Part A – Python Programs

Chapter 1 - Python Revision Tour

1. Write a program to find the sum and average of list in python.

Code:

```
#Creating list with user defined list
#Declare an empty list
l=[]

while True:
    #Input through user
    n=int(input("Enter no. to add into the list:"))
    #Add element into the end of list
    l.append(n)

    #Condition to break loop by pressing X
    ch=input("Press 'X' to stop:")
    if ch in 'xX':
        break

#Variable for sum and length
s=0
n=len(l)
for i in l:
    #Computing Sum of elements
    s+=i

#Computing average
avg=s/n

#Output
print("The list is:",l)
print("Sum is:",s)
print(f"Average is:{avg:.2f}")
```

Output:

```
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\
average.py
Enter no. to add into the list:24
Press any key for more | Press 'X' to stop:f
Enter no. to add into the list:47
Press any key for more | Press 'X' to stop:g
Enter no. to add into the list:21
Press any key for more | Press 'X' to stop:e
Enter no. to add into the list:64
Press any key for more | Press 'X' to stop:x
The list is: [24, 47, 21, 64]
Sum is: 156
Average is:39.00
```

2. Write a program to remove duplicates from the dictionary

Code:

```
#A program to remove dulicates from the dictionary

#Creating empty dictionary
d={}
#Ask user to enter number of key:value pairs
n=int(input("Enter no. of elements:"))

for i in range(n):
    k=input("Enter Key:")
    v=input("Enter value:")
    d[k]=v

#Declare empty dictionary to store unique key-value pairs
uni_dict={}

#Traverse dictionary to remove duplicates and store into unique dictionary
for k in d:
    val=d[k]
    if val not in uni_dict.values():
        uni_dict[k]=val

#Output
print("Dictionary After removing duplicates:",uni_dict)
```

Output:

```
Enter no. of elements:4
Enter Key:Jan
Enter value:31
Enter Key:Feb
Enter value:29
Enter Key:Mar
Enter value:31
Enter Key:Apr
Enter value:30
Dictionary After removing duplicates: {'Jan': '31', 'Feb': '29', 'Apr': '30'}
```

Chapter 2 – Working with functions

3. Write a program to define a function to accept two integers m and n respectively as argument and display prime numbers between them.

Code:

```
#A program to display prime numbers from a range
# Function to check and display prime in the range
def prime_m_to_n(m,n):
    for i in range(m, n + 1):
        # all prime numbers are greater than 1
        if i > 1:
            for j in range(2, i):
                if (i % j) == 0:
                    break
            else:
                print(i, end=' ')

# Taking input for m and n
m = int(input("Enter the value of m: "))
n = int(input("Enter the value of n: "))

# Calling the function
prime_m_to_n(m, n)
```

Output:

```
>>>
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File/
ion prime bumbers.py
Enter the value of m: 10
Enter the value of n: 30
11 13 17 19 23 29
```

4. Write a program to compute interest based on different arguments given in the function.

Code:

```
# Function to calculate interest
def compute_int(pri, r=5, t=1, comp=False):

    """Calculates simple or compound interest.
    Parameters:
    pri: Principal amount
    r : Interest rate (default is 5%)
    t: Time period in years (default is 1 year)
    comp : If True, calculates compound interest, otherwise simple interest (default is False)"""

    if comp:
        """Compound Interest formula: A = P(1 + r/n)^(nt),
         where n = 1 for annual compounding"""
        amt = pri * (1 + r / 100) ** t
        i = amt - pri
    else:
        # Simple Interest formula: SI = (P * R * T) / 100
        i = (pri * r * t) / 100
    return i

# Testing the function with various argument combinations
p=int(input("Enter Principle:"))
r=float(input("Enter custom rate:"))
t=float(input("Enter time:"))

# Case 1: Only principal provided, default rate and time
print("Interest (Simple, default):", compute_int(1000))

# Case 2: Principal and rate provided, default time
print("Interest (Simple, custom rate):", compute_int(p, r))
```

```
# Case 3: Principal, rate, and time provided
print("Interest (Simple, custom rate and time):", compute_int(p, r, t))

# Case 4: Compound interest, with custom rate and time
print(f"Interest (Compound, custom rate and time): {compute_int(p, r, t, True):.2f}")
```

Output:

```
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\
est Calculation.py
Enter Principle:5000
Enter custom rate:10
Enter time:3
Interest (Simple, default): 50.0
Interest (Simple, custom rate): 500.0
Interest (Simple, custom rate and time): 1500.0
Interest (Compound, custom rate and time): 1655.00
```

5. Write a program to largest element in a list using a function. Pass list object as argument.

Code:

```
# Function to find the largest element in a list
def find_largest(l):
    # Check the list is empty or not
    if not l:
        # Return None if the list is empty
        return None

    # Assume the first element is the large
    large = l[0]
    for i in l:
        # Check if other element than first is large
        if i > large:
            # Update the largest if a bigger number is found
            large = i
    return large

# Taking input as a list
lst = eval(input("Enter list as input:"))

# Calling the function and printing the result
large_ele = find_largest(lst)
if large_ele is not None:
    print(f"The largest element in the list is: {large_ele}")
else:
    print("The list is empty.")
```

Output:

```
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File
st from list.py
Enter list as input:[34,56,78,32,11]
The largest element in the list is: 78
```

6. Write a program to count the number of vowels in passed string to the function.

Code:

```
# A program to count the number of vowels in passed string to the function
def count_vowels(s):
    # Define a set of vowels (both lowercase and uppercase)
    vowels = "aeiouAEIOU"

    # Initialize a counter variable
    count = 0

    # Loop through each character in the string
    for char in s:
        if char in vowels:
            count += 1
    return count

# Passing string and calling function
input_string = input("Enter a string: ")
print("Number of vowels:", count_vowels(input_string))
```

```
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File
s in string.py
Enter a string: Hello this is string
Number of vowels: 5
```

Chapter 3 – Exception Handling

7. Write a program to take two integers and perform addition of two numbers handle exception when non-integers entered by user.

Code:

```
# A program to raise exception when a negative number is entered
# Define a function to add and return error
def add(n1,n2):
    #Checking for error
    if n1<0 or n2<0:
        raise ValueError("Negative number entered!")
    else:
        #Performing Addition
        return n1 + n2

#Handling exception using try
try:
    no1 = int(input("Enter number1: "))
    no2 = int(input("Enter number2: "))
    print(f"The result is: {add(no1,no2)}")
except ValueError as e:
    print(f"Error: {e}")
```

Output:

```
>>>
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\
on Program 2.py
Enter number1: 8
Enter number2: -9
Error: Negative number entered!
>>>
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\
on Program 2.py
Enter number1: -8
Enter number2: 5
Error: Negative number entered!
>>>
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\
on Program 2.py
Enter number1: 42
Enter number2: 9
The result is: 51
```

8. Write a program to find the square root of a positive number, raise a custom exception when user enters a non-integer number

Code:

```
# Importing math module
import math

#Define a function
def find_square_root():
    # Handling exception
    try:
        # Input a number
        num = float(input("Enter a number: "))
        # Condition to check positive integer
        if num <= 0 or num % 1 != 0:
            raise ValueError("Input must be a positive integer only")
        # Calculating Square root
        square_root = math.sqrt(num)
        # Output
        print(f"The square root of {num} is {square_root}")
    #Generating Error message
    except ValueError as e:
        print(f"Custom Error: {e}")

find_square_root()
```

Output:

```
>>>
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File/
tion Program 2.py
Enter a number: 5.65
Custom Error: Input must be a positive integer
>>>
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File/
tion Program 2.py
Enter a number: 25
The square root of 25.0 is 5.0
```

Chapter 4 – File Handling

9. Write a program to replace entered word from a text file india.txt.

Code:

```
# A program to replace entered word from a text file

# Define a function
def replace_word(f, old_word, new_word):
    # Open a file to read
    with open(f, 'r') as file:
        # Store data read from file indt object
        dt = file.read()

    # Checking the presence of old word and generating appropriate message
    if old_word not in dt:
            print(f"The word '{old_word}' was not found in the file.")
            return
    # Replace old word with new word
    new_dt = dt.replace(old_word, new_word)

    # Writing new data in the file
    with open(f, 'w') as file:
            file.write(new_dt)
    #Output
    print()
    print(f"The word '{old_word}' has been replaced with '{new_word}'
successfully.")
    print()
    print(new_dt)

# Main Code
f = 'india.txt'
old_word = input("Enter the word to be replaced: ")
new_word = input("Enter the new word: ")
replace_word(f, old_word, new_word)
```

Output:

```
>>>
=== RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/Practical File/9 File handling 1.py ===
Enter the word to be replaced: India
Enter the new word: Bharat

The word 'India' has been replaced with 'Bharat' successfully.

Bharat stands as a beacon of cultural diversity and unity, with a heritage that spans millennia. It's
a land where ancient traditions coexist with modernity, where the spirit of democracy thrives. With i
ts rich history of literature, arts, and sciences, Bharat has made countless contributions to the glo
bal stage. The nation's resilience is evident in its journey from colonial rule to becoming one of th
e world's largest economies.

Bharat's scenic landscapes, from the Himalayas to the beaches of Goa, attract millions of tourists ea
ch year. The country's culinary diversity is a feast for the senses, offering flavors that range from
the spicy curries of the south to the rich sweets of the north. Festivals like Diwali, Holi, and Eid
showcase Bharat's vibrant spirit and communal harmony.

Moreover, Bharat's progress in technology and space exploration, epitomized by missions like Chandray
aan and Mangalyaan, reflect its forward-thinking vision. As the world's largest democracy, Bharat's p
luralistic society exemplifies the strength in unity. Pride in Bharat is not just in its past glory,
but in its dynamic present and promising future.
```

10. Write a menu driven program to store and manipulate data in binary file to insert, update, delete and display records. The data has following structure:

Data Represents	Patient Information
Dictionary	{'P_ID':101,'P_Name':'Shiv','Charges':25000}
File Name	patient.dat

Menu:

1. Add Patient
2. Display Patient
3. Search Patient
4. Update Patient
5. Delete Patient
6. Exit

Code:

```
# import module to handle binary files data
import pickle
# A variable to mark as record found to counter loop iteration
flag = False

# Accepting data for Dictionary and dumping into binary file
def add_pat():
    # Enter data as Input to add in dictionary
    P_ID = int(input('Enter patient_id:'))
    pname = input('Enter Patient Name:')
    charges = int(input('Enter Charges:'))

    # Creating the dictionary
    rec = {'P_ID':P_ID,'P_Name':pname,'Charges':charges}

    # Opens a file and Writing the Dictionary
    f = open('patient.dat','ab')
    pickle.dump(rec,f)
    f.close()
    print("Record Addedd successfully.")
```

```
# Reading and displaying the records
def display_pat():
    #Opening a binary file and reading data individually
    f = open('patient.dat','rb')
    while True:
        try:
            rec = pickle.load(f)
            print('*'*40)
            print('Patient ID:',rec['P_ID'])
            print('Name:',rec['P_Name'])
            print('charges:',rec['Charges'])
            print('*'*40)
        except EOFError:
            break
    f.close()

#Searching a record based on patient name
def search_pat(pname):
    # Open file to search patient record
    f = open('patient.dat','rb')

    # Iterating records using while loop
    while True:
        # Hnalding exception Ran out of Input and EOFError
        try:
            # Load records into a python object
            rec = pickle.load(f)
            # Matching and displaying records
            if rec['P_Name'] == pname:
                print('Patient ID:',rec['P_ID'])
                print('Name:',rec['P_Name'])
                print('Charges:',rec['Charges'])
```

```
            flag = True
        except EOFError:
            break

    if flag == False:
        print('No Record found...')
    f.close()



# charges modification for a P_ID
def update_pat():
    # Prompt P_ID and charges to modify record
    pid = int(input('Enter a P_ID:'))
    flag=False
    # Open a file to verify record exists or not
    f = open('patient.dat','rb')

    # An empty list to store the record found for the update
    reclst = []
    while True:
        try:
            rec = pickle.load(f)
            reclst.append(rec)
            flag=True
        except EOFError:
            break
    f.close()

    c = int(input('Enter new Charges:'))
    for i in range (len(reclst)):
        if reclst[i]['P_ID']==pid:
            reclst[i]['Charges'] = c
    f = open('patient.dat','wb')
```

```
for i in reclst:
        pickle.dump(i,f)
    f.close()
    print("Record modified successfully...")

# Deleting a record based on P_ID
def deleteRec():
    flag=False
    f = open('patient.dat','rb')
    pid = int(input('Enter a P_ID:'))
    reclst = []
    while True:
        try:
            rec = pickle.load(f)
            reclst.append(rec)
        except EOFError:
            break
    f.close()
    f = open('patient.dat','wb')
    for i in reclst:
        if i['P_ID']==pid:
            continue
            pickle.dump(i,f)
            print("Record Deleted...")
        f.close()

while True:
    print('''
1. Add Patient
2. Display Patient
3. Search Patient
4. Update Patient
5. Delete Patient
```

```
6. Exit
''')
    ch = int(input('Enter your choice:'))
    if ch == 1:
        add_pat()
    elif ch == 2:
        display_pat()
    elif ch == 3:
        pn = input('Enter a patient name to search:')
        search_pat(pn)
    elif ch == 4:
        update_pat()
    elif ch == 5:
        deleteRec()
    elif ch==6:
        print("Bye Bye, Thank you for Interactions")
        break
    else:
        print("Invalid Choice")
```

Output:

Main Menu

```
>>>
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\Practical File\10 binary file
og.py

1. Add Patient
2. Display Patient
3. Search Patient
4. Update Patient
5. Delete Patient
6. Exit

Enter your choice:
```

Add Patient

```
Enter your choice:1
Enter patient_id:111
Enter Patient Name:Tejas
Enter Charges:6500
Record Addedd successfully.
```

Display Patient

```
Enter your choice:2
****************************************
Patient ID: 101
Name: Krishna
charges: 24500
****************************************
****************************************
Patient ID: 102
Name: Sudama
charges: 45000
****************************************
****************************************
Patient ID: 111
Name: Tejas
charges: 6500
****************************************
```

Search Patient

```
Enter your choice:3
Enter a patient name to search:Tejas
Patient ID: 111
Name: Tejas
Charges: 6500
```

Update Patient

```
Enter your choice:4
Enter a P_ID:111
Enter new Charges:7000
Record modified successfully...
```

Delete Patient

```
Enter your choice:5
Enter a P_ID:102
Record Deleted...
```

Exit

```
Enter your choice:6
Bye Bye, Thank you for Interactions
```

11. Create a CSV file by entering user-id and password, read and search the password for given userid.

Code:

```
# importing csv file
import csv
# creating a list to add record
users=[]

# define function to add user
def add_user():
    # Taking input username and password
    un=input("Enter Username:")
    pwd=input("Enter Password:")
    # Creating csv file and opening csv file in writing mode
    f=open("user.csv","a",newline='')
    # Creating writer object
    w=csv.writer(f)
    # Creating a row with username and password
    w.writerow([un,pwd])
    # Closing the file
    f.close()
    print("User added...")

# Defining a seach function
def search():
    # Open a file to search password
    f=open("user.csv","r")
    # Reading data from csv file
    r=csv.reader(f)
    # Enter username to search password
    un=input("Enter Username to search:")
    # Traversing through records
    for i in r:
        # Verifying usernames and displaying password
```

```
        if i[0]==un:
            print("Your password for ", i[0], " is:",i[1])
    f.close()


# Creating main menu
while True:
    print('''
    1. Add user
    2. Search Password
    3. Exit
    ''')
    ch=int(input("Enter Your Choice:"))
    if ch==1:
        add_user()
    elif ch==2:
        search()
    elif ch==3:
        break
    else:
        print("Invalid Choice")
```

Output:

Main Menu

```
>>>
= RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\
le\11 Csv File Program.py

    1. Add user
    2. Search Password
    3. Exit

Enter Your Choice:
```

Add User

```
    1. Add user
    2. Search Password
    3. Exit

Enter Your Choice:1
Enter Username:Administrator
Enter Password:Admin@123
User added...
```

Search Password

```
    1. Add user
    2. Search Password
    3. Exit

Enter Your Choice:2
Enter Username to search:Administrator
Your password for  Administrator  is: Admin@123
```

12. Write a program to create a binary file candidates.dat which stores the following information in list as follows:

Structure of Data

Field	Data Type
Candidate_id	Integer
Candidate_Name	String
Designation	String
Experience	Float

(i)Write a function append() to write data into binary file.

(ii) Write a function display() to display data from binary file for all candidates whose experience is more than 10.

Code:

```
# Module to work with binary files
import pickle
# A function to accept data and append into binary file
def append():
    with open("Candidates.dat",'ab') as f:
        C_id=int(input("Enter Candidate ID: "))
        C_nm=input("Enter Candidate name: ")
        C_dg=input("Enter Designation: ")
        C_ex=float(input("Enter Experience: "))
        rec=[C_id,C_nm,C_dg,C_ex]
        pickle.dump(rec,f)
```

```
# Display data after reading file contents
def display():
    with open("Candidates.dat",'rb') as f:
        while True:
            try:
                rec=pickle.load(f)
                if rec[-1]>10:
                    print(rec)
            except EOFError:
                break
# Calling functions
append()
display()
```

Output:

```
>>>
    = RESTART: D:\Website Content\TutorialAICSIP\2024-25\
    le\12 Binary File Program.py
    Enter Candidate ID: 101
    Enter Candidate name: Dhaval
    Enter Designation: Manager
    Enter Experience: 12
    [101, 'Dhaval', 'Manager', 12.0]
>>>
```

13. Write a program to generate a CSV file named happiness.csv. Each record of CSV contains these data:

Name of the country	String
Population of country	Integer
No. people participated in survey	Integer
No. of people who are Happy	Integer

Write user defined functions to do the following:

(i) Insert records into CSV

(ii) Display records from CSV

(iii) Update records

(iv) Delete record

Code:

```
# Import a module to work with csv
import csv

# a list object to store records
h=[]

# A function to create header row, run only once
def header_row():
    f=open("happiness.csv",'w',newline='')
    wo=csv.writer(f)
    wo.writerow(['Country','Population','Participants', 'Happy People'])
    print("A CSV file with header is created...")
    f.close()

# A function to insert a record
def insert_rec():
    with open('happiness.csv','a',newline='') as f:
        country=input("Enter name of country:")
        population=float(input("Enter Population (In Billion):"))
```

```
        participants=float(input("Enter no. of participants (In Million):"))
        happy=float(input("Enter no. of people who are happy (In Million):"))
        h=[country, population, participants, happy]
        wo=csv.writer(f)
        wo.writerow([country,population,participants,happy])
        print("Record inserted...")

# A function to display records
def display_rec():
    with open('happiness.csv','r',newline='') as f:
        ro=csv.reader(f)
        for i in ro:
            print(i)

# Update a record
def update_rec():
    c=input("Enter country to update records:")
    updated_row=[]
    flg=False
    with open('happiness.csv','r') as f:
        ro=csv.reader(f)
        for i in ro:
            if i[0]==c:

                flg=True
                population=float(input("Enter Population (In Billion):"))
                participants=float(input("Enter no. of participants (In Million):"))
                happy=float(input("Enter no. of people who are happy (In Million):"))
                i[1]=population

                i[2]=participants

                i[3]=happy

            updated_row.append(i)
        if flg==True:
```

```
            with open('happiness.csv','w',newline='') as f:
                wo=csv.writer(f)
                wo.writerows(updated_row)
                print('Record Updated...')
        else:
            print('Record not found...')

# Delete Record
def delete_rec():
    c=input("Enter country to update records:")
    deleted_row=[]
    flg=False
    with open('happiness.csv','r') as f:
        ro=csv.reader(f)
        for i in ro:
            if i[0] != c:
                flg=True
                deleted_row.append(i)
        if flg==True:
            with open('happiness.csv','w',newline='') as f:
                wo=csv.writer(f)
                wo.writerows(deleted_row)
                print('Record Deleted...')
        else:
            print('Record not found...')

# Main Menu
while True:
    print('''
1. Create header row (Excute at once)

2. Insert country record

3. Display happiness record
```

```
4. Update Record

5. Delete Record

6. Exit

''')
    ch = int(input('Enter your choice:'))
    if ch==1:

        header_row()
    elif ch == 2:

        insert_rec()
    elif ch == 3:

        display_rec()
    elif ch == 4:

       update_rec()
    elif ch == 5:

        delete_rec()
    elif ch == 6:

        print("Bye Bye, Thank you for Interactions")
        break
    else:
        print("Invalid Choice")
```

Output: Main Menu

```
>>>
    = RESTART: D:\Website Content\TutorialAICSIP\2024-25\CS XII\
    le\13 CSV file program to update and delete.py

    1. Create header row (Excute at once)
    2. Insert country record
    3. Display happiness record
    4. Update Record
    5. Delete Record
    6. Exit

    Enter your choice:
```

Create header row (Execute at Once)

```
Enter your choice:1
A CSV file with header is created...
```

Insert Country Record

```
Enter your choice:2
Enter name of country:India
Enter Population (In Billion):1.6
Enter no. of participants (In Million):5.5
Enter no. of people who are happy (In Million):1.2
Record inserted...
```

Display Record

```
Enter your choice:3
['Country', 'Population', 'Participants', 'Happy People']
['India', '1.6', '5.5', '1.2']
```

Update Record

```
Enter your choice:4
Enter country to update records:India
Enter Population (In Billion):1.6
Enter no. of participants (In Million):5.5
Enter no. of people who are happy (In Million):1.4
Record Updated...
```

Delete record

```
Enter your choice:5
Enter country to delete records:USA
Record Deleted...
```

14. Write a menu drive program for stack operations:

Code:

```
def check_stack_isEmpty(stk):
    if stk==[]:
        return True
    else:
        return False
```

```
s=[] # An empty list to store stack elements, initially its empty
top = None # This is top pointer for push and pop operation
def main_menu():
    while True:
        print("Stack Implementation")
        print("1 - Push")
        print("2 - Pop")
        print("3 - Peek")
        print("4 - Display")
        print("5 - Exit")
        ch = int(input("Enter the your choice:"))
        if ch==1:
            el = int(input("Enter the value to push an element:"))
            push(s,el)
        elif ch==2:
            e=pop_stack(s)
            if e=="UnderFlow":
                print("Stack is underflow!")
            else:
                print("Element popped:",e)
        elif ch==3:
            e=pop_stack(s)
            if e=="UnderFlow":
                print("Stack is underflow!")
            else:
                print("The element on top is:",e)
        elif ch==4:
            display(s)
        elif ch==5:
            break
        else:
            print("Sorry, You have entered invalid option")
```

```
def push(stk,e):
    stk.append(e)
    top = len(stk)-1
def display(stk):
    if check_stack_isEmpty(stk):
        print("Stack is Empty")
    else:
        top = len(stk)-1
        print(stk[top],"-Top")
        for i in range(top-1,-1,-1):
            print(stk[i])
def pop_stack(stk):
    if check_stack_isEmpty(stk):
        return "UnderFlow"
    else:
        e = stk.pop()
        if len(stk)==0:
            top = None
        else:
            top = len(stk)-1
        return e
def peek(stk):
    if check_stack_isEmpty(stk):
        return "UnderFlow"
    else:
        top = len(stk)-1
        return stk[top]
main_menu()
```

Output:

Main Menu

```
>>>
= RESTART: D:\Website Content\TutorialAICSIP\
le\14 stack operations.py
Stack Implementation
1 - Push
2 - Pop
3 - Peek
4 - Display
5 - Exit
```

Push Element

```
Enter the your choice:1
Enter the value to push an element:11
An element pushed into the stack.
```

Pop Element

```
Enter the your choice:2
Element popped: 44
```

Peek Element

```
Enter the your choice:3
The element on top is: 32
```

Display Stack

```
Enter the your choice:4
74 -Top
96
50
11
```

15. Write a program to Push an item into stack where a dictionary contains the details of stationary items as follows:
stn = {"Shirt":700,"T-Shirt":750,"Jeans":1150,"Trouser":400}

(a) Write a function to push an element into stack for those items names whose price is more than 850. Also display the count of elements pushed into the stack.

(b) Write a function to pop an element to remove the element and display appropriate message when the stack is empty.

Code:

```
stn = {}
while True:
    item=input("Enter itemname:")
    price=float(input("Enter Price:"))
    stn[item]=price
    ch=input("Press X to stop:")
    if ch in 'xX':
        break

def Push(stk,ele):
    stk.append(ele)
    return stk

def Pop(stk):
    if stk==[]:
        return "underflow"
    else:
        return stk.pop()
stak=[]
c=0
for i in stn:
    if stn[i]>35:
        Push(stak,i)
```

```
        c+=1
print("Total Stack elements are:",c)


while True:
    if stak!=[]:
        print(Pop(stak))
    else:
        print("Stack is Empty")
        break
```

Output:

```
>>>
    = RESTART: D:\Website Content\TutorialAICSIP
    le\15 Stack Program 2.py
    Enter itemname:Shirt
    Enter Price:550
    Press X to stop:d
    Enter itemname:T-Shirt
    Enter Price:750
    Press X to stop:d
    Enter itemname:Jeans
    Enter Price:480
    Press X to stop:x
    Total Stack elements are: 3
    Jeans
    T-Shirt
    Shirt
    Stack is Empty
```

Part B – MySQL Queries

Chapter 11 – Simple Queries in SQL

Set 1 – Based on Database Basic Commands

Write SQL Commands for following:

1. Create a database named practical_2025

Ans.: mysql> CREATE DATABASE practical_2025;

```
Query OK, 1 row affected (0.05 sec)
```

2. Check database is present or not in the list of databases

Ans.: Mysql> show databases;

```
+--------------------+
| Database           |
+--------------------+
| cent_sorcerer      |
| class12            |
| db1                |
| information_schema |
| itemdb             |
| mysql              |
| performance_schema |
| practical_2025     |
| sakila             |
| sports             |
| sys                |
| world              |
+--------------------+
12 rows in set (0.08 sec)
```

3. Open a database

Ans.: mysql>use practical_2025

```
Database changed
mysql>
```

4. Create a database temp

Ans.: mysql> create database temp;

```
Query OK, 1 row affected (0.05 sec)
```

5. Delete a database temp

Ans.: drop database temp;

```
Query OK, 0 rows affected (0.04 sec)
```

Set 2 – Based on DDL Commands

Consider the following MOVIES table and write the SQL commands based on it.

Movie_ID	MovieName	Type	ReleaseDate	ProductionCost	BusinessCost
M001	Dahek	Action	2022/01/26	1245000	1300000
M002	Attack	Action	2022/01/28	1120000	1250000
M003	Looop Lapeta	Thriller	2022/02/01	250000	300000
M004	Badhai Do	Drama	2022/02/04	720000	68000
M005	Shabaash Mithu	Biography	2022/02/04	1000000	800000
M006	Gehraiyaan	Romance	2022/02/11	150000	120000

1. Create above table, assign Movie_ID as a primary key and insert records as above

```
mysql> CREATE TABLE Movies ( Movie_ID VARCHAR(5) PRIMARY KEY,
MovieName VARCHAR(50),
Type VARCHAR(20),
ReleaseDate DATE,
ProductionCost INT,
BusinessCost INT );
    -> );
Query OK, 0 rows affected (0.08 sec)
```

```
INSERT INTO Movies (Movie_ID, MovieName, Type, ReleaseDate,
ProductionCost, BusinessCost) VALUES
('M001', 'Dahek', 'Action', '2022-01-26', 1245000, 1300000),
('M002', 'Attack', 'Action', '2022-01-28', 1120000, 1250000),
('M003', 'Looop Lapeta', 'Thriller', '2022-02-01', 250000, 300000),
('M004', 'Badhai Do', 'Drama', '2022-02-04', 720000, 68000),
('M005', 'Shabaash Mithu', 'Biography', '2022-02-04', 1000000, 800000),
('M006', 'Gehraiyaan', 'Romance', '2022-02-11', 150000, 120000);
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
+----------+---------------+-----------+-------------+----------------+--------------+
| Movie_ID | MovieName     | Type      | ReleaseDate | ProductionCost | BusinessCost |
+----------+---------------+-----------+-------------+----------------+--------------+
| M001     | Dahek         | Action    | 2022-01-26  |        1245000 |      1300000 |
| M002     | Attack        | Action    | 2022-01-28  |        1120000 |      1250000 |
| M003     | Looop Lapeta  | Thriller  | 2022-02-01  |         250000 |       300000 |
| M004     | Badhai Do     | Drama     | 2022-02-04  |         720000 |        68000 |
| M005     | Shabaash Mithu | Biography | 2022-02-04  |        1000000 |       800000 |
| M006     | Gehraiyaan    | Romance   | 2022-02-11  |         150000 |       120000 |
+----------+---------------+-----------+-------------+----------------+--------------+
6 rows in set (0.01 sec)
```

2. Show the structure of table

mysql> desc movies;

```
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Movie_ID       | varchar(5)  | NO   | PRI | NULL    |       |
| MovieName      | varchar(50) | YES  |     | NULL    |       |
| Type           | varchar(20) | YES  |     | NULL    |       |
| ReleaseDate    | date        | YES  |     | NULL    |       |
| ProductionCost | int         | YES  |     | NULL    |       |
| BusinessCost   | int         | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

3. Add a column named **collection** with decimal data type with structure 12, 2

mysql> alter table movies add column collection decimal(12,2);

```
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc movies;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| Movie_ID       | varchar(5)   | NO   | PRI | NULL    |       |
| MovieName      | varchar(50)  | YES  |     | NULL    |       |
| Type           | varchar(20)  | YES  |     | NULL    |       |
| ReleaseDate    | date         | YES  |     | NULL    |       |
| ProductionCost | int          | YES  |     | NULL    |       |
| BusinessCost   | int          | YES  |     | NULL    |       |
| collection     | decimal(12,2) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

4. Change the data type of movie_id to varchar

mysql> alter table movies modify column movie_id varchar(4);

```
Query OK, 6 rows affected (0.08 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

Table Affected:

```
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| movie_id       | varchar(4)    | NO   | PRI | NULL    |       |
| mname          | varchar(50)   | YES  |     | NULL    |       |
| Type           | varchar(20)   | YES  |     | NULL    |       |
| ReleaseDate    | date          | YES  |     | NULL    |       |
| ProductionCost | int           | YES  |     | NULL    |       |
| BusinessCost   | int           | YES  |     | NULL    |       |
| collection     | decimal(12,2) | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

5. Rename a column MovieName to MName

mysql> alter table movies rename moviename to mname;

```
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| movie_id       | varchar(4)    | NO   | PRI | NULL    |       |
| mname          | varchar(50)   | YES  |     | NULL    |       |
| Type           | varchar(20)   | YES  |     | NULL    |       |
| ReleaseDate    | date          | YES  |     | NULL    |       |
| ProductionCost | int           | YES  |     | NULL    |       |
| BusinessCost   | int           | YES  |     | NULL    |       |
| collection     | decimal(12,2) | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

6. Delete a column collection

mysql> alter table movies drop column collection;

```
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Set 3 – Select queries including order by

1. Display the movie name, type, releasedate and total cost of all movies

   mysql > select mname, type, releasedate, productioncost + businesscost as 'total cost' from movies;

```
+----------------+-----------+-------------+------------+
| mname          | type      | releasedate | total cost |
+----------------+-----------+-------------+------------+
| Dahek          | Action    | 2022-01-26  |    2545000 |
| Attack         | Action    | 2022-01-28  |    2370000 |
| Looop Lapeta   | Thriller  | 2022-02-01  |     550000 |
| Badhai Do      | Drama     | 2022-02-04  |     788000 |
| Shabaash Mithu | Biography | 2022-02-04  |    1800000 |
| Gehraiyaan     | Romance   | 2022-02-11  |     270000 |
+----------------+-----------+-------------+------------+
6 rows in set (0.00 sec)
```

2. Display all details of action and thriller movies

   mysql> select * from movies where type in ('action','thriller');

```
+---------+-------------+----------+-------------+----------------+--------------+
| movie_id | mname       | Type     | ReleaseDate | ProductionCost | BusinessCost |
+---------+-------------+----------+-------------+----------------+--------------+
| M001    | Dahek       | Action   | 2022-01-26  |        1245000 |      1300000 |
| M002    | Attack      | Action   | 2022-01-28  |        1120000 |      1250000 |
| M003    | Looop Lapeta | Thriller | 2022-02-01  |         250000 |       300000 |
+---------+-------------+----------+-------------+----------------+--------------+
3 rows in set (0.00 sec)
```

3. Display movie name, types and releasedate of all movies released in the month of February, 2022

   mysql > select mname, type, releasedate from movies where releasedate between '2022-02-01' and '2022-02-28';

```
+----------------+-----------+-------------+
| mname          | type      | releasedate |
+----------------+-----------+-------------+
| Looop Lapeta   | Thriller  | 2022-02-01  |
| Badhai Do      | Drama     | 2022-02-04  |
| Shabaash Mithu | Biography | 2022-02-04  |
| Gehraiyaan     | Romance   | 2022-02-11  |
+----------------+-----------+-------------+
4 rows in set (0.00 sec)
```

4. Display the details of movies whose name ends with letter k.

mysql > select * from movies where mname like '%k';

```
+----------+--------+--------+-------------+----------------+--------------+
| movie_id | mname  | Type   | ReleaseDate | ProductionCost | BusinessCost |
+----------+--------+--------+-------------+----------------+--------------+
| M001     | Dahek  | Action | 2022-01-26  |        1245000 |      1300000 |
| M002     | Attack | Action | 2022-01-28  |        1120000 |      1250000 |
+----------+--------+--------+-------------+----------------+--------------+
2 rows in set (0.00 sec)
```

5. Display the details of movies which is not released yet

mysql > select * from movies where releasedate is null;

```
+----------+---------+--------+-------------+----------------+--------------+
| movie_id | mname   | Type   | ReleaseDate | ProductionCost | BusinessCost |
+----------+---------+--------+-------------+----------------+--------------+
| M007     | Sarfira | Action | NULL        |        1250000 |      1200000 |
+----------+---------+--------+-------------+----------------+--------------+
1 row in set (0.00 sec)
```

6. Display the movie name type release date and production cost in the descending order of production cost

mysql> select mname, type, releasedate, productioncost from movies order by productioncost desc;

```
+---------------+-----------+-------------+-----------------+
| mname         | type      | releasedate | productioncost  |
+---------------+-----------+-------------+-----------------+
| Sarfira       | Action    | NULL        |         1250000 |
| Dahek         | Action    | 2022-01-26  |         1245000 |
| Attack        | Action    | 2022-01-28  |         1120000 |
| Shabaash Mithu | Biography | 2022-02-04  |         1000000 |
| Badhai Do     | Drama     | 2022-02-04  |          720000 |
| Looop Lapeta  | Thriller  | 2022-02-01  |          250000 |
| Gehraiyaan    | Romance   | 2022-02-11  |          150000 |
+---------------+-----------+-------------+-----------------+
7 rows in set (0.00 sec)
```

Set 4 – Group by, having and aggregate functions

1. Display the number of movies for each type

   mysql> select type, count(*) from movies group by type;

```
+-----------+----------+
| type      | count(*) |
+-----------+----------+
| Action    |        3 |
| Thriller  |        1 |
| Drama     |        1 |
| Biography |        1 |
| Romance   |        1 |
+-----------+----------+
5 rows in set (0.00 sec)
```

2. Display the number of unique movies

   mysql> select count(distinct type) from movies;

```
+-----------------------+
| count( distinct type) |
+-----------------------+
|                     5 |
+-----------------------+
1 row in set (0.00 sec)
```

3. Display the maximum production cost for each movie type

   mysql> select type, max(productioncost) from movies group by type;

```
+-----------+-----------------------+
| type      | max(productioncost)   |
+-----------+-----------------------+
| Action    |               1250000 |
| Thriller  |                250000 |
| Drama     |                720000 |
| Biography |               1000000 |
| Romance   |                150000 |
+-----------+-----------------------+
5 rows in set (0.00 sec)
```

4. Display the date of movies released latest

mysql> select max(releasedate) from movies;

```
+-------------------+
| max(releasedate) |
+-------------------+
| 2022-02-11        |
+-------------------+
1 row in set (0.00 sec)
```

5. Display the average business cost of movies for each type

Mysql> select type, avg(businesscost) from movies group by type;

```
+-----------+--------------------+
| type      | avg(businesscost) |
+-----------+--------------------+
| Action    |      1250000.0000 |
| Thriller  |       300000.0000 |
| Drama     |        68000.0000 |
| Biography |       800000.0000 |
| Romance   |       120000.0000 |
+-----------+--------------------+
5 rows in set (0.00 sec)
```

6. Display the type and number of movies for each type where movie type is more than two

mysql> select type, count(*) from movies group by type having count(*)>2;

```
+--------+----------+
| type   | count(*) |
+--------+----------+
| Action |        3 |
+--------+----------+
1 row in set (0.00 sec)
```

Set 5 – Join Queries

Create two tables as follows and write given queries below:

Table - artist

artist_id	artist_name	label_owner
101	Vishal Shekhar	Sony Music
120	Vishal Mishra	Zee Music
125	Udit Narayan	T-Series

Table – Song

song_id	artist_id	name
1111	101	Jhume Jo Pathan
1112	120	Khubsoorat
1113	125	Papa Kehte hai
1114	101	Swag Se Swagat
1115	120	Narazagi
1116	125	Phir Bhi Dil Hai Hindustani
1117	125	Mei Nikal Gaddi Leke
1118	120	Pehle Bhi Mein
1119	101	Jai Jai Shiv Shankar

1. Display the cartesian product of artist and song tables:
   mysql > select * from artist, song;

artist_id	artist_name	label_owner	song_id	artist_id	name
125	Udit Narayan	T-Series	1111	101	Jhume Jo Pathan
120	Vishal Mishra	Zee Music	1111	101	Jhume Jo Pathan
101	Vishal Shekhar	Sony Music	1111	101	Jhume Jo Pathan
125	Udit Narayan	T-Series	1112	120	Khubsoorat
120	Vishal Mishra	Zee Music	1112	120	Khubsoorat
101	Vishal Shekhar	Sony Music	1112	120	Khubsoorat
125	Udit Narayan	T-Series	1113	125	Papa Kehte hai
120	Vishal Mishra	Zee Music	1113	125	Papa Kehte hai
101	Vishal Shekhar	Sony Music	1113	125	Papa Kehte hai
125	Udit Narayan	T-Series	1114	101	Swag Se Swagat
120	Vishal Mishra	Zee Music	1114	101	Swag Se Swagat
101	Vishal Shekhar	Sony Music	1114	101	Swag Se Swagat
125	Udit Narayan	T-Series	1115	120	Narazagi
120	Vishal Mishra	Zee Music	1115	120	Narazagi
101	Vishal Shekhar	Sony Music	1115	120	Narazagi
125	Udit Narayan	T-Series	1116	125	Phir Bhi Dil Hai Hindustani
120	Vishal Mishra	Zee Music	1116	125	Phir Bhi Dil Hai Hindustani
101	Vishal Shekhar	Sony Music	1116	125	Phir Bhi Dil Hai Hindustani
125	Udit Narayan	T-Series	1117	125	Mei Nikal Gaddi Leke
120	Vishal Mishra	Zee Music	1117	125	Mei Nikal Gaddi Leke
101	Vishal Shekhar	Sony Music	1117	125	Mei Nikal Gaddi Leke
125	Udit Narayan	T-Series	1118	120	Pehle Bhi Mein
120	Vishal Mishra	Zee Music	1118	120	Pehle Bhi Mein
101	Vishal Shekhar	Sony Music	1118	120	Pehle Bhi Mein
125	Udit Narayan	T-Series	1119	101	Jai Jai Shiv Shankar
120	Vishal Mishra	Zee Music	1119	101	Jai Jai Shiv Shankar
101	Vishal Shekhar	Sony Music	1119	101	Jai Jai Shiv Shankar

2. Display the records of artist and song using equijoin.

   mysql > select * from artist, song where artist.artist_id=song.artist_id;

artist_id	artist_name	label_owner	song_id	artist_id	name
101	Vishal Shekhar	Sony Music	1111	101	Jhume Jo Pathan
101	Vishal Shekhar	Sony Music	1114	101	Swag Se Swagat
101	Vishal Shekhar	Sony Music	1119	101	Jai Jai Shiv Shankar
120	Vishal Mishra	Zee Music	1112	120	Khubsoorat
120	Vishal Mishra	Zee Music	1115	120	Narazagi
120	Vishal Mishra	Zee Music	1118	120	Pehle Bhi Mein
125	Udit Narayan	T-Series	1113	125	Papa Kehte hai
125	Udit Narayan	T-Series	1116	125	Phir Bhi Dil Hai Hindustani
125	Udit Narayan	T-Series	1117	125	Mei Nikal Gaddi Leke

3. Display the records of artist and song tables using natural join.

   mysql > select * from artist natural join song;

artist_id	artist_name	label_owner	song_id	name
101	Vishal Shekhar	Sony Music	1111	Jhume Jo Pathan
101	Vishal Shekhar	Sony Music	1114	Swag Se Swagat
101	Vishal Shekhar	Sony Music	1119	Jai Jai Shiv Shankar
120	Vishal Mishra	Zee Music	1112	Khubsoorat
120	Vishal Mishra	Zee Music	1115	Narazagi
120	Vishal Mishra	Zee Music	1118	Pehle Bhi Mein
125	Udit Narayan	T-Series	1113	Papa Kehte hai
125	Udit Narayan	T-Series	1116	Phir Bhi Dil Hai Hindustani
125	Udit Narayan	T-Series	1117	Mei Nikal Gaddi Leke

4. Display artist name, label and songs of sony music company.

   mysql > select artist_name, label_owner, name from artist, song where artist.artist_id=song.artist_id and label_owner='sony music';

artist_name	label_owner	name
Vishal Shekhar	Sony Music	Jhume Jo Pathan
Vishal Shekhar	Sony Music	Swag Se Swagat
Vishal Shekhar	Sony Music	Jai Jai Shiv Shankar

5. Display artist name, song name from artist and song which song starts with ‘p’.
   mysql > select artist_name, name from artist a, song s where a.artist_id = s.artist_id and s.name like ‘p%’;

artist_name	name
Udit Narayan	Papa Kehte hai
Udit Narayan	Phir Bhi Dil Hai Hindustani
Vishal Mishra	Pehle Bhi Mein

Part C – Python & MySQL Connectivity Programs

Chapter 14 – Interface Python with MySQL

1. Write a program to connect with mysql database and insert a record into database.

Code:

```
import mysql.connector as my
cn=my.connect(host='localhost',user='root',passwd='root',database
='practical_2025')
cur=cn.cursor()
aid=int(input("Enter Artist ID:"))
aname=input("Enter Artist Name:")
lbl=input("Enter Label:")
cur.execute("insert into artist
values({},'{}','{}')".format(aid,aname,lbl))
cn.commit()
print("Record Inserted...")
cn.close()
```

Output:

```
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/
le/pymysql1.py
Enter Artist ID:115
Enter Artist Name:KK
Enter Label:Sony Music
Record Inserted...
```

artist_id	artist_name	label_owner
101	Vishal Shekhar	Sony Music
115	KK	Sony Music
120	Vishal Mishra	Zee Music
125	Udit Narayan	T-Series

2. Write a program to connect with mysql database and update a record into database.

Code:

```
import mysql.connector as my
cn=my.connect(host='localhost',user='root',passwd='root',database
='practical_2025')
cur=cn.cursor()
aid=int(input("Enter Artist ID:"))
aname=input("Enter Artist Name:")
lbl=input("Enter Label:")
cur.execute("update artist set artist_name='{}',label_owner='{}'
where artist_id={}".format(aname,lbl,aid))
cn.commit()
print("Record Updated...")
cn.close()
```

Output:

```
= RESTART: D:/Website Content/TutorialAICSIP/2024-25
le/pymysql2.py
Enter Artist ID:115
Enter Artist Name:Sonu Nigam
Enter Label:Sony Music
Record Updated...
```

artist_id	artist_name	label_owner
101	Vishal Shekhar	Sony Music
115	Sonu Nigam	Sony Music
120	Vishal Mishra	Zee Music
125	Udit Narayan	T-Series

3. Write a program to connect with mysql database and delete a record into database.

Code:

```
import mysql.connector as my

cn=my.connect(host='localhost',user='root',passwd='root',database
='practical_2025')

cur=cn.cursor()

aid=int(input("Enter Artist ID to delete:"))

cur.execute("delete from artist where

artist_id={}".format(aname,lbl,aid))

cn.commit()

print("Record Deleted...")

cn.close()
```

Output:

```
Enter Artist ID to delete:115
Record Deleted...
```

Table:

artist_id	artist_name	label_owner
101	Vishal Shekhar	Sony Music
120	Vishal Mishra	Zee Music
125	Udit Narayan	T-Series

4. Write a program to connect with mysql database display record of particular label under the artist is working.

Code:

```
import mysql.connector as my

cn=my.connect(host='localhost',user='root',passwd='root',database
='practical_2025')

cur=cn.cursor()

lbl=input("Enter Label:")

cur.execute("select * from artist where label_owner='{}'".format(lbl))

dt=cur.fetchall()

for i in dt:

        print(i)

cn.close()
```

Output:

```
= RESTART: D:/Website Content/TutorialAICSIP/2024-25/CS XII/
le/pymysql4.py
Enter Label:Sony Music
(101, 'Vishal Shekhar', 'Sony Music')
```