RAMAKRISHNA MISSION (C.B.S.E.), GWALIOR

ACADEMIC SESSION :2022-23

C.S. ASSIGNMENT FILE

SUBMITTED TO:
C.S. DEPARTMENT
MR. HEMANT DIXIT SIR

SUBMITTED BY:
SANSKAR SHRIVASTAVA
CLASS : XII 'C'

CERTIFICATE

This is to certify that, Sanskar Shrivastava, of Class 12th C has successfully completed the project file of C.S. Practicals under the Guidance of Hemant Dixit Sir during the Year 2022-23.

Signature of Principal
Swami Supradiptananda Maharaj

Signature of Teacher
Hemant Dixit Sir

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who have provided me with the opportunity to perform this project. I take this opportunity to thank our head of the school, **Swami Supradiptananda Maharaj**, who was always supportive and helpful in fulfilling all our academic requirements. I would like to thank my **C.S. Teacher Mr. Hemant Dixit Sir**, whose valuable guidance has helped me to complete the project. His suggestions and instructions have served as a major contributor towards the successful outcome of this project. I would like to thank my parents for giving encouragement, enthusiasm and invaluable assistance to me. Last but not the least; I would like to thank all my classmates who have supported me in various aspects.

SANSKAR SHRIVASTAVA
XII 'C'

INDEX

1.EXTENDING LIST

```
In [3]: # Python program to demonstrate
        # Addition of elements in a List

        # Creating a List
        List = [1, 2, 'three', 4]
        print("Initial List: ")
        print(List)
        #extending list
        List.extend([8, 'Python', 'Always'])
        print("\nList after performing Extend Operation: ")
        print(List)
```

OUTPUT :

```
Initial List:
[1, 2, 'three', 4]

List after performing Extend Operation:
[1, 2, 'three', 4, 8, 'Python', 'Always']
```

2. POPING ITEM

```
List = [1,2,3,4,5]

# Removing element from the
# Set using the pop() method at specific location
List.pop(2)
print("\nList after popping an element: ")
print(List)
```

OUTPUT :

```
List after popping an element:
[1, 2, 4, 5]
```

3.SLICING LIST

```
# Creating a List
List = ['C', 'O', 'M', 'P', 'U', 'T',
        'E', 'R', 'W', 'O', 'R', 'K', 'S']
print("Initial List: ")
print(List)

# Print elements from beginning to a specific point using Slice
Sliced_List = List[:-5]
print("\nElements sliced till 6th element from last: ")
print(Sliced_List)
```

OUTPUT :

```
Initial List:
['C', 'O', 'M', 'P', 'U', 'T', 'E', 'R', 'W', 'O', 'R', 'K', 'S']

Elements sliced till 6th element from last:
['C', 'O', 'M', 'P', 'U', 'T', 'E', 'R']
```

4.EVEN SQUARER

```
#printing even square list
Even_square = []

for x in range(1, 11):
    if x % 2 == 0:
        Even_square.append(x**2)


print(Even_square)
```

OUTPUT : `[4, 16, 36, 64, 100]`

5.REMOVE METHOD

```
# Creating a List
List = [1, 2, 3, 4, 5, 6,
        7, 8, 9, 10, 11, 12]
print("Initial List: ")
print(List)

# Removing elements from List
# using Remove() method
List.remove(5)
List.remove(6)
print("\nList after Removal of two elements: ")
print(List)
```

OUTPUT :

```
Initial List:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

List after Removal of two elements:
[1, 2, 3, 4, 7, 8, 9, 10, 11, 12]
```

6.MEAN CALCULATOR OF LIST

```
# calculating mean of a list
list1 = eval(input("Enter list : "))
length = len(list1)
mean = sum = 0
for i in range(0, length):
    sum +=list1[i]
mean = sum/length
print("Given list is :", list1)
print("The mean of the given list is :", mean)
```

OUTPUT :

```
Enter list : [1, 2, 3, 4]
Given list is : [1, 2, 3, 4]
The mean of the given list is : 2.5
```

7.SORT

```
#Sort
val = eval(input("Enter a list :") )
print("Original list :", val)
val = val * 2
print("Replicated list :", val)
val. sort( )
print("Sorted in ascending order :", val)
val.sort(reverse = True)
print("Sorted in descending order :", val)
```

OUTPUT :

```
Enter a list :[1, 2, 4, 8, 16, 32]
Original list : [1, 2, 4, 8, 16, 32]
Replicated list : [1, 2, 4, 8, 16, 32, 1, 2, 4, 8, 16, 32]
Sorted in ascending order : [1, 1, 2, 2, 4, 4, 8, 8, 16, 16, 32, 32]
Sorted in descending order : [32, 32, 16, 16, 8, 8, 4, 4, 2, 2, 1, 1]
```

8. DUPLICATE ELEMENT CHECKER

```
# Duplicate Element Checker
listA = eval(input("Enter list1 :" ))
listB = eval(input("Enter list2 :"))
len1 = len(listA)
len2 = len(listB)
for a in range(len1) :
    ele = listA[a]
    if ele in listB :
        print("Overlapped")
        break
else :
    print("Separated" )
```

OUTPUT :

```
Enter list1 :[1, 3, 4, 5, 7, 9]
Enter list2 :[2, 4, 6, 9, 11]
Overlapped
```

9. DUPLICATE REMOVER

```
# Duplicate Remover
val = eval(input("Enter a list :") )
tmp = []
print("original list is :", val)
for a in val :
    if a not in tmp :
        tmp . append (a)
val = list(tmp)
print("List after removing duplicates :", val)
```

OUTPUT :

```
Enter a list :[1, 2, 4, 8, 16, 32, 64, 128]
original list is : [1, 2, 4, 8, 16, 32, 64, 128]
List after removing duplicates : [1, 2, 4, 8, 16, 32, 64, 128]
```

10. TUPLE CREATION USING EXISTING SEQUENCE

```
# Creating tuple from Existing Sequences
T = tuple('HELLO')
print(T)
```

OUTPUT :

```
('H', 'E', 'L', 'L', 'O')
```

11. TRAVERSING A TUPLE

```
# Traversing a Tuple
T = (2,3,4,5,6)
for a in T:
    print(T[a])
```

OUTPUT :

```
2
3
4
5
6
```

12. JOINING TUPLE

```
# Joing Tuple
tpl1 = (1, 3, 4)
tpl2 = (6, 7, 8)
Megatuple = tpl1+tpl2
print(Megatuple)
```

OUTPUT :

```
(1, 3, 4, 6, 7, 8)
```

13. SLICING TUPLE

```
# Slicing Tuple
tpl = (10, 12, 14, 16, 20, 22, 24, 30)
seq = tpl[3:-3]
print(seq)
```

OUTPUT :

```
(16, 20)
```

14. TUPLE DIFFERENTIATOR

```
tup = eval(input ("Enter input for tuple : "))
t1 = tup[ 2:8:2]
t2 = tup[ -3:-9:-2]
t3 = t2[ : : -1]
if t3 == t1 :
    print("The two tuples contain the same elements in reversed order.")
else:
    print("The two tuples contain different elements.")
```

OUTPUT :

```
Enter input for tuple : 1, 2, 3, 4, 5, 6, 7, 8
The two tuples contain different elements.
```

15. INDEX OF THE MINIMUM ELEMENT

```
#Index of minimum element
tup = eval(input("Enter a tuple : "))
mn = min(tup)
print("Minimum element", mn,"is at index", tup.index(mn) )
```

OUTPUT :

```
Enter a tuple : 23, 24, 25, 9, 10
Minimum element 9 is at index 3
```

16. MINIMUM AND MAXIMUM MARKS TELLER

```
# Minimum and Maximum marks Teller
Student = eval(input ("Enter input class then marks : "))
print("Mininmum marks obtained : ", min(Student[1]))
print("Mininmum marks obtained : ", max(Student[1]))
```

OUTPUT :

```
Enter input class then marks : (11, (90, 90, 40,))
Mininmum marks obtained :  40
Mininmum marks obtained :  90
```

17. MIXED KEYS

```
Dict = {1: 'Python', 2: 'For', 3: 'Life'}
print("\nDictionary with the use of Integer Keys: ")
print(Dict)

# Creating a Dictionary
# with Mixed keys
Dict = {'Name': 'Python', 1: [1, 2, 3, 4]}
print("\nDictionary with the use of Mixed Keys: ")
print(Dict)
```

OUTPUT :

```
Dictionary with the use of Integer Keys:
{1: 'Python', 2: 'For', 3: 'Life'}

Dictionary with the use of Mixed Keys:
{'Name': 'Python', 1: [1, 2, 3, 4]}
```

18. CREATING A NESTED DICTIONARY

```
# Creating a Nested Dictionary
Dict = {1: 'Python', 2: 'For',
        3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'Life'}}

print(Dict)
```

OUTPUT :

```
{1: 'Python', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Life'}}
```

19. GET() METHOD

```
# Creating a Dictionary
Dict = {1: 'Python', 'name': 'For', 3: 'Life'}

# accessing a element using get() method
print("Accessing a element using get:")
print(Dict.get(3))
```

OUTPUT :

```
Accessing a element using get:
Life
```

20. ACCESSING ELEMENT USING KEY

```
# Creating a Dictionary
Dict = {'Dict1': {1: 'Python'},
        'Dict2': {'Name': 'For'}}

# Accessing element using key
print(Dict['Dict1'])
print(Dict['Dict1'][1])
print(Dict['Dict2']['Name'])
```

OUTPUT :

```
{1: 'Python'}
Python
For
```

21. POP() METHOD

```
# Creating a Dictionary
Dict = {1: 'Computer', 'name': 'For', 3: 'Life'}

# Deleting a key using pop() method
pop_ele = Dict.pop(1)
print('\nDictionary after deletion: ' + str(Dict))
print('Value associated to poped key is: ' + str(pop_ele))
```

OUTPUT :

```
Dictionary after deletion: {'name': 'For', 3: 'Life'}
Value associated to poped key is: Computer
```

22. LOOPING DICTIONARY

```
#Looping Dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

for x, y in thisdict.items():
    print(x, y)
```

OUTPUT :

```
brand Ford
model Mustang
year 1964
```

23. COPYING DICTIONARY

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

OUTPUT :

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

24. SORT

```
# Dictionary Methods
marks = {}.fromkeys(['Math', 'English', 'Science'], 0)

# Output: {'English': 0, 'Math': 0, 'Science': 0}
print(marks)

for item in marks.items():
    print(item)

# Output: ['English', 'Math', 'Science']
print(list(sorted(marks.keys())))
```

OUTPUT :

```
{'Math': 0, 'English': 0, 'Science': 0}
('Math', 0)
('English', 0)
('Science', 0)
['English', 'Math', 'Science']
```

25. CREATING TUPLE FROM INPUT SEQUENCE

```
# Creating Tuple from Input Sequence
t1 = tuple(eval(input("Enter input for tuple1: ")))
t2 = tuple(eval(input("Enter input for tuple2: ")))
t3 = tuple(eval(input("Enter input for tuple3: ")))
print("Tuple1:", t1)
print("Tuple2:", t2)
print("Tuple3:", t3)
```

OUTPUT :

```
Enter input for tuple1: "abcdef"
Enter input for tuple2: 3, 5, 6
Enter input for tuple3: [11, 12, 13]
Tuple1: ('a', 'b', 'c', 'd', 'e', 'f')
Tuple2: (3, 5, 6)
Tuple3: (11, 12, 13)
```

INDEX-2

1. Write a PL/SQL program to print “HELLO WORLD”.

2. Write a PL/SQL code for inverting a number. (Example: 1234 – 4321)

3. Write a PL/SQL code to find the greatest number among three with Anonymous blocks.

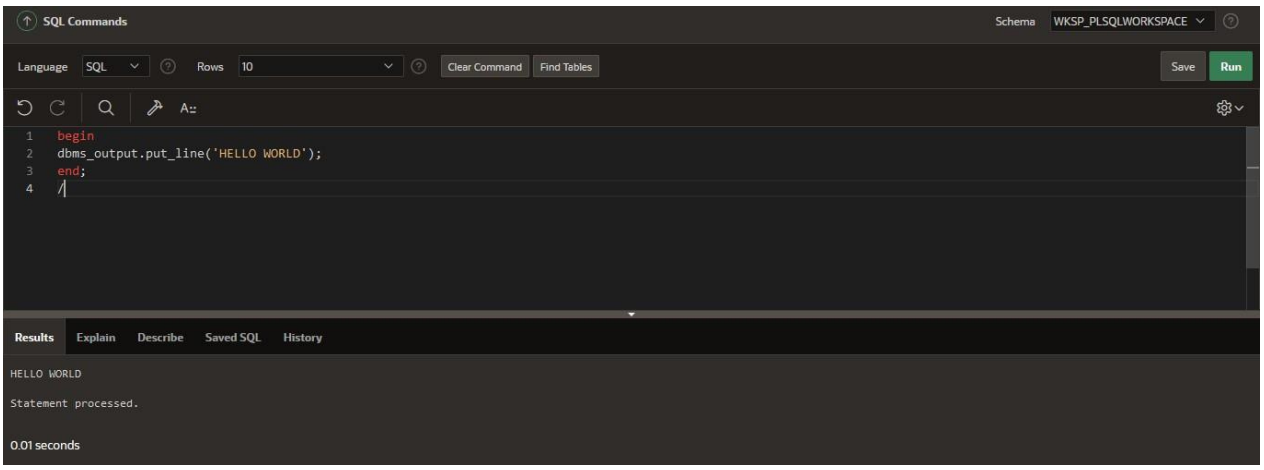
4. Write a PL/SQL code to calculate the area of a circle where radius takes values from 3 to 7.
Store the calculated area in Table AREA. The schema of table is given below: AREA (Radius, Area)

5. Write a PL/SQL program to accept a number and find the factorial of the number.

6. Write a PL/SQL program to display the months between two dates of a year

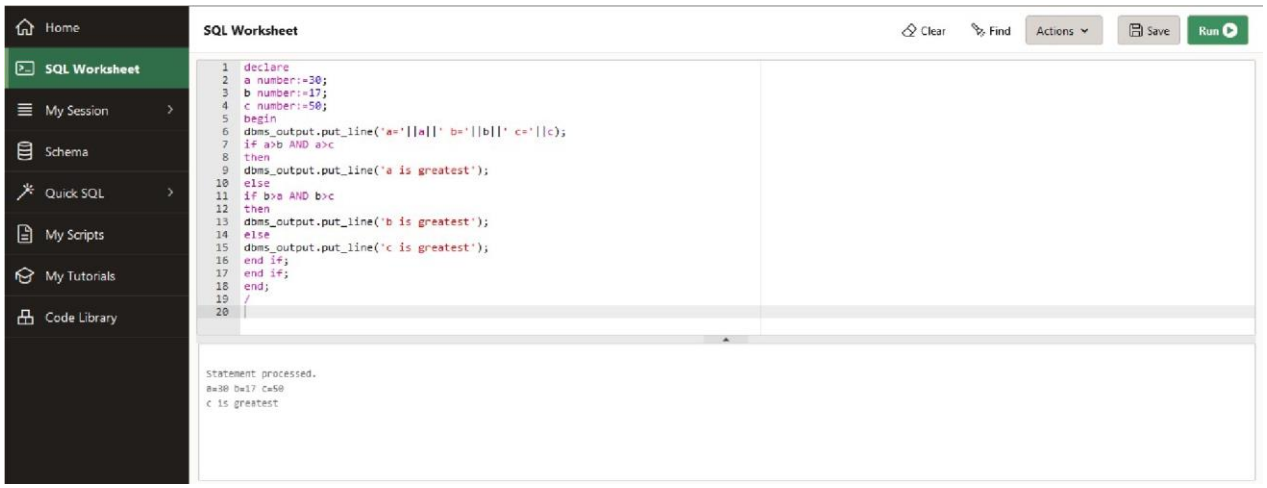
SQL Programs

1. WRITE A PL/SQL PROGRAM TO PRINT “HELLO WORLD”.



2. WRITE A PL/SQL CODE FOR INVERTING A NUMBER. (EXAMPLE: 1234 – 4321)

3. WRITE A PL/SQL CODE TO FIND THE GREATEST NUMBER AMONG THREE WITH ANONYMOUS BLOCKS.



4. WRITE A PL/SQL CODE TO CALCULATE THE AREA OF A CIRCLE WHERE RADIUS TAKES VALUES FROM 3 TO 7.
STORE THE CALCULATED AREA IN TABLE AREA. THE SCHEMA OF TABLE IS GIVEN BELOW: AREA (RADIUS, AREA)

5. WRITE A PL/SQL PROGRAM TO ACCEPT A NUMBER AND FIND THE FACTORIAL OF THE NUMBER.

6. WRITE A PL/SQL PROGRAM TO DISPLAY THE MONTHS BETWEEN TWO DATES OF A YEAR

INDEX-3

PYTHON AND SQL CONNECTIVITY

1. USE PYTHON TO CONNECT TO A MYSQL DATABASE

main.py

```python
import mysql.connector
db_connection = mysql.connector.connect(
  host="localhost",
  user="root",
  passwordd="root"
)
print(db_connection)
```

Line 5 : Col 11

Debugger × Shell × Console × +

```
<mysql.connector.connection.MySQLConnection object at 0x000002338A4C6B00>
```

2. CREATING A TABLE

main.py

```python
import mysql.connector
mydb = mysql.connector.connect(host="localhost",user = "root",passwd="",database = "School")

mycursor= mydb.cursor()
mycursor.execute("CREATE TABLE Student (Rollno int(3) Primary key)")
```

Line 5 : Col 69

Debugger × Shell × Console × +

```
>
```

3. WRITE A PROGRAM THAT USES PYTHON AND SQL TOGETHER TO CREATE A SIMPLE DATABASE APPLICATION.

main.py

```python
import pymysql
db=pymysql.connect("localhost","root","")
c=db.cursor()
sql="create database bank;"
c.execute(sql)
db.close()
```

Line 6 : Col 11

Debugger × Shell × >_ Console × +

```
Replit: Updating package configuration

--> python3 -m poetry add pymysql
Using version ^1.0.2 for PyMySQL
```

4.WRITE A PROGRAM USING PYTHON AND SQL TO UPDATE A RECORD IN TABLE.

```python
import mysql.connector

# Connect to the database
conn = mysql.connector.connect(user='username',
                               password='password',
                               host='host',
                               database='database')

# Create a cursor object
cursor = conn.cursor()

def update_user(id, name, age):
    # Update a user in the table
    cursor.execute("UPDATE users SET name=?, age=? WHERE id=?", (name, age, id))
    conn.commit()

# Update a user
update_user(1, "Alice Smith", 26)

# Close the connection
conn.close()
```

THANK YOU