEY,
    Student_Name VARCHAR(50),
    Course_Name VARCHAR(50),
    Duration INT,
    Fee INT,
    Batch_Prefer VARCHAR(10)
);
```

(ii) SELECT Statements:

a. List the names of students in each course:

SELECT Student_Name FROM STUDENT;

b. Display the course name along with their duration for the courses whose fee is more than 30000:

SELECT Course_Name, Duration FROM STUDENT WHERE Fee > 30000;

c. Display the names of students in alphabetical order:

SELECT Student_Name FROM STUDENT ORDER BY Student_Name;

d. Display the count of students who prefer the Evening batch:

SELECT COUNT(*) FROM STUDENT WHERE Batch_Prefer = 'Evening';

e. Display the average duration of courses from the table:

SELECT AVG(Duration) FROM STUDENT;

f. Increase the fee by 10% for Mobile App course:

UPDATE STUDENT SET Fee = Fee * 1.10 WHERE Course_Name = 'Mobile App';

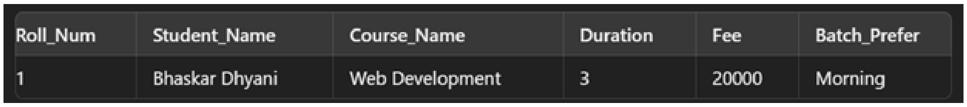
g. Delete rows from table student where duration of the course is not known:

DELETE FROM STUDENT WHERE Duration IS NULL;

(iii) Output of SQL Queries:

a. SELECT * FROM STUDENT WHERE Student_Name LIKE '%h' AND Fee < 25000;

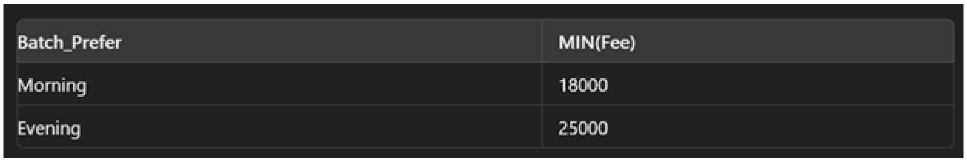
Output:



Roll_Num	Student_Name	Course_Name	Duration	Fee	Batch_Prefer
1	Bhaskar Dhyani	Web Development	3	20000	Morning

b. SELECT MIN(Fee) FROM STUDENT GROUP BY Batch_Prefer;

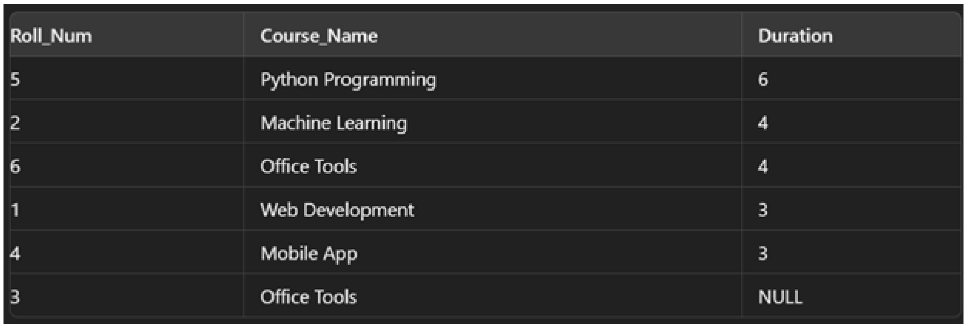
Output:



Batch_Prefer	MIN(Fee)
Morning	18000
Evening	25000

c. SELECT Roll_Num, Course_Name, Duration FROM STUDENT ORDER BY Duration DESC;

Output:



Roll_Num	Course_Name	Duration
5	Python Programming	6
2	Machine Learning	4
6	Office Tools	4
1	Web Development	3
4	Mobile App	3
3	Office Tools	NULL

6. (i) **SELECT Statements:**

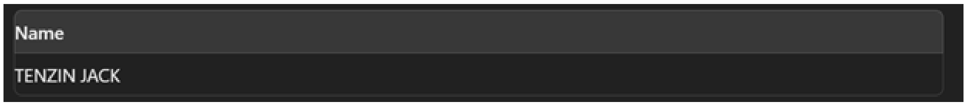
a. List the names of salespersons who are dealing with Books in Noida.

SELECT Name

FROM SALESPERSON

WHERE Product = 'Books' AND City = 'Noida';

Output:



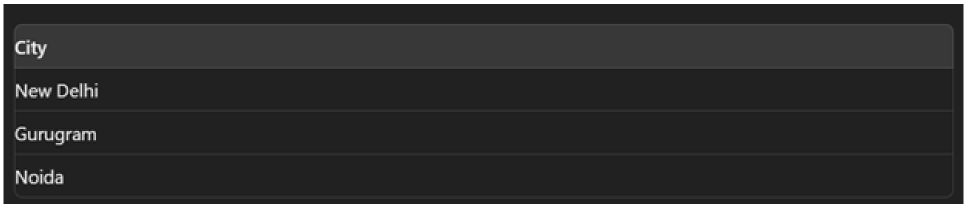
Name
TENZIN JACK

b. Display the names of cities without any repetition.

SELECT DISTINCT City

FROM SALESPERSON;

Output:



City
New Delhi
Gurugram
Noida

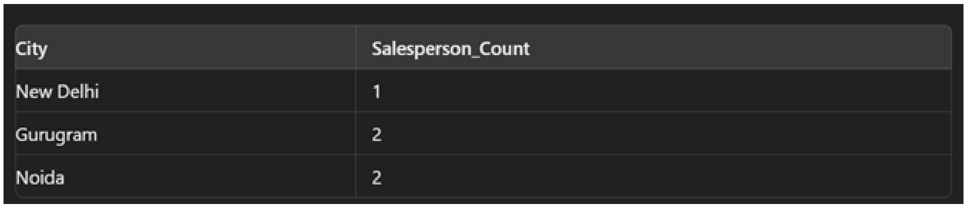
c. Display the count of salespersons in each city.

SELECT City, COUNT(*) AS Salesperson_Count

FROM SALESPERSON

GROUP BY City;

Output:



City	Salesperson_Count
New Delhi	1
Gurugram	2
Noida	2

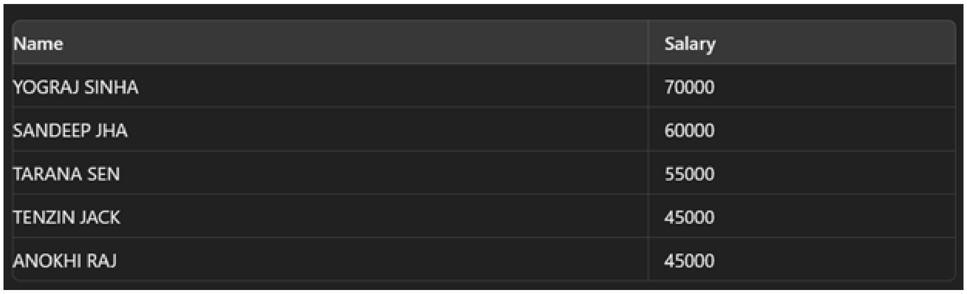
d. Display the name and salary of salespersons in descending order of Salary.

SELECT Name, Salary

FROM SALESPERSON

ORDER BY Salary DESC;

Output:



Name	Salary
YOGRAJ SINHA	70000
SANDEEP JHA	60000
TARANA SEN	55000
TENZIN JACK	45000
ANOKHI RAJ	45000

e. Delete the salesperson whose salary is more than 60000.

DELETE FROM SALESPERSON

WHERE Salary > 60000;

After this operation, YOGRAJ SINHA will be removed from the table.

f. Add a field Contact_No in the table SALESPERSON to store the mobile number of salespersons.

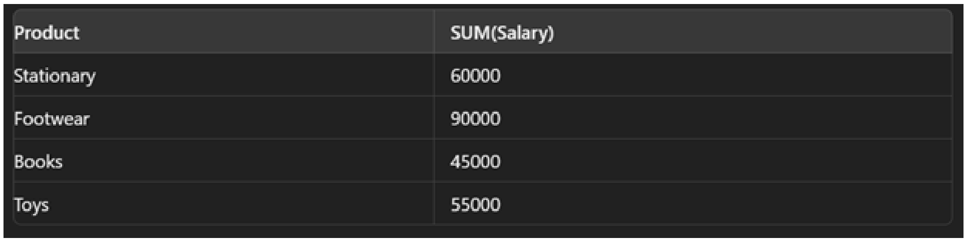
ALTER TABLE SALESPERSON

ADD Contact_No VARCHAR(15);

(ii) **Updated Outputs for the given SQL queries:**

a. SELECT Product, SUM(Salary) FROM SALESPERSON GROUP BY Product;

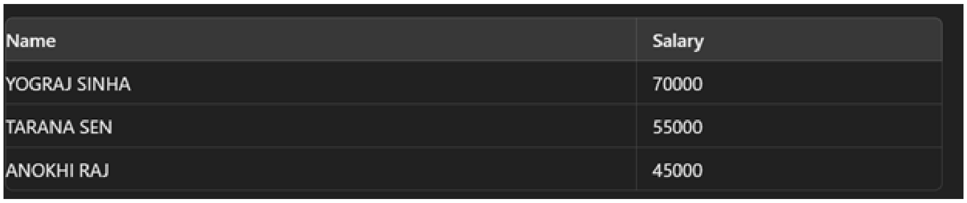
Output:



Product	SUM(Salary)
Stationary	60000
Footwear	90000
Books	45000
Toys	55000

b. SELECT Name, Salary FROM SALESPERSON WHERE Product IN ('Toys','Footwear');

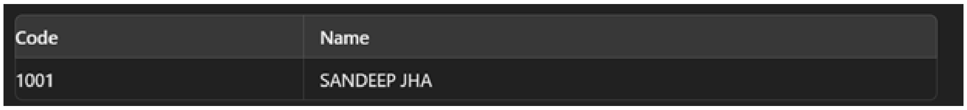
Output:



Name	Salary
YOGRAJ SINHA	70000
TARANA SEN	55000
ANOKHI RAJ	45000

c. SELECT Code, Name FROM SALESPERSON WHERE Salary > 50000 AND Name LIKE '%E%';

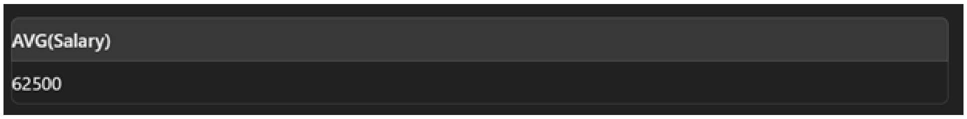
Output:



Code	Name
1001	SANDEEP JHA

d. SELECT AVG(Salary) FROM SALESPERSON WHERE CITY='Gurugram';

Output:



AVG(Salary)
62500

7. (i) **SQL Queries**

a. To insert a new tuple in the MOBILE table whose M_Price is not yet known.

INSERT INTO MOBILE (M_Id, M_Company, M_Name, Launch_Date)

VALUES ('MB007', 'OnePlus', 'OnePlus5', NULL, '2018-01-01');

b. Display the mobile name and name of the company of the mobile phones whose price is greater than 5000.

SELECT M_Name, M_Company

FROM MOBILE

WHERE M_Price > 5000;

Output:

```
+---------+-----------+
| M_Name  | M_Company |
+---------+-----------+
| XperiaM | Sony      |
| SefieEx | Oppo      |
+---------+-----------+
```

c. List the name of the mobile phones along with their price that were launched in the year 2017.

SELECT M_Name, M_Price

FROM MOBILE

WHERE YEAR(Launch_Date) = 2017;

Output:

```
+----------+---------+
| M_Name   | M_Price |
+----------+---------+
| XperiaM  | 7500    |
+----------+---------+
```

d. Display M_Company, M_Name, and M_Price in descending order of their launch date.

SELECT M_Company, M_Name, M_Price

FROM MOBILE

ORDER BY Launch_Date DESC;

Output:

```
+-----------+---------+---------+
| M_Company | M_Name  | M_Price |
+-----------+---------+---------+
| Sony      | XperiaM | 7500    |
| Micromax  | Unite3  | 4500    |
| Samsung   | Galaxy  | 4500    |
| Nokia     | N1100   | 2250    |
| Oppo      | SefieEx | 8500    |
+-----------+---------+---------+
```

e. List the details of a mobile whose name starts with "S" or ends with "a".

SELECT *

FROM MOBILE

WHERE M_Name LIKE 'S%' OR M_Name LIKE '%a';

```
Output:

+--------+-----------+---------+---------+-------------+
| M_Id   | M_Company | M_Name  | M_Price | Launch_Date |
+--------+-----------+---------+---------+-------------+
| MB001  | Samsung   | Galaxy  | 4500    | 2013-02-12  |
| MB006  | Oppo      | SefieEx | 8500    | 2010-08-21  |
+--------+-----------+---------+---------+-------------+
```

f. Display the names of the mobile companies having prices between 3000 and 5000.

SELECT M_Company

FROM MOBILE

WHERE M_Price BETWEEN 3000 AND 5000;

```
Output:

+-----------+
| M_Company |
+-----------+
| Samsung   |
| Micromax  |
+-----------+
```

(ii) **Outputs for the given SQL queries**

a. SELECT MAX(Launch_Date), MIN(Launch_Date) FROM MOBILE;

```
Output:

+------------------+------------------+
| MAX(Launch_Date) | MIN(Launch_Date) |
+------------------+------------------+
| 2017-11-20       | 2010-08-21       |
+------------------+------------------+
```

b. SELECT AVG(M_Price) FROM MOBILE;

```
Output:

+--------------+
| AVG(M_Price) |
+--------------+
| 5450.00      |
+--------------+
```

c. SELECT M_Name, Launch_Date FROM MOBILE WHERE M_Company='Nokia' OR M_Price IS NULL;

Output:

```
+--------+--------------+
| M_Name | Launch_Date  |
+--------+--------------+
| N1100   | 2011-04-15   |
| OnePlus5 | 2018-01-01 |
+--------+--------------+
```

The OnePlus5 entry is included since it has a NULL price, matching the condition.

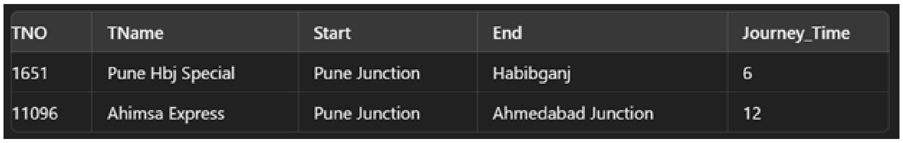
8. (i) **SQL Queries**

a. To display details of all trains which start from Pune Junction.

SELECT * FROM TRAINS

WHERE Start = 'Pune Junction';

Output:



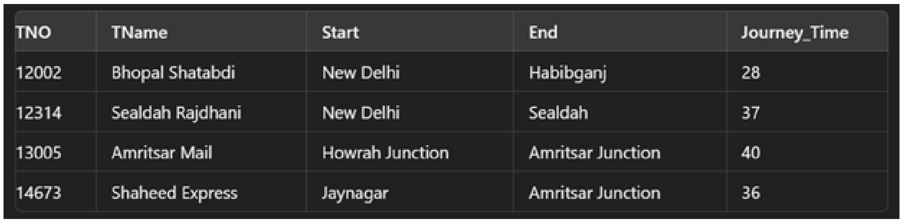
TNO	TName	Start	End	Journey_Time
1651	Pune Hbj Special	Pune Junction	Habibganj	6
11096	Ahimsa Express	Pune Junction	Ahmedabad Junction	12

b. To display details of trains that have a duration of more than 15 hours.

SELECT * FROM TRAINS

WHERE Journey_Time > 15;

Output:



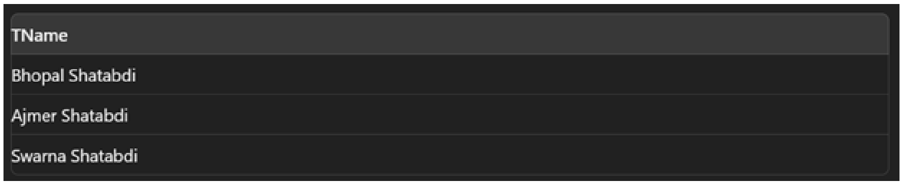
TNO	TName	Start	End	Journey_Time
12002	Bhopal Shatabdi	New Delhi	Habibganj	28
12314	Sealdah Rajdhani	New Delhi	Sealdah	37
13005	Amritsar Mail	Howrah Junction	Amritsar Junction	40
14673	Shaheed Express	Jaynagar	Amritsar Junction	36

c. To display the names of trains that end with the word "Shatabdi".

SELECT TName FROM TRAINS

WHERE TName LIKE '%Shatabdi';

Output:



TName
Bhopal Shatabdi
Ajmer Shatabdi
Swarna Shatabdi

d. To delete the record of trains that start from Jaynagar.

```
DELETE FROM TRAINS
WHERE Start = 'Jaynagar';
```

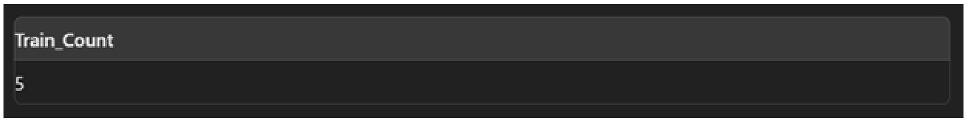
e. To change the duration of Swarna Shatabdi to 7 hours.

```
UPDATE TRAINS
SET Journey_Time = 7
WHERE TName = 'Swarna Shatabdi';
```

f. To display the count of trains that end at New Delhi.

```
SELECT COUNT(*) AS Train_Count
FROM TRAINS
WHERE End = 'New Delhi';
```

Output:

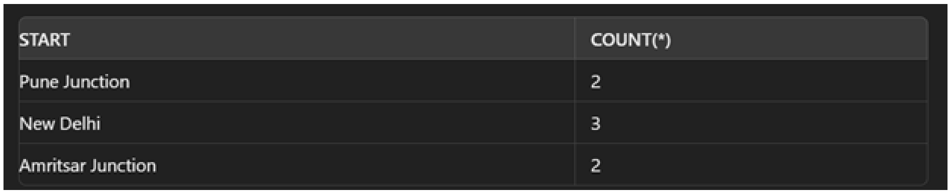


Train_Count
5

(ii) **Outputs for the given SQL queries**

a. SELECT START, COUNT(*) FROM TRAINS GROUP BY START HAVING COUNT(*)>1;

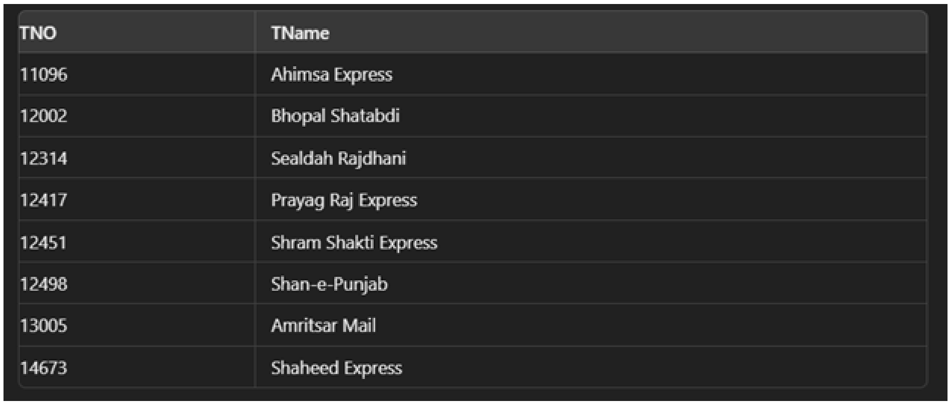
Output:



START	COUNT(*)
Pune Junction	2
New Delhi	3
Amritsar Junction	2

b. SELECT TNO, TName FROM TRAINS WHERE Journey_Time > 6 ORDER BY TNO;

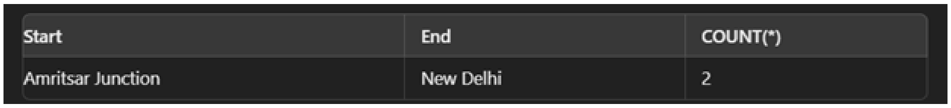
Output:



TNO	TName
11096	Ahimsa Express
12002	Bhopal Shatabdi
12314	Sealdah Rajdhani
12417	Prayag Raj Express
12451	Shram Shakti Express
12498	Shan-e-Punjab
13005	Amritsar Mail
14673	Shaheed Express

c. SELECT Start, End, COUNT(*) FROM TRAINS GROUP BY Start, End HAVING COUNT(*)=2;

Output:



Start	End	COUNT(*)
Amritsar Junction	New Delhi	2

9. (i) **Write the command to create the table STUDENT.**

```
CREATE TABLE STUDENT (
    RollNO INT PRIMARY KEY,
    Name VARCHAR(50),
    Class VARCHAR(5),
    DOB DATE,
    Gender CHAR(1),
    City VARCHAR(50),
    Marks INT
);
```

(ii) **Write an SQL command to insert a new tuple in the table STUDENT whose values are:**

Roll No-10, Name-Aziz, Class-XI, DOB: 15-7-1996, Gender-M, City-Delhi, and marks are not yet known.

```
INSERT INTO STUDENT (RollNO, Name, Class, DOB, Gender, City)
VALUES (10, 'Aziz', 'XI', '1996-07-15', 'M', 'Delhi');
```

(iii) **Write SQL queries for the following based on the table STUDENT:**

(a) **To display the average marks in each class.**

```
SELECT Class, AVG(Marks) AS AvgMarks
FROM STUDENT
GROUP BY Class;
```

(b) **To display the average marks in each class, gender-wise.**

```
SELECT Class, Gender, AVG(Marks) AS AvgMarks
FROM STUDENT
GROUP BY Class, Gender;
```

(c) **To display the number of students in each city.**

```
SELECT City, COUNT(*) AS NumberOfStudents
```

```
FROM STUDENT
GROUP BY City;
```

(d) **To display the records from table STUDENT in alphabetical order of the names of the students.**

```
SELECT *
FROM STUDENT
ORDER BY Name;
```

(e) **To display RollNO, Class, DOB and City for students whose marks are between 450 and 551.**

```
SELECT RollNO, Class, DOB, City
FROM STUDENT
WHERE Marks BETWEEN 450 AND 551;
```

(f) **To increase marks of all students by 20 who are in class XII.**

```
UPDATE STUDENT
SET Marks = Marks + 20
WHERE Class = 'XII';
```

(iv) **What will be the output produced on the execution of each of the following queries:**

(a) SELECT COUNT(*), City FROM STUDENT GROUP BY CITY HAVING COUNT(*)>1;

This query will display the number of students in each city where there is more than one student. The output will be:

```
+----------+--------+
| COUNT(*) | City   |
+----------+--------+
| 2        | Delhi  |
| 2        | Moscow |
| 2        | Mumbai |
+----------+--------+
```

(b) SELECT MAX(DOB), MIN(DOB) FROM STUDENT;

This query will display the latest and earliest dates of birth from the STUDENT table. The output will be:

```
+------------+------------+
| MAX(DOB)   | MIN(DOB)   |
+------------+------------+
| 1995-12-08 | 1993-05-07 |
+------------+------------+
```

(c) SELECT NAME, GENDER FROM STUDENT WHERE CITY='Delhi';

This query will display the names and genders of students who are from Delhi. The output will be:

```
+--------+--------+
| Name   | Gender +
+--------+--------+
| Sanal  | F      |
| Store  | M      |
+--------+--------+
```

10. (i) **SQL Statement to Create the Table**

```
CREATE TABLE ACCOUNTS (
    ANo INT PRIMARY KEY,
    AName VARCHAR(50),
    Branch VARCHAR(50),
    Contact_Number VARCHAR(15),
    Balance_Amount DECIMAL(10, 2),
    Account_Type VARCHAR(20)
);
```

(ii) **SQL Command to Insert a New Tuple**

```
INSERT INTO ACCOUNTS (ANo, AName, Branch, Contact_Number, Balance_
Amount, Account_Type)
VALUES (106, 'Joe Mathews', 'Goa', NULL, 76392.90, 'Savings');
```

(iii) **SQL Queries**

a. To display details of all account holders who have a savings account

SELECT * FROM ACCOUNTS WHERE Account_Type = 'Savings';

b. To display the Name and Contact Number of all account holders Branch wise

SELECT Branch, AName, Contact_Number FROM ACCOUNTS ORDER BY Branch;

c. To display the names of the account holders whose balance is less than 20000

SELECT AName FROM ACCOUNTS WHERE Balance_Amount < 20000;

d. To display Name and Branch of account holders whose contact number is not known

SELECT AName, Branch FROM ACCOUNTS WHERE Contact_Number IS NULL;

e. To display the names of Branch only once

SELECT DISTINCT Branch FROM ACCOUNTS;

(iv) **Output of the Given SQL Queries**

a. SELECT ANO, ANAME FROM ACCOUNTS WHERE BRANCH NOT IN ('CHENNAI', 'BANGALORE');

This query selects the account number and name of the account holders whose branch is neither 'Chennai' nor 'Bangalore'. The output will be:

```
+-----+--------------+
| ANo | AName        |
+-----+--------------+
| 103 | Ali Reza     |
| 105 | Simran Kaur  |
+-----+--------------+
```

b. SELECT DISTINCT ANO FROM ACCOUNTS;

This query selects distinct account numbers from the ACCOUNTS table. Since all account numbers are unique, the output will be:

```
+-----+
| ANo |
+-----+
| 101 |
| 102 |
| 103 |
| 104 |
| 105 |
+-----+
```

c. SELECT ANO, COUNT(*), BALANCE_AMOUNT FROM ACCOUNTS GROUP BY ANO HAVING COUNT(*)> 1;

This query groups the results by account number and counts the number of rows for each account number, then filters the results to only show groups with more than 1 row. Since each account number is unique, the count will always be 1 for each, so no rows will meet the HAVING COUNT(*) > 1 condition. The output will be:

```
Empty set
```

Assertion and Reasoning Based Questions

1. d 2. c 3. b

Case-based Questions

1. (i) Display the structure of the table:

```
DESCRIBE Stationary;
```

(ii) Display records whose quantity is less than 5:

```
SELECT * FROM Stationary WHERE QTY < 5;
```

(iii) Display the name and price of stationary items whose names end with 'Marker':

```
SELECT SNAME, PRICE FROM Stationary WHERE SNAME LIKE '%Marker';
```

(iv) Display the price of the most expensive item:

```
SELECT MAX(PRICE) FROM Stationary;
```

(v) Display the average price of all items:

```
SELECT AVG(PRICE) FROM Stationary;
```

(vi) Display the brand name and count of items of each brand:

```
SELECT BRAND, COUNT(*) FROM Stationary GROUP BY BRAND;
```

(vii) Display the total number of records in the table:

```
SELECT COUNT(*) FROM Stationary;
```

(viii) Display the records in descending order of quantity:

```
SELECT * FROM Stationary ORDER BY QTY DESC;
```

(ix) Display the records whose price is not known:

```
SELECT * FROM Stationary WHERE PRICE IS NULL;
```

(x) Display the name of each brand only once:

```
SELECT DISTINCT BRAND FROM Stationary;
```

2. (i) Display the name and price of each watch:

```
SELECT WName, Price FROM Watches;
```

(ii) Change the price of Apple iWatch to 59350:

```
UPDATE Watches SET Price = 59350 WHERE WName = 'Apple iWatch';
```

(iii) Display the records of all Sports watches:

```
SELECT * FROM Watches WHERE Type = 'Sports Watch';
```

(iv) Display the names of all brands, without repetition:

```
SELECT DISTINCT Brand FROM Watches;
```

(v) Display the count of designer watches:

```
SELECT COUNT(*) FROM Watches WHERE Type = 'Designer';
```

(vi) Display the count of watches of each brand:

```
SELECT Brand, COUNT(*) FROM Watches GROUP BY Brand;
```

(vii) Display the name and brand of watches that have been purchased in the year 2020:

```
SELECT WName, Brand FROM Watches WHERE YEAR(YOP) = 2020;
```

11. SQL—Working with Multiple Tables

Assessment

A.	1. c	2. c	3. b	4. a	
B.	1. True	2. True	3. False	4. False	5. False
C.	1. attributes	2. equality	3. foreign	4. primary	

D. 1. **Referential Integrity Constraints:** Referential integrity constraints ensure that a foreign key value always refers to an existing primary key value in another table.

Example:

An EMPLOYEE works in a DEPARTMENT, these two entities are related via Dept_No of the table EMPLOYEE, which refers to the primary key Dept_No of DEPARTMENT.

DBMS uses foreign keys to enforce the integrity of the database. For example, DBMS will disallow an attempt to insert a tuple in the EMPLOYEE table having a Dept_No that is not present in the DEPARTMENT table. Using SQL, we can specify Dept_No as a foreign key in the table EMPLOYEE and indicate that it references the primary key Dept_No of the table DEPARTMENT as follows:

```
FOREIGN KEY (Dept_No) REFERENCES DEPARTMENT (Dept_No)
CREATE TABLE EMPLOYEE
(
ID INT PRIMARY KEY,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(30),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL,
Dept_No SMALLINT,
    FOREIGN KEY(Dept_No) REFERENCES DEPARTMENT(Dept_No)
);
```

2. SQL supports a **NATURAL JOIN** operator that removes the duplicate column common to both the tables and positions the common attribute as the first column in the result. A **NATURAL JOIN** query to get the details of all the managers of all the departments may be formulated as follows:

   ```
   SELECT *
   FROM EMPLOYEE
   NATURAL JOIN DEPARTMENT;
   ```

3. Suppose we want to get complete details of all the departments' managers. The following SQL query joins the tables EMPLOYEE and DEPARTMENT to achieve this:

   ```
   SELECT *
   FROM DEPARTMENT, EMPLOYEE
   WHERE EMPLOYEE.Dept_No = DEPARTMENT.Dept_No;
   ```

   Note that the result of executing the above query yields the column Dept_No twice, once for each table. As the above query joins only those tuples which have identical values of Dept_No, such a join is called equijoin.

E. 1. a.
```
SELECT STUDENT.Roll_Num, SUBJECT.Subject_Name
FROM STUDENT
JOIN SUBJECT ON STUDENT.Subject_No = SUBJECT.Subject_No;
```

b.
```
SELECT STUDENT.Roll_Num, STUDENT.Subject_Name, SUBJECT.Dept_No
FROM STUDENT
JOIN SUBJECT ON STUDENT.Subject_No = SUBJECT.Subject_No;
```

c.
```
SELECT STUDENT.Roll_Num, STUDENT.Subject_Name
FROM STUDENT
JOIN SUBJECT ON STUDENT.Subject_No = SUBJECT.Subject_No
WHERE SUBJECT.Subject_Name = 'Chemistry';
```

d.
```
SELECT STUDENT.Subject_Name
FROM STUDENT
JOIN SUBJECT ON STUDENT.Subject_No = SUBJECT.Subject_No
```

```
   WHERE SUBJECT.Subject_Name = 'Physics';
```

e.
```
SELECT SUBJECT.Subject_Name, COUNT(STUDENT.Roll_Num) AS StudentCount
FROM SUBJECT
LEFT JOIN STUDENT ON SUBJECT.Subject_No = STUDENT.Subject_No
GROUP BY SUBJECT.Subject_Name;
```

f.
```
SELECT SUBJECT.Subject_Name, STUDENT.Roll_Num, STUDENT.Subject_Name
FROM SUBJECT
LEFT JOIN STUDENT ON SUBJECT.Subject_No = STUDENT.Subject_No;
```

2. a.
```
SELECT DEPARTMENT.Dept_Name, SUBJECT.Subject_Name
FROM DEPARTMENT
JOIN SUBJECT ON DEPARTMENT.Dept_No = SUBJECT.Dept_No
ORDER BY DEPARTMENT.Dept_Name, SUBJECT.Subject_Name;
```

b.
```
SELECT  DEPARTMENT.Dept_Name,  COUNT(SUBJECT.Subject_No)  AS
NumberOfSubjects
FROM DEPARTMENT
LEFT JOIN SUBJECT ON DEPARTMENT.Dept_No = SUBJECT.Dept_No
GROUP BY DEPARTMENT.Dept_Name;
```

c.
```
SELECT SUBJECT.Subject_Name, SUBJECT.Dept_No
FROM SUBJECT
WHERE SUBJECT.Subject_No IN (SELECT DISTINCT Subject_No FROM TEACHER);
```

3. a.
```
SELECT *
FROM PASSENGERS
WHERE TNO = 12030;
Output: |

+------+-------+------------+--------+-----+------------+
| PNR  | TNO   | PNAME      | GENDER | AGE | TRAVELDATE |
+------+-------+------------+--------+-----+------------+
| P004 | 12030 | S K SAXENA | MALE   | 42  | 2018-10-12 |
| P005 | 12030 | S SAXENA   | FEMALE | 35  | 2018-10-12 |
| P006 | 12030 | P SAXENA   | FEMALE | 12  | 2018-10-12 |
| P008 | 12030 | J K SHARMA | MALE   | 65  | 2018-05-09 |
| P009 | 12030 | R SHARMA   | FEMALE | 58  | 2018-05-09 |
+------+-------+------------+--------+-----+------------+
```

b.
```
SELECT COUNT(*) AS FemalePassengers
FROM PASSENGERS
WHERE TNO = 12030 AND GENDER = 'FEMALE';
```

```
Output:

+--------------------+
| FemalePassengers  |
+--------------------+
| 3                  |
+--------------------+
```

c.
```
SELECT P.PNAME, T.TNAME
FROM PASSENGERS P
JOIN TRAINS T ON P.TNO = T.TNO;
Output:

+--------------+-----------------+
| PNAME        | TNAME           |
+--------------+-----------------+
| R N AGRAWAL | Amritsar Mail   |
| P TIWARY     | Ajmer Shatabdi  |
| S TIWARY     | Ajmer Shatabdi  |
| S K SAXENA   | Swarna Shatabdi |
| S SAXENA     | Swarna Shatabdi |
| P SAXENA     | Swarna Shatabdi |
| N S SINGH    | Amritsar Mail   |
| J K SHARMA   | Swarna Shatabdi |
| R SHARMA     | Swarna Shatabdi |
+--------------+-----------------+
```

d.
```
SELECT DISTINCT TRAVELDATE
FROM PASSENGERS
JOIN TRAINS ON PASSENGERS.TNO = TRAINS.TNO
WHERE TRAINS.TNAME = 'Ajmer Shatabdi';
Output:

+-------------+
| TRAVELDATE |
+-------------+
| 2018-10-12 |
+-------------+
```

e.
```
Part a.
Output:

+-----------------+--------------+
| TNAME           | PNAME        |
+-----------------+--------------+
| Amritsar Mail   | R N AGRAWAL |
| Swarna Shatabdi | S K SAXENA   |
+--------------+-----------------+
Part b.

Output:

+--------+------------------+---------------+
| TNO    | TNAME            | NoOfBookings  |
+--------+------------------+---------------+
| 12015  | Ajmer Shatabdi   | 2             |
| 12030  | Swarna Shatabdi  | 5             |
| 13005  | Amritsar Mail    | 2             |
+--------+------------------+---------------+
```

4. (i) a.
```
SELECT SHIPMENT.OrderNo, PRODUCT.Pname
FROM SHIPMENT
JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo;
```
b.
```
SELECT SHIPMENT.OrderNo
FROM SHIPMENT
JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
WHERE PRODUCT.Brand = 'Whirlpool';
```
c.
```
SELECT DISTINCT PRODUCT.Pname
FROM SHIPMENT
JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
WHERE SHIPMENT.Price > 70000;
```
d.
```
SELECT PRODUCT.Pname, MIN(SHIPMENT.Price) AS MinPrice, SHIPMENT.
SNo
FROM SHIPMENT
JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
GROUP BY PRODUCT.Pname, SHIPMENT.SNo;
```
e.
```
SELECT PRODUCT.City, SUM(SHIPMENT.Quantity) AS TotalQuantity
FROM SHIPMENT
JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
GROUP BY PRODUCT.City;
```
(ii) a.
```
SELECT SUPPLIER.Sname
FROM SUPPLIER
JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
WHERE SHIPMENT.PNo = 201;
```
b.
```
SELECT SUPPLIER.Sname
FROM SUPPLIER
```

```
JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
GROUP BY SUPPLIER.Sname
HAVING COUNT(SHIPMENT.OrderNo) > 2;
```

c.
```
SELECT SUPPLIER.Sname, COUNT(SHIPMENT.OrderNo) AS NumberOfOrders
FROM SUPPLIER
JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
GROUP BY SUPPLIER.Sname;
```

d.
```
SELECT DISTINCT SUPPLIER.Sname
FROM SUPPLIER
JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
WHERE SUPPLIER.SCity = 'Mumbai' AND SHIPMENT.Price > 50000;
```

e.
```
SELECT DISTINCT SUPPLIER.Sname
FROM SUPPLIER
JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
WHERE SUPPLIER.SCity = 'Mumbai' AND SHIPMENT.Quantity > 40;
```

f.
```
SELECT SHIPMENT.*
FROM SHIPMENT
JOIN SUPPLIER ON SHIPMENT.SNo = SUPPLIER.SNo
WHERE SUPPLIER.SCity = 'Karnataka';
```

Assertion and Reasoning Based Questions

1. a 2. a

Case-based Questions

a.
```
SELECT Theatre.AUDI, Troupe.TNAME
FROM Theatre
JOIN Troupe ON Theatre.TROUPID = Troupe.TROUPID;
```

b.
```
SELECT Troupe.TNAME, Theatre.TITLE, Theatre.TICKET_PRICE
FROM Theatre
JOIN Troupe ON Theatre.TROUPID = Troupe.TROUPID
WHERE Theatre.AUDI = 1;
```

c.
```
SELECT LANGUAGE, COUNT(*) AS PlayCount
FROM Theatre
GROUP BY LANGUAGE;
```

d.
```
SELECT TNAME
FROM Troupe
ORDER BY TNAME;
```
e.
```
SELECT Theatre.TITLE, Troupe.TNAME
FROM Theatre
JOIN Troupe ON Theatre.TROUPID = Troupe.TROUPID
WHERE Troupe.CITY = 'Delhi'
ORDER BY Theatre.SHOWDATE;
```
f.
```
ALTER TABLE Troupe
ADD COLUMN CONTACTNO VARCHAR(10);
```

12. Python Interface with MySQL

Assessment

A.	1. b	2. a	3. c	4. d	5. a	6. d
	7. c	8. c				
B.	1. False	2. True	3. False	4. False	5. True	
C.	1. connect()	2. cursor()	3. SQL queries	4. format()	5. rowcount	

D. 1. Two methods to pass values of attributes to the SQL query in the execute() method are:

- Using **positional placeholders (%s)**: This method uses %s as a placeholder in the SQL query, and a tuple containing the values is passed as the second argument to the execute() method.
- Using **named placeholders (%(name)s)**: This method uses named placeholders in the SQL query, and a dictionary containing the key-value pairs for the attributes is passed as the second argument to the execute() method.

2. We may also use the format() method to provide attribute values in a tuple. The substitutions in the string are specified using curly braces '{' and '}', and the corresponding values to be substituted are passed as the comma-separated list of arguments. The resultant string also referred as formatted string can then be used as a query to be executed by execute() method as shown in following code snippet:

```
#Using placeholder for inserting user-provided attribute values into
the row
   print("Enter the Employee details:")
   aadhaar = input("ID: ")
```

```
        name = input("Name: ")
        gender = input("Gender: ")
        salary = int(input("Salary: "))
        dno = int(input("Department Number: "))
        query = "INSERT INTO EMPLOYEE VALUES('{}', '{}', '{}', {}, {});"\
                  .format(aadhaar, name, gender, salary, dno)
        company.execute(query)
```

3. Yes, the following two statements for inserting a row in STUDENT table are equivalent because the order of key-value pairs in the dictionary does not matter as long as the keys match the placeholders in the query.

4. **Reading from a CSV file:**

```
import csv
with open('fileName.csv', mode='r') as file:
    csvReader = csv.reader(file)
    for row in csvReader:
        # process each row
```

Writing to a CSV file:

```
import csv
with open('fileName.csv', mode='w', newline='') as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow(['column1', 'column2', 'column3'])  # writing header
    csvWriter.writerow([value1, value2, value3])  # writing rows
```

5.

```
import pymysql
# Establish connection to the database
def connectToDatabase():
    return pymysql.connect(
        host='localhost',
        user='your_username',
        password='your_password',
        database='SCHOOL'
    )
```

```
# Function to enter data into the TEACHER table
def enterData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    id = int(input("Enter Teacher ID: "))
    firstName = input("Enter First Name: ")
    lastName = input("Enter Last Name (or NULL if not applicable): ")
    if lastName.upper() == 'NULL':
        lastName = None
    subjectNo = int(input("Enter Subject No: "))
    contactNum = input("Enter Contact Number: ")
    salary = float(input("Enter Salary: "))
    emailID = input("Enter Email ID: ")

    sql = "INSERT INTO TEACHER (ID, First_Name, Last_Name, Subject_No,
    Contact_Num, Salary, Email_ID) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(sql, (id, firstName, lastName, subjectNo, contactNum,
    salary, emailID))
    connection.commit()
    print("Data entered successfully.")

    cursor.close()
    connection.close()

# Function to display data from the TEACHER table
def displayData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    sql = "SELECT * FROM TEACHER"
    cursor.execute(sql)
    results = cursor.fetchall()
    for row in results:
            print(row)

        cursor.close()
        connection.close()

# Function to update contact number of a teacher in the TEACHER table
def updateData():
        connection = connectToDatabase()
        cursor = connection.cursor()

        id = int(input("Enter Teacher ID to update: "))
        newContactNum = input("Enter new Contact Number: ")

        sql = "UPDATE TEACHER SET Contact_Num = %s WHERE ID = %s"
        cursor.execute(sql, (newContactNum, id))
        connection.commit()
        print("Data updated successfully.")

        cursor.close()
        connection.close()

# Function to delete data from the TEACHER table
def deleteData():
        connection = connectToDatabase()
        cursor = connection.cursor()

        id = int(input("Enter Teacher ID to delete: "))

        sql = "DELETE FROM TEACHER WHERE ID = %s"
        cursor.execute(sql, (id,))
        connection.commit()
        print("Data deleted successfully.")
```

```
    cursor.close()
    connection.close()

# Main function to display the menu and call appropriate functions
def main():
    while True:
        print("\nMenu:")
        print("1. ENTER Data")
        print("2. DISPLAY Data")
        print("3. UPDATE Data")
        print("4. DELETE Data")
        print("5. EXIT")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            enterData()
        elif choice == 2:
            displayData()
        elif choice == 3:
            updateData()
        elif choice == 4:
            deleteData()
        elif choice == 5:
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Notes:

- Replace 'your_username' and 'your_password' with your actual MySQL username and password.

- Ensure the MySQL server is running and the SCHOOL database along with TEACHER and SUBJECT tables exist before running the program.
- The user-defined functions enterData, displayData, updateData, and deleteData perform the respective operations.
- The main menu loop continues to prompt the user for input until they choose to exit by selecting option 5.

6.
```
import pymysql

# Establish connection to the database
def connectToDatabase():
    return pymysql.connect(
        host='localhost',
        user='your_username',
        password='your_password',
        database='INVENTORY'
    )

# Function to accept data into the SALE table
def acceptData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    billID = input("Enter Bill ID: ")
    custName = input("Enter Customer Name: ")
    itemPurchased = input("Enter Item Purchased: ")
    price = float(input("Enter Price: "))
    cgst = float(input("Enter CGST: "))

    sql = "INSERT INTO SALE (Bill_ID, Cust_Name, Item_Purchased, Price,
    CGST) VALUES (%s, %s, %s, %s, %s)"
   cursor.execute(sql, (billID, custName, itemPurchased, price, cgst))
    connection.commit()
    print("Data entered successfully.")
    cursor.close()
    connection.close()

# Function to display entire data along with total price
def displayData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    sql = "SELECT *, (Price + (CGST/100 * Price)) AS Total_Price FROM
    SALE"
    cursor.execute(sql)
    results = cursor.fetchall()

    print("+----------+------------+-----------------+---------+-------+
    ------------+")
    print("| Bill_ID  | Cust_Name | Item_Purchased | Price | CGST |
    Total_Price|")
    print("+----------+------------+-----------------+---------+-------+
    ------------+")

    for row in results:
        print("| {:<8} | {:<9} | {:<14} | {:<6} | {:<4} | {:<11}
        |".format(
            row[0], row[1], row[2], row[3], row[4], round(row[5], 2)))

    print("+----------+------------+-----------------+---------+-------+
    ------------+")

    cursor.close()
    connection.close()

# Function to update the price based on the Bill_ID
def updatePrice():
    connection = connectToDatabase()
    cursor = connection.cursor()

    billID = input("Enter Bill ID to update: ")
    newPrice = float(input("Enter new Price: "))

    sql = "UPDATE SALE SET Price = %s WHERE Bill_ID = %s"
    cursor.execute(sql, (newPrice, billID))
    connection.commit()
    print("Price updated successfully.")

    cursor.close()
    connection.close()

# Function to delete rows where Item_Purchased is 'Saree'
def deleteSaree():
    connection = connectToDatabase()
    cursor = connection.cursor()

    sql = "DELETE FROM SALE WHERE Item_Purchased = 'Saree'"
    cursor.execute(sql)
    connection.commit()
    print("Rows with 'Saree' as Item_Purchased deleted successfully.")

    cursor.close()
    connection.close()

# Main function to display the menu and call appropriate functions
def main():
    while True:
        print("\nMenu:")
        print("1. ACCEPT Data")
        print("2. DISPLAY Data")
        print("3. UPDATE Price")
        print("4. DELETE 'Saree' Data")
        print("5. EXIT")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            acceptData()
        elif choice == 2:
            displayData()
        elif choice == 3:
            updatePrice()
        elif choice == 4:
            deleteSaree()
        elif choice == 5:
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Notes:

- Replace 'your_username' and 'your_password' with your actual MySQL username and password.
- Ensure the MySQL server is running and the INVENTORY database along with the SALE table exists before running the program.
- The program includes user-defined functions acceptData, displayData, updatePrice, and deleteSaree to perform the respective operations.
- The main menu loop continues to prompt the user for input until they choose to exit by selecting option 5.

7.
```
import pymysql

# Function to establish a connection to the database
def connectToDatabase():
    return pymysql.connect(
        host='localhost',
            user='root',
            password='abc#123',
            database='SHIPMENTDB'
        )

# Function to create the tables SUPPLIER, PRODUCT, and SHIPMENT
def createTables():
    connection = connectToDatabase()
    cursor = connection.cursor()

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS SUPPLIER (
            SNo INT PRIMARY KEY,
            Sname VARCHAR(255),
            SCity VARCHAR(255)
        )
    """)

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS PRODUCT (
            PNo INT PRIMARY KEY,
            Pname VARCHAR(255),
            Brand VARCHAR(255),
            City VARCHAR(255)
        )
    """)

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS SHIPMENT (
            OrderNo INT PRIMARY KEY,
            SNo INT,
            PNo INT,
            Price DECIMAL(10, 2),
                Quantity INT,
                FOREIGN KEY (SNo) REFERENCES SUPPLIER(SNo),
                FOREIGN KEY (PNo) REFERENCES PRODUCT(PNo)
            )
        """)

        connection.commit()
        cursor.close()
        connection.close()

# Function to accept data and store it in the database
def acceptData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    # Accept SUPPLIER data
    suppliers = [
        (101, 'Aradhaya Pvt. L', 'Mumbai'),
        (102, 'XYZ Enterprises', 'Bangalore'),
        (103, 'Komal Enterpris', 'Delhi'),
        (104, 'Cloudtail', 'Patna'),
        (105, 'PQR Enterprises', 'Uttrakhand'),
        (106, 'Tech Solutions', 'Bangalore'),
        (107, 'XYZ Enterprises', 'Mumbai')
    ]
    cursor.executemany("INSERT INTO SUPPLIER VALUES (%s, %s, %s)", 
    suppliers)

    # Accept PRODUCT data
    products = [
        (201, 'Refrigerator', 'Whirlpool', 'Chennai'),
        (202, 'Washing Machine', 'Samsung', 'Kolkata'),
        (203, 'Television', 'Sony', 'Mumbai'),
    (204, 'Television', 'Samsung', 'Kolkata'),
    (205, 'Washing Machine', 'Whirlpool', 'Chennai'),
    (206, 'Refrigerator', 'Whirlpool', 'Delhi'),
    (207, 'Microwave Oven', 'IFB', 'Mumbai'),
    (208, 'Microwave', 'Samsung', 'Mumbai'),
    (209, 'Air Fryer', 'IFB', 'Chandigarh'),
    (210, 'Air Fryer', 'Philips', 'Lucknow')
]
cursor.executemany("INSERT INTO PRODUCT VALUES (%s, %s, %s, %s)",
products)

# Accept SHIPMENT data
shipments = [
    (1000, 101, 201, 50000.00, 50),
    (1001, 101, 202, 70000.00, 30),
    (1002, 102, 202, 90000.00, 75),
    (1003, 104, 205, 25000.00, 20),
    (1004, 105, 205, 55000.00, 30),
    (1005, 102, 207, 48000.00, 50),
    (1006, 105, 210, 23000.00, 30),
    (1007, 104, 209, 60000.00, 75),
    (1008, 102, 208, 65000.00, 20),
    (1009, 103, 207, 81000.00, 30),
    (1010, 106, 204, 56000.00, 30),
    (1011, 107, 210, 38000.00, 50),
    (1012, 101, 201, 72000.00, 50)
]
cursor.executemany("INSERT INTO SHIPMENT VALUES (%s, %s, %s, %s,
%s)", shipments)

connection.commit()
cursor.close()
connection.close()
```

```
# Function to retrieve and display all data from the database
def displayData():
    connection = connectToDatabase()
    cursor = connection.cursor()

    cursor.execute("SELECT * FROM SUPPLIER")
    suppliers = cursor.fetchall()
    print("\nSUPPLIER Table:")
    print("+-----+-----------------+-------------+")
    print("| SNo | Sname           | SCity       |")
    print("+-----+-----------------+-------------+")
    for row in suppliers:
        print("| {:<3} | {:<15} | {:<11} |".format(row[0], row[1],
        row[2]))
    print("+-----+-----------------+-------------+")

    cursor.execute("SELECT * FROM PRODUCT")
    products = cursor.fetchall()
    print("\nPRODUCT Table:")
    print("+-----+-----------------+-------------+------------+")
    print("| PNo | Pname           | Brand       | City       |")
    print("+-----+-----------------+-------------+------------+")
    for row in products:
        print("| {:<3} | {:<15} | {:<11} | {:<10} |".format(row[0],
        row[1], row[2], row[3]))
    print("+-----+-----------------+-------------+------------+")

    cursor.execute("SELECT * FROM SHIPMENT")
    shipments = cursor.fetchall()
    print("\nSHIPMENT Table:")
    print("+---------+-----+-----+----------+----------+")
    print("| OrderNo | SNo | PNo | Price    | Quantity |")
    print("+---------+-----+-----+----------+----------+")
    for row in shipments:
        print("| {:<7} | {:<3} | {:<3} | {:<8} | {:<8} |".format(row[0],
        row[1], row[2], row[3], row[4]))
    print("+---------+-----+-----+----------+----------+")

    cursor.close()
    connection.close()

# Function to generate reports
def generateReports():
    connection = connectToDatabase()
    cursor = connection.cursor()

    # Report a: Order number and name of the product supplied in each
    shipment
    cursor.execute("""
        SELECT SHIPMENT.OrderNo, PRODUCT.Pname
        FROM SHIPMENT
        JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
    """)
    result = cursor.fetchall()
    print("\nReport a: Order number and name of the product supplied
    in each shipment")
    print("+---------+-----------------+")
    print("| OrderNo | Product Name    |")
    print("+---------+-----------------+")
    for row in result:
        print("| {:<7} | {:<15} |".format(row[0], row[1]))
    print("+---------+-----------------+")

    # Report b: Order number of the products of Whirlpool brand
    cursor.execute("""
        SELECT SHIPMENT.OrderNo
        FROM SHIPMENT
    JOIN PRODUCT ON SHIPMENT.PNo = PRODUCT.PNo
    WHERE PRODUCT.Brand = 'Whirlpool'
""")
result = cursor.fetchall()
print("\nReport b: Order number of the products of Whirlpool brand")
print("+----------+")
print("| OrderNo |")
print("+----------+")
for row in result:
    print("| {:<7} |".format(row[0]))
print("+----------+")

# Report c: Names of suppliers who have supplied more than two
orders
cursor.execute("""
    SELECT SUPPLIER.Sname
    FROM SUPPLIER
    JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
    GROUP BY SUPPLIER.SNo
    HAVING COUNT(SHIPMENT.OrderNo) > 2
""")
result = cursor.fetchall()
print("\nReport c: Names of suppliers who have supplied more than
two orders")
print("+------------------------+")
print("| Supplier Name          |")
print("+------------------------+")
for row in result:
    print("| {:<22} |".format(row[0]))
print("+------------------------+")

# Report d: Number of orders supplied by each supplier
cursor.execute("""
            SELECT SUPPLIER.Sname, COUNT(SHIPMENT.OrderNo) AS OrderCount
            FROM SUPPLIER
            JOIN SHIPMENT ON SUPPLIER.SNo = SHIPMENT.SNo
            GROUP BY SUPPLIER.SNo
        """)
        result = cursor.fetchall()
        print("\nReport d: Number of orders supplied by each supplier")
        print("+------------------------+--------------+")
        print("| Supplier Name          | Order Count |")
        print("+------------------------+--------------+")
        for row in result:
            print("| {:<22} | {:<11} |".format(row[0], row[1]))
        print("+------------------------+--------------+")

        cursor.close()
        connection.close()

    # Main function to display the menu and call appropriate functions
    def main():
        createTables()
        acceptData()

        while True:
            print("\nMenu:")
            print("1. DISPLAY Data")
            print("2. GENERATE Reports")
            print("3. EXIT")
            choice = int(input("Enter your choice: "))

            if choice == 1:
                displayData()
            elif choice == 2:
                generateReports()
        elif choice == 3:
            break
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

Note: Replace 'root' and 'abc#123' with the appropriate username and password for your MySQL database. Also, ensure the MySQL server is running and accessible.

Assertion and Reasoning Based Questions

1. a 2. a

Case-based Questions

1.
```
import pymysql


def createConnection():
    connection = pymysql.connect(
        host="HOST1",
        user="ORANGE",
        password="education@1",
        database="YOUR_DATABASE_NAME"   # Replace with your database
        name
    )
    return connection


def createTable():
    connection = createConnection()
    cursor = connection.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Client (
            clientId INT PRIMARY KEY,
            cName VARCHAR(100) NOT NULL,
            city VARCHAR(100) NOT NULL,
                contactNo VARCHAR(15)
            )
        """)
        connection.close()

    def insertClientData():
        connection = createConnection()
        cursor = connection.cursor()
        clientId = int(input("Enter Client ID: "))
        cName = input("Enter Client Name: ")
        city = input("Enter Client City: ")
        contactNo = input("Enter Client Contact Number: ")

        sql = "INSERT INTO Client (clientId, cName, city, contactNo) VALUES
        (%s, %s, %s, %s)"
        values = (clientId, cName, city, contactNo)
        cursor.execute(sql, values)
        connection.commit()
        connection.close()
        print("Client data inserted successfully.")

    def displayClientsInDelhi():
        connection = createConnection()
        cursor = connection.cursor()
        sql = "SELECT * FROM Client WHERE city = 'Delhi'"
        cursor.execute(sql)
        rows = cursor.fetchall()

        if rows:
            print("Clients in Delhi:")
            for row in rows:
                print(f"Client ID: {row[0]}, Name: {row[1]}, City: {row[2]},
                Contact No: {row[3]}")
    else:
        print("No clients found in Delhi.")
    connection.close()

def mainMenu():
    createTable()
    while True:
        print("\nMain Menu")
        print("1. Insert Client Data")
        print("2. Display Clients in Delhi")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            insertClientData()
        elif choice == '2':
            displayClientsInDelhi()
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    mainMenu()
```

Replace YOUR_DATABASE_NAME with the actual name of the database where the Client table will be created.

2.
```
import pymysql

def createConnection():
    connection = pymysql.connect(
        host="HOST1",
        user="ORANGE",
        password="education@1",
        database="Adventure"
    )
    return connection

def updateCityForTroupe():
    connection = createConnection()
    cursor = connection.cursor()
    troupeId = input("Enter Troupe ID: ")
    newCity = input("Enter New City: ")

    sql = "UPDATE Troupe SET CITY = %s WHERE TROUPID = %s"
    values = (newCity, troupeId)
    cursor.execute(sql, values)
    connection.commit()
    connection.close()
    print("City updated successfully for Troupe ID:", troupeId)

def main():
    updateCityForTroupe()

if __name__ == "__main__":
    main()
```

Make sure the Adventure database and Troupe table already exist with the appropriate schema before running this program.