RDBMS PRACTICAL RECORD

4th Semester BSc.CS Calicut University RDBMS Practical Solved

DEPARTMENT OF COMPUTER SCIENCE
LAKSHMI NARAYANA ARTS AND SCIENCE COLLEGE, MAYANNUR

PROGRAM 1

Create a table customer (cust_no varchar (5), cust_name varchar (15), age number, phone varchar (10))

a) insert 5 records and display it

b) add new field d_birth with date datatype

c) create another table cust_phone with fields cust_name and phone from customer table

d) remove the field age

e) change the size of the cust_name to 25

f) delete all the records from the table

g) rename the table cutomer to cust

h) drop the table

Table 1

```
Create table customer (cust_no varchar(5), cust_name varchar(15), age  int, phone varchar(10) );

a) insert into customer(cust_no,cust_name,age,phone)values('1','ravi',25,'52634189');
insert into customer(cust_no,cust_name,age,phone)values('2','ani',26,' 63544189');
insert into customer(cust_no,cust_name,age,phone)values('3',' jibin', 28, 44025189');
insert into customer(cust_no,cust_name,age,phone)values('4','resma',27,'8254879');
insert into customer(cust_no,cust_name,age,phone)values('5','meena',26,'989644');
select * from customer;
b) ALTER TABLE customer ADD d_birth date;
c) ALTER TABLE customer ADD PRIMARY KEY (phone);

create table cust_phone as (select cust_name, phone from customer);
d) ALTER TABLE customer DROP COLUMN age;
select * from customer;
```

e) ALTER TABLE customer ALTER column cust_name type varchar(25);

f) truncate table cust_phone;

g) ALTER TABLE customer RENAME TO cust;

h) drop table cust_phone;

✧✧

PROGRAM 2

Create a table sale_man (salesman_no primary key, s_name not null, place, phone unique) Create table sales_order (order_no primary key order_date not null salesman_no foreign key references salesman_no in sales_man del_type values should be either P or F (check constraints) order_status values should be 'Inprocess','Fullfilled','Backorder', 'Cancelled' (check constraints))

a) Insert few records in both tables

c) Delete primary key from sales_man table

b) Delete Foreign key and Check constraints from sales_order table

d) Add primary key in sales_man using ALTER TABLE

e) Add foreign key and CHECK constraints in sales_order table using ALTER TABLE

Table 1

Create table sales_man(salesman_no int,s_name varchar(15) NOT NULL,place varchar(15),phone int unique,constraint pksalesman primary key(salesman_no));

Table 2

Create table sales_order (order_no int,order_date date NOT NULL,salesman_no int,del_type varchar(5) ,order_status varchar(10) ,constraint pksalesorder primary key(order_no),constraint fksalesorder foreign key(salesman_no) references sales_man(salesman_no),constraint cksalesorder1 check (del_type='p' or

del_type='f'),constraint cksalesorder2 check(order_status='inprocess' or order_status='fullfilled' or order_status='backorder' or order_status='cancelled'));

a)

Table 1

Insert into sales_man(salesman_no,s_name,place,phone)values(1,'intel','palakkad','9547455');

insert into sales_man(salesman_no ,s_name ,place,phone)values(2,'wipro','kozhikode','89547455');

insert into sales_man(salesman_no ,s_name ,place ,phone)values(3,'rolex','malappuram','88547455');

select * from sales_man;

Table 2

insert into sales_order (order_no,order_date,salesman_no,del_type,order_status) values(1,'02-02-2019',1,'f','fullfilled');

insert into sales_order (order_no,order_date,salesman_no,del_type,order_status) values(2,'12-02-2019',2,'p','inprocess');

insert into sales_order (order_no,order_date,salesman_no,del_type,order_status) values(3,'02-12-2019',2,'p','cancelled');

select * from sales_order;

c) ALTER TABLE sales_man DROP CONSTRAINT pksalesman;

b) ALTER TABLE sales_order DROP CONSTRAINT fksalesorder;

ALTER TABLE sales_order DROP CONSTRAINT cksalesorder1;

d) ALTER TABLE sales_man ADD CONSTRAINT pksalesman primary key(salesman_no);

e) ALTER TABLE sales_order ADD CONSTRAINT fksalesorder foreign key(salesman_no) references sales_man(salesman_no);

ALTER TABLE sales_order ADD CONSTRAINT cksalesorder1 check (del_type='p' or del_type='f');

✧✧✧

PROGRAM 3

Create a table Hospital with the fields (doctorid, doctorname, department, qualification, experience). Write the queries to perform the following.

a) Insert 5 records

b) Display the details of Doctors

c) Display the details of doctors who have the qualification 'MD'

d) Display all doctors who have more than 5 years' experience but do not have the qualification 'MD' e) Display the doctors in 'Skin' department f) update the experience of doctor with doctored='D003' to 5

g) Delete the doctor with DoctorID='D005'

```
Create table Hospital (doctorid varchar(15) ,doctorname varchar(15),department varchar(15),qualification varchar(15),experience int);
```

a)

```
insert into Hospital(doctorid,doctorname,department,qualification,experience)values('D001','visakh','skin','MD',5);

insert into Hospital (doctorid,doctorname,department,qualification,experience)values('D002','anju','ortho','MBBS',6);

insert into Hospital (doctorid,doctorname,department,qualification,experience)values('D003','akhil','skin','MBBS',4);

insert into Hospital(doctorid,doctorname,department,qualification,experience)values('D004','vinu','dental','MD',7);

insert into Hospital (doctorid,doctorname,department,qualification,experience)values('D005','jithu','surgen','MD',4);
```

b) select * from hospital;

c)elect * from hospital where qualification='MD';

d)select * from hospital where EXPERIENCE>5 and qualification!='MD';

e)select * from hospital where DEPARTMENT='skin';

f) update hospital set experience=5 where doctorid='D003';

select * from hospital ;

g)delete from hospital where doctorid='D005';

✧✧

PROGRAM 4

Create the following tables Bank_customer (accno primary key, cust_name, place) Deposit (accno foreign key, deposit_no, damount) Loan (accno foreign key loan_no, Lamount) Write the following queries

a) Display the details of the customers

b) Display the customers along with deposit amount who have only deposit with the bank

c) Display the customers along with loan amount who have only loan with the bank

d) Display the customers they have both loan and deposit with the bank

e) Display the customer who have neither a loan nor a deposit with the bank

Table 1

Create table Bank_customer (accno int, cust_name varchar(15), place varchar(15),constraint pkbankcustomer primary key(accno));

insert into Bank_customer (accno, cust_name,place)values(100,'nithin','palakkad');

insert into Bank_customer (accno, cust_name,place)values(101,'sanjay','kozhikode');

insert into Bank_customer (accno, cust_name,place)values(102,'meenu','malappuram');

insert into Bank_customer (accno, cust_name,place)values(104,'anju','malappuram');

Table 2

Create table Deposit(accno int,deposit_no int,damount int,constraint fkdeposit foreign key(accno)

references Bank_customer(accno));

insert into Deposit(accno,deposit_no,damount)values(100,2,15000);

insert into Deposit(accno,deposit_no,damount)values(101,3,37000);

insert into Deposit(accno,deposit_no,damount)values(102,4,41000);

Table 3

Create table Loan(accno int,loan_no int,Lamount int,constraint fkloan foreign key(accno)

references Bank_customer(accno));

insert into Loan(accno ,loan_no,Lamount)values(100,10,50000);

insert into Loan(accno ,loan_no,Lamount)values(101,11,20000);

a) select * from Bank_customer ;

select * from Deposit ;

select * from Loan;

b) SELECT Bank_customer.ACCNO,Bank_customer.CUST_NAME,

Bank_customer.PLACE,Deposit.DEPOSIT_NO,Deposit.DAMOUNT

FROM Bank_customer INNER JOIN Deposit

```
ON Bank_customer.ACCNO = Deposit.ACCNO ;

c) SELECT Bank_customer.ACCNO,Bank_customer.CUST_NAME,
Bank_customer.PLACE,Loan.loan_no,Loan.Lamount
FROM Bank_customer INNER JOIN Loan
ON Bank_customer.ACCNO = Loan.ACCNO;

d) SELECT a.ACCNO,a.CUST_NAME,a.PLACE,b.DEPOSIT_NO,c.LOAN_NO
FROM Bank_customer a,Deposit b,Loan c
WHERE a.ACCNO=b.ACCNO
AND a.ACCNO=c.ACCNO;

e) SELECT * FROM Bank_customer
WHERE NOT EXISTS
(SELECT * FROM Loan,deposit
WHERE Bank_customer.ACCNO in ( Deposit.ACCNO ,Loan.ACCNO));
```

✧✧

PROGRAM 5

Create a table employee with fields (EmpID, EName, Salary, Department, and Age). Insert some records. Write SQL queries using aggregate functions and group by clause

A. Display the total number of employees.

B. Display the name and age of the oldest employee of each department.

C. Display the average age of employees of each department

D. Display departments and the average salaries

E. Display the lowest salary in employee table

F. Display the number of employees working in purchase department

G. Display the highest salary in sales department; H. Display the difference between highest and lowest salary.

Create table employee(EmpID int, EName varchar(15),Salary int, Department varchar(15),Age int);

insert into
employee(EmpID,EName,Salary,Department,Age)values(1,'ajith',25000,'purchase',35);

insert into
employee(EmpID,EName,Salary,Department,Age)values(2,'bindu',17000,'sales',26);

insert into

employee(EmpID,EName,Salary,Department,Age)values(3,'nithin',21000,'purchase',28);

insert into
employee(EmpID,EName,Salary,Department,Age)values(4,'rohan',32000,'sales',48);

insert into
employee(EmpID,EName,Salary,Department,Age)values(5,'anita',22000,'packing',28);

insert into
employee(EmpID,EName,Salary,Department,Age)values(6,'kiran',52000,'Manager',56);

select * from employee;

a) select count(*) from employee;

b) select Department,max(Age) from employee group by Department;

b) SELECT department,avg(age) FROM employee group by department;

c) SELECT department,avg(salary) FROM employee group by department;

d) SELECT min(salary) FROM employee;

e) SELECT count(department) FROM employee

where department='purchase';

f) SELECT max(salary) as "highest sal in sales" FROM employee where

department='sales';

g) SELECT max(salary)-min(salary) from employee;

✧✧✧

PROGRAM 6

Create a table product with the fields (Product_code primary key, Product_Name, Category,

Quantity, Price).

Insert some records Write the queries to perform the following.

a. Display the records in the descending order of Product_Name

b. Display Product_Code, Product_Name with price between 20 and 50

c. Display the details of products which belongs to the categories of ‘bath soap’, ‘paste’, or ‘washing powder’

d. Display the products whose Quantity less than 100 or greater than 500

e. Display the products whose names starts with 's'

f. Display the products which not belongs to the category 'paste'

g. Display the products whose second letter is 'u' and belongs to the Category 'washing powder'

```
Create table product(Product_code int , Product_Name varchar(15), Category
varchar(15),
Quantity int, Price int,constraint pkproduct primary key(Product_code));
insert into product(Product_code,Product_Name,Category, Quantity ,
Price)values(1,'colgate','paste',150,15);
insert into product(Product_code,Product_Name,Category, Quantity ,
Price)values(2,'lux','bathsoap',50,25);
insert into product(Product_code,Product_Name,Category, Quantity ,
Price)values(3,'surf','washing powder',10,45);
insert into product(Product_code,Product_Name,Category, Quantity ,
```

Price)values(4,'dove','cream',170,145);

insert into product(Product_code,Product_Name,Category, Quantity ,

Price)values(5,'rexona','bathsoap',50,35);

select * from product;

a) select * from product order by PRODUCT_NAME desc;

b) select PRODUCT_CODE,PRODUCT_NAME from product where PRICE between 20 and 50;

c) select * from product where Category in('bathsoap','paste','washing powder');

d) select * from product where QUANTITY<100 or QUANTITY>500;

e) select * from product where Product_Name like 's%';

f) select * from product where CATEGORY!='paste';

g) select * from product where PRODUCT_NAME like '_u%' and CATEGORY='washingpowder';

✧✧

PROGRAM 7

Consider the employee database given below. Give an expression in SQL for each of the

following queries:

EMPLOYEE (Employee-Name, City)

WORKS (Employee-Name, Company-Name, Salary)

COMPANY (Company-Name, City)

MANAGES (Employee-Name, Manager-Name)

A) Find the names of all employees who work in Infosys

B) Find the names and cities of residence of all employees who works in Wipro

C) Find the names, and cities of all employees who work in Infosys and earn more than Rs.

10,000.

D) Find the employees who live in the same cities as the companies for which they work.

E) Find all employees who do not work in Wipro Corporation.

F) Find the company that has the most employees.

Table 1

```
create table EMPLOYE(Employee_Name varchar(20), City varchar(15));
insert into EMPLOYE(Employee_Name,City)values('visakh','palakkad');
insert into EMPLOYE(Employee_Name,City)values('prajith','kozhikode');
insert into EMPLOYE(Employee_Name,City)values('jeson','trissur');
insert into EMPLOYE(Employee_Name,City)values('midhun','banglore');
insert into EMPLOYE(Employee_Name,City)values('akhil','cochi');
select * from EMPLOYE;
```

Table 2

```
create table WORKS (Employee_Name varchar(20), Company_Name varchar(20),
Salary int);
insert into WORKS (Employee_Name , Company_Name ,
Salary)values('visakh','wipro',25000);
insert into WORKS (Employee_Name , Company_Name ,
Salary)values('prajith','infosys',20000);
insert into WORKS (Employee_Name , Company_Name ,
Salary)values('midhun','wipro',15000);
```

insert into WORKS (Employee_Name , Company_Name , Salary)values('jeson','tcs',25000);

insert into WORKS (Employee_Name , Company_Name , Salary)values('akhil','wipro',15000);

select * from WORKS;

Table 3

create table COMPANY (Company_Name varchar(20), City varchar(20));

insert into COMPANY (Company_Name , City)values('wipro','banglore');

insert into COMPANY (Company_Name , City)values('infosys','cochi');

insert into COMPANY (Company_Name , City)values('tcs','trissur');

insert into COMPANY (Company_Name , City)values('xlr','palakkd');

select * from COMPANY;

Table 4

create table MANAGES (Employee_Name varchar(20), Manager_Name varchar(20));

insert into MANAGES (Employee_Name , Manager_Name)values('visakh','jibin');

insert into MANAGES (Employee_Name , Manager_Name)values('midhun','meena');

insert into MANAGES (Employee_Name , Manager_Name)values('prajith','praveen');

insert into MANAGES (Employee_Name , Manager_Name)values('jeson','raji');

insert into MANAGES (Employee_Name , Manager_Name)values('akhil','rasmi');

select * from MANAGES;

a) select EMPLOYEE_NAME,COMPANY_NAME from works where company_name='infosys';

b) select works.EMPLOYEE_NAME,employe.CITY from employe,WORKS where

employe.EMPLOYEE_NAME=works.EMPLOYEE_NAME and COMPANY_NAME='wipro';

c) select works.EMPLOYEE_NAME,employe.CITY from employe,WORKS where

employe.EMPLOYEE_NAME=works.EMPLOYEE_NAME and COMPANY_NAME='infosys' and

salary>10000;

d) select distinct works.company_name,company.CITY,employe.EMPLOYEE_NAME,employe.city from

employe,company,works where employe.CITY=COMPANY.city and

works.COMPANY_NAME=company.COMPANY_NAME;

e) select EMPLOYEE_NAME,COMPANY_NAME from works where COMPANY_NAME!='wipro' ;

f) SELECT COMPANY_NAME, count(*) as NUM FROM WORKS GROUP BY COMPANY_NAME;

✧✧✧

PROGRAM 8

Write a program code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding value of calculated area in an empty table named areas with field's radius and area.

create table areas(radius int,area int);

create or replace function calarea() returns void as

'

```
declare
 rad int:=3;
area int;
begin
loop
area:=3.14*rad*rad;
insert into areas values(rad,area);
rad:=rad+1;
exit when rad>7;
end loop;
end;
'language'plpgsql';

select calarea();
select * from areas;
```

✧✧✧

PROGRAM 9

Write a program block to calculate the electricity bill by accepting cust_no and units_consumed

```
create table bill(cons_no int PRIMARY KEY, units int, amount float);
create or replace function elebill(int,int) returns void as
'
DECLARE
cons_no alias for $1;
units alias for $2;
amount float;
BEGIN
amount=units*6.40;
```

```
insert into bill values(cons_no,units,amount);
END;
'language'plpgsql';

select elebill(123,234);
select elebill(124,456);

select * from bills;
```

✧✧

PROGRAM 10

Create a procedure to print Fibonacci number up to a limit, limit is passed as an argument

```
create or replace function fibo(int)returns text as
'
declare
a int:=0;
b int:=1;
c int;
n alias for $1;
begin
raise notice ''the fibanocci series is:'';
while a<=n
loop
raise notice ''%'',a;
c:=a+b;
a:=b;
b:=c;
end loop;
```

```
end;
 'language'plpgsql';
select fibo(15);
```

✧✧✧

PROGRAM 11

Create a function to check whether a given number is prime or not

```
create or replace function prime(int) returns text as
'
declare
n alias for $1;
i int:=2;
count int:=1;
msg text;
begin
for i in 2..n/2
loop
if mod(n,i)=0 then
count:=0;
exit;
end if;
end loop;
if count=1 then
msg:=n||"is a prime number";
else
msg:=n||"is not prime number";
end if;
return msg;
end;
```

```
'language'plpgsql';
```

✧✧✧

PROGRAM 12

create a table emp_salary(empno,enamedept,salary) Write a function to return the average salary of a particular department by accepting departmentname as argument.

```
CREATE TABLE emp_salary (
  empno int,
  ename text,
  dept text,
  salary int
);
insert into emp_salary values(101,'raju','cs',10000);
insert into emp_salary values(102,'ramu','maths',15000);
insert into emp_salary values(103,'ram','maths',20000);
insert into emp_salary values(104,'raj','cs',20000);
CREATE OR REPLACE FUNCTION average_salary(department text)
RETURNS float AS
$func$
DECLARE
  average float;
BEGIN
  SELECT AVG(salary) INTO average
  FROM emp_salary
  WHERE dept = department;

  RETURN average;
END
```

```
$func$ LANGUAGE plpgsql;
select average_salary('maths');
```

✧✧✧

PROGRAM 13

An examination has been conducted to a class of 10 students and 4 scores of each student have been provided in the data along with their reg_no, name, total and avg_score. Assign null values to the fields total and average. Write program block to do the following Find the total and average of each student. Update the table with the calculated values Assign a letter grade to each student based on the average Score as avg_score between 90 and 100 - A avg_score 75 -89 – B avg_score 60- 74 - C avg_score 50 -59 - D avg_score below 50 – Failed

```
CREATE TABLE exam_result (rollno int NOT NULL, avg_score int NOT
NULL,grade text);


INSERT INTO exam_result (rollno, avg_score)
VALUES(1, 90),(2, 89),(3, 88),(4, 75),(5, 60),(6, 59),(7, 50),(8, 49),(9, 40),(10, 30);


UPDATE exam_result
SET grade =
 CASE
  WHEN avg_score BETWEEN 90 AND 100 THEN 'A'
  WHEN avg_score BETWEEN 75 AND 89 THEN 'B'
  WHEN avg_score BETWEEN 60 AND 74 THEN 'C'
  WHEN avg_score BETWEEN 50 AND 59 THEN 'D'
  ELSE 'E'
 END;
```

✧✧✧

PROGRAM 14

Create two tables Book (BookID, BookName, Author, Publisher) and Book_Del (Date_of_Del, BookID, BookName). Create and application to generate a trigger before deleting a record from book table. The trigger procedure should insert the deleted BookID and BookName along with current date to the table Book_Del.

Step 1: Create the Book table

```
CREATE TABLE Book (
  BookID serial PRIMARY KEY,
  BookName varchar(255),
  Author varchar(255),
  Publisher varchar(255)
);
```

Step 2: Create the Book_Del table

```
CREATE TABLE Book_Del (
  Date_of_Del date,
  BookID integer,
  BookName varchar(255)
);
```

Step 3: Create the trigger function

```
CREATE OR REPLACE FUNCTION book_delete_trigger()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO Book_Del (Date_of_Del, BookID, BookName)
  VALUES (CURRENT_DATE, OLD.BookID, OLD.BookName);
  RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Step 4: Create the trigger on the Book table

```
CREATE TRIGGER delete_book_trigger
BEFORE DELETE ON Book
FOR EACH ROW EXECUTE PROCEDURE book_delete_trigger();

INSERT INTO Book (BookName, Author, Publisher)
VALUES ('Book 1', 'Author 1', 'Publisher 1'),
       ('Book 2', 'Author 2', 'Publisher 2'),
       ('Book 3', 'Author 3', 'Publisher 3');

DELETE FROM Book WHERE BookID = 2;

SELECT * FROM Book_Del;
```

✧✧

PROGRAM 15

Create a procedure to print factorials of the numbers from 1 to 10

```
CREATE OR REPLACE FUNCTION print_factorials()
RETURNS VOID AS $$
DECLARE
n INTEGER;
factorial BIGINT;
BEGIN
FOR n IN 1..10 LOOP
  factorial := 1;
  FOR i IN 1..n LOOP
    factorial := factorial * i;
  END LOOP;
```

```
    RAISE NOTICE 'Factorial of % is %', n, factorial;

  END LOOP;
END;
$$ LANGUAGE plpgsql;
 Select print_factorials();
```