|  |
| --- |
| NAIPUNNYA PUBLIC SCHOOL KOCHI  **SEAPORT-AIRPORT ROAD,THRIKKAKARA,KOCHI -682021**    **COMPUTER SCIENCE**  **PRACTICAL RECORD TERM-2**    **NAME: …………………………………………………………………………..**  **CLASS: …………………………………………………………………………..**  **REG NO: ……………………………………… YEAR: .....................................**      *Certified that this is a Bonafide Record of Practical Work*      **PRINCIPAL Teacher-in-charge**      Submitted for the Practical Examination held on……………………………………..…..….    at………………………………………………………………………………………...……      Examiner: Date…………………......... |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL NO.** | **DATE** | INDEX  **PROGRAM NAME** | **PAGE NO:** | **SIGN** |
|  |  | **PYTHON** |  |  |
| 1 |  | Write a program to implement all basic operations of stack, such as adding element (PUSH), removing element(POP) and displaying the stack elements(TRAVERSAL) using list data structure in Python. | 1 |  |
| 2 |  | Write a program to display unique vowels present in the given word using stack data structure, | 6 |  |
| 3 |  | Write a menu based program to perform push and pop operations on a Stack.  Each node of the Stack contains the following Member’s details as given below:  Member id integer  Member Name string  Age integer | 8 |  |
|  |  | **SQL** |  |  |
| 4 |  | SQL commands (Table: Employee and Department) SQL queries using CREATE,INSERT, SELECT,DISTINCT, BETWEEN, IN. | 14 |  |
| 5 |  | SQL commands (Table: Employee and Department) SQL queries using equi JOIN and natural JOIN | 18 |  |
| 6 |  | SQL commands (Table: Teacher)  Queries for Aggregate functions- SUM( ), AVG( ), MIN( ), MAX( ), COUNT( ) | 21 |  |
| 7 |  | SQL commands (Table: Admin)  SQL queries using ALTER,UPDATE,LIKE. | 23 |  |
| 8 |  | SQL commands (Table: Student)  SQL queries using ORDER BY,GROUP BY, HAVING | 26 |  |
|  |  | **Python with MySQL (Database connectivity)** |  |  |
| 9 |  | Write a program to connect Python with MySQL using database connectivity and perform the following operations on data in database BookShop:   * CREATE A TABLE * INSERT THE DATA * UPDATE THE RECORD * DELETE THE DATA | 29 |  |

**PRACTICALS TERM 2 – PYTHON QUESTIONS**

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **1.** | Write a program to implement all basic operations of stack , such as adding element (PUSH), removing element(POP) and displaying the stack elements(TRAVERSAL) using  list. |
| **SOURCE**  **CODE:** | stack=[]  n=int(input('Enter the limit : '))  def PUSH():  if isFull():  print('Stack overflow.')  else:  x=eval(input('Enter the element : '))  stack.append(x)  def POP():  if isEmpty():  print('Stack underflow.')  else:  stack.pop()  def TRAVERSAL():  if isEmpty():  print('Stack underflow')  else:  for i in stack[::-1]:  print(i)  def PEEK():  if isEmpty():  print('Stack underflow')  else:  print(stack[-1])  def isFull():  if len(stack)==n:  return True  else:  return False  def isEmpty():  if len(stack)==0:  return True  else:  return False  while True:  print('1.Push\n2.Pop\n3.Peek\n4.Traversal\n5.Exit')  c=int(input('Enter your choice : '))  if c==1:  print()  x=int(input('Enter the limit (max=%s): '%(n-len(stack))))  for i in range(x):  PUSH()  elif c==2:  POP()  elif c==3:  print('The last element is :',end=' ')  PEEK()  elif c==4:  print('The elements are :')  TRAVERSAL()  elif c==5:  break  else:  print('Invalid entry.')  print() |
| **OUTPUT:** | Enter the limit : 3  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 1  Enter the limit (max=3): 3  Enter the element : 1  Enter the element : 2  Enter the element : 3  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 3  The last element is : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 4  The elements are :  2  1  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 5  >>> |

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **2.** | Write a program to display unique vowels present in the given word using stack. |
| **SOURCE**  **CODE:** | def PUSH(x):  stack.append(x)  def TRAVERSAL(x):  for i in stack[::-1]:  print(i)  x=input('Enter the word : ')  v='aeiou'  stack=[]  for i in x:  if i.lower() in v and i not in stack:  PUSH(i)  TRAVERSAL(stack) |
| **OUTPUT:** | 1)Enter the word : Computer  e  u  o  2)Enter the word : Source  e  u  o  3)Enter the word : Output  u |

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **3.** | Write a menu based program to perform push and pop operations on a Stack.  Each node of the Stack contains the following Member’s details as given below:  Member id integer  Member Name string  Age integer |
| **SOURCE**  **CODE:** | stack=[]  n=int(input('Enter the limit : '))  def PUSH():  if isFull():  print('Stack overflow.')  else:  l=[]  l.append(int(input('Enter the Member Id : ')))  l.append(input('Enter the Member name : '))  l.append(int(input('Enter the age : ')))  stack.append(l)  def POP():  if isEmpty():  print('Stack underflow.')  else:  stack.pop()  def TRAVERSAL():  if isEmpty():  print('Stack underflow')  else:  for i in stack[::-1]:  print('Member Id : %s\nMember Name : %s\nAge : %s\n'%(i[0],i[1],i[2]))    def PEEK():  if isEmpty():  print('Stack underflow')  else:  print('Member Id : %s\nMember Name : %s\nAge : %s\n'%(stack[-1][0],stack[-1][1],stack[-1][2]))  def isFull():  if len(stack)==n:  return True  else:  return False  def isEmpty():  if len(stack)==0:  return True  else:  return False  while True:  print('1.Push\n2.Pop\n3.Peek\n4.Traversal\n5.Exit')  c=int(input('Enter your choice : '))  print()  if c==1:  print()  if isFull():  print('Stack overflow.')  continue  x=int(input('Enter the limit (max=%s): '%(n-len(stack))))  for i in range(x):  if isFull():  print('Stack overflow.')  continue  PUSH()  elif c==2:  POP()  elif c==3:  print('The last element is :')  PEEK()  elif c==4:  print('The elements are :')  TRAVERSAL()  elif c==5:  break  else:  print('Invalid entry.')  print() |
| **OUTPUT:** | Enter the limit : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 1  Enter the limit (max=2): 2  Enter the Member Id : 1  Enter the Member name : a  Enter the age : 2  Enter the Member Id : 2  Enter the Member name : b  Enter the age : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 1  Stack overflow.  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 1  Enter the limit (max=1): 1  Enter the Member Id : 2  Enter the Member name : b  Enter the age : 3  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 3  The last element is :  Member Id : 2  Member Name : b  Age : 3  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 4  The elements are :  Member Id : 2  Member Name : b  Age : 3  Member Id : 1  Member Name : a  Age : 2  1.Push  2.Pop  3.Peek  4.Traversal  5.Exit  Enter your choice : 5  >>> |

**PRACTICALS TERM 2 – SQL QUESTIONS**

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
|  | Table : **Employee**      Table : **Department** |
| **4.** | Write SQL Commands for questions (1) to (v) based on the tables Employee &  Department  I. Create tables Employee & Department  II. Insert the first record into the tables Employee & Department  III. Display the various department numbersfrom the table Employee. A department  number should be displayed only once.  IV. Display the employee number, name and salary ofthose employees whose salary is  between 35000 and 40000.  V. Display the names and salaries of all the employees who are working neither in West  zone nor in Centre zone. |
| **SOURCE**  **CODE:** | 1. create table Employee(EmpNo int, EmpName varchar(20), Salary int, Zone varchar(10), Age int, Grade char(1), DeptId int);   create table Department(Deptid int, DeptName varchar(15),  MinSal int, MaxSal int, HOD int);   1. insert into Employee values(1001, ‘R Jain’, 30000, West, 28, ‘A’, 10);   insert into Department values(10, ‘Sales’, 25000, 35000, 1);   1. select distinct DeptId from Employee; 2. select EmpNo, EmpName, Salary from Employee where 35000<Salary<40000; 3. select EmpName, Salary from Employee where Zone not in (‘West’,’Centre’); |
| **OUTPUT:** |  |

|  |  |  |
| --- | --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** | |
| **5.** | Write SQL Commands for questions (1) to (viii) based on the tables Employee & Department.  I. To get the name of the column Deptid to D\_id.  II. Display the name of those employees whose names starts with ‘H’.  III. List the name of employees not having any Grade.  IV. Display the list of employees in descending order of employee code.  V. Find the average salary at each department.  VI. Find maximum salary of each department and display the name of that department which has maximum salary more than 37000.  VII. To delete the records whose grade is not entered.  VIII. Display the name and salary of those employees whose grade is A and from the sales department after incrementing by10% | |
| **SOURCE**  **CODE:** | 1. alter table Department change column Deptid D\_id int;   alter table Employee change column Deptid D\_id int;   1. select EmpName from Employee where EmpName like ‘H%’; 2. select EmpName from Employee where Grade=Null; 3. select \* from Employee order by EmpNo desc; 4. select avg(Salary) from Employee group by D\_id; 5. select max(Salary), DeptName from Employee, Department where Employee.D\_id=Department.D\_id group by DeptName; 6. delete from Employee where Grade is Null; 7. select EmpName, Salary\*1.1 as Salary from Employee where D\_id=10; | |
| **OUTPUT:** | 1. 1     1           1. 1 2. 1      1. 1      1. 1 | |
| **QUESTION**  **NO.** | | **OBJECTIVE & SOLUTIONS** |
| **6.** | | Write SQL Commands for questions 1 to 5 based on the table TEACHER  1) To show all information about the teachers whose salary is greater than 20000.  2) To list all female teachers who are from History department.  3) To list all names of all teachers beginning with ‘M’ sorted by Name in descending order.  4) To count number of teachers with age less than 32.  5) To display the maximum salary . |
| **SOURCE**  **CODE:** | | 1. Select \* from TEACHER where SALARY>20000; 2. Select NAME from TEACHER where SEX=’F’; 3. Select NAME from TEACHER where NAME like ‘M%’ order by NAME desc; 4. Select Count(\*) from TEACHER where AGE>32; 5. Select max(SALARY) from TEACHER; |
| **OUTPUT:** | | 1. 1 2. 1 3. 1 4. 1 5. 1 |

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **7.** | Write SQL Commands for questions 1 to 3 on the basis of table ADMIN and give the output for queries 4 and 5.    1) To alter the table to add new column EXPERIENCE.  2) To update table ADMIN by giving all staff 10 yrs experience.  3) To display the records in the descending order of staff name .  4) To display the number of staff names beginning with letter ‘R’.  5) To display the number of teachers in each subject |
| **SOURCE**  **CODE:** | 1. Alter table ADMIN add EXPERIENCE int; 2. Update ADMIN set EXPERIENCE=10; 3. Select \* from ADMIN order by TNAME; 4. Select count(TNAME) from ADMIN where TNAME like ‘R%’; 5. Select count(\*), SUBJECT from ADMIN group by SUBJECT; |
| **OUTPUT:** | 1. 1      1. 1      1. 1 2. 1 |

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **8.** | Write queries for (i) to (v) based on the table STUDENT    1) To get the SUBJECT and the average marks scored by the students in that subject from the table STUDENT.  2) To change the name of the column ST\_CODE to ADMN\_NO.  3) To get the student names sorted by marks in the descending order.  4) To get the number of students who secured more than 80% marks from the table student.  5) To get the student names that start with "a" and are at least 5 characters in length |
| **SOURCE**  **CODE:** | 1. Select SUBJECT, avg(MARKS) from STUDENT group by SUBJECT; 2. Alter table STUDENT change ST\_CODE ADMN\_NO int; 3. Select ST\_NAME from STUDENT order by MARKS desc; 4. Select count(\*) from STUDENT where MARKS>80; 5. Select ST\_NAME from STUDENT where ST\_NAME like ‘A%’ and length(ST\_NAME)>4; |
| **OUTPUT:** | 1. 1      1. 1 2. 1 3. 1 |

**PRACTICALS TERM 2 – SQL INTERFACING WITH PYTHON**

|  |  |
| --- | --- |
| **QUESTION**  **NO.** | **OBJECTIVE & SOLUTIONS** |
| **9** | Write a program to connect Python with MySQL using database connectivity  and perform the following operations on data in database BookShop: Fetch,  Update and delete the data.  A) CREATE A TABLE  B) INSERT THE DATA  C) UPDATE THE RECORD  D) DELETE THE DATA |
| **SOURCE**  **CODE:** | import mysql.connector  mydb=mysql.connector.connect(  host='localhost',  user='root',  password='1125899839733759',  database='bookshop')  cursor=mydb.cursor()  #creating the table  cursor.execute('create table book(bookno int, bookname varchar(20), price float, author varchar(20),publisher varchar(20))')  while True:  c=int(input('1. Insert data\n2. Update a record.\n3. Delete a record\n4. Exit\nEnter your choice (1,2,3,4) : '))  print()  if c==1:  l=()  l+=(int(input('Enter the book number : '))),  a=input('Enter the name of the book : ')  a='"'+a+'"'  l+=(a),  l+=(eval(input('Enter the price : '))),  a=input('Enter the name of the author : ')  a='"'+a+'"'  l+=(a),  a=input('Enter the publisher : ')  a='"'+a+'"'  l+=(a),  cursor.execute('insert into book values(%s,%s,%s,%s,%s)'%(l))  mydb.commit()  print('\nRecord inserted.')  elif c==2:  l=[]  n=int(input('Enter the book number of the record to be updated : '))  cursor.execute('select bookno from book')  for i in cursor:  for j in i:  l.append(j)  if n in l:  a=int(input('1. Bookname\n2. Price\n3. Author\n4. Publisher\nEnter the value to be updated : '))  if a==1:  nn=int(input('Enter the new bookname : '))  cursor.execute('update book set bookname=%s where bookno=%s'%(nn,n))  mydb.commit()  elif a==2:  np=eval(input('Enter the new price of the book : '))  cursor.execute('update book set price=%s where bookno=%s'%(np,n))  mydb.commit()  elif a==3:  na=input('Enter the author : ')  cursor.execute('update book set author=%s where bookno=%s'%(na,n))  mydb.commit()  elif a==4:  np=int(input('Enter the publisher : '))  cursor.execute('update book set publisher=%s where bookno=%s'%(np,n))  mydb.commit()  else:  print('Invalid entry.')  break  else:  print('Record not found.')  elif c==3:  l=[]  n=int(input('Enter the book number of the record to be deleted : '))  cursor.execute('select bookno from book')  for i in cursor:  for j in i:  l.append(j)  if n in l:  cursor.execute('delete from book where bookno=%s'%(n))  mydb.commit()  print('Record deleted.')  else:  print('Record not found.')  elif c==4:  break  else:  print('Invalid entry.')  print() |

|  |  |
| --- | --- |
| **OUTPUT:** | Output in python:  1. Insert data  2. Update a record.  3. Delete a record  4. Exit  Enter your choice (1,2,3,4) : 1  Enter the book number : 1  Enter the name of the book : a  Enter the price : 2  Enter the name of the author : a  Enter the publisher : a  Record inserted.  1. Insert data  2. Update a record.  3. Delete a record  4. Exit  Enter your choice (1,2,3,4) : 1  Enter the book number : 2  Enter the name of the book : b  Enter the price : 4  Enter the name of the author : b  Enter the publisher : b  Record inserted.  1. Insert data  2. Update a record.  3. Delete a record  4. Exit  Enter your choice (1,2,3,4) : 2  Enter the book number of the record to be updated : 1  1. Bookname  2. Price  3. Author  4. Publisher  Enter the value to be updated : 2  Enter the new price of the book : 1  1. Insert data  2. Update a record.  3. Delete a record  4. Exit  Enter your choice (1,2,3,4) : 3  Enter the book number of the record to be deleted : 2  Record deleted.  1. Insert data  2. Update a record.  3. Delete a record  4. Exit  Enter your choice (1,2,3,4) : 4  >>>  Effects on database bookshop:  Table created:    2 Records inserted :    1st record updated :    2nd record deleted : |