SURYA SEETHARAMAN AM.EN.U4CSE12501

DATABASE MANAGEMENT SYSTEMS LABSHEET 8

1. Write a function to do multiple updates.

```
create table log_tab
(
        str Varchar(20) primary key,
        str_length int
);

insert into log_tab values('hello',5);
insert into log_tab values('surya',5);
insert into log_tab values('anjana',6);
insert into log_tab values('darshana',7);
insert into log_tab values('anu',3);
insert into log_tab values('nil',0);
insert into log_tab values('tina',4);

select * from log_tab;
```

	str character varying(20)	str_length integer
1	hello	5
2	surya	5
3	anjana	6
4	darshana	7
5	anu	3
6	nil	0
7	tina	4

```
create or replace function logfunc1 (text) returns int as
'
      DECLARE
         logtxt ALIAS FOR $1;
      BEGIN
          alter table log_tab add column description Varchar(20);
         update log_tab set description=logtxt where str_length=5;
         RETURN str_length;
      END;
'

language 'plpgsql';

select logfunc1('string of length 5');
```

select * from log_tab;

	str character varying(20)	str_length integer	description character varying(20)
1	anjana	6	
2	darshana	7	
3	anu	3	
4	nil	0	
5	tina	4	
6	hello	5	string of length 5
7	surya	5	string of length 5

```
create or replace function logfunc1 (text,int) returns int as
'
        DECLARE
           logtxt ALIAS FOR $1;
           len ALIAS FOR $2;
        BEGIN
           update log_tab set description=logtxt where str_length=len;
           RETURN len;
        END;
'

language 'plpgsql';

select logfunc1('string of length 5',5);
select logfunc1('string of length 0',0);
select logfunc1('string of length 7',7);
select logfunc1('string of length 4',4);
select logfunc1('string of length 3',3);
select logfunc1('string of length 6',6);

select logfunc1('string of length 6',6);
```

	logfunc1 integer
1	6

select * from log_tab;

	str character varying(20)	str_length integer	description character varying(20)
1	anjana	6	string of length 6
2	anu	3	string of length 3
3	tina	4	string of length 4
4	darshana	7	string of length 7
5	nil	0	string of length 0
6	hello	5	string of length 5
7	surya	5	string of length 5

2. Create table instructor(ino,iname,salary,depname) and department(depname,budget)
Write a function to extract the count of instructors in a department whose name is passed as argument.
Write a select statement like
select depname,iname,salary from instructor121 where fn2(depname)>=5;

```
create table instructors5
(
        ino int primary key,
        iname Varchar(20),
        salary float,
        dept_name Varchar(20)
);

insert into instructors5 values(1,'surya',300000,'CSE');
insert into instructors5 values(2,'savita',40000,'MEC');
insert into instructors5 values(3,'vipin',55000,'CSE');
insert into instructors5 values(4,'shankar',200000,'EEE');
insert into instructors5 values(5,'bithin',660000,'MEC');
insert into instructors5 values(6,'giri',70000,'CSE');
insert into instructors5 values(7,'sowmya',320000,'EEE')

select * from instructors5;
```

	ino integer	iname character varying(20)	salary double precision	dept_name character varying(20)
1	1	surya	300000	CSE
2	2	savita	40000	MEC
3	3	vipin	55000	CSE
4	4	shankar	200000	EEE
5	5	bithin	660000	MEC
6	6	giri	70000	CSE
7	7	sowmya	320000	EEE

```
create table departments5
(
        dept_name Varchar(20) primary key,
        budget float
);

insert into departments5 values('CSE',50000);
insert into departments5 values('MEC',4000);
insert into departments5 values('EEE',20000);

select * from departments5;
```

	dept_name character varying(20)	budget double precision
1	CSE	50000
2	MEC	4000
3	EEE	20000

```
create or replace function inst_dept (text) returns int as
'
      DECLARE
         deptname ALIAS FOR $1;
         answer int;
      BEGIN
         select count(ino) into answer from instructors5 where dept_name=deptname;
         RETURN answer;
      END;
'

language 'plpgsql';
```

select inst_dept('CSE');

	inst_dept integer
1	3

select ino,depname,iname,salary from instructor5 where inst_dept(depname)>2;

	ino integer	iname character varying(20)	salary double precision	dept_name character varying(20)
1	1	surya	300000	CSE
3	3	vipin	55000	CSE
6	6	giri	70000	CSE

3. Create a table register
Write a function to register a student for the courses of a particular semester and year if he has currently acquired enough credits else raise exception.

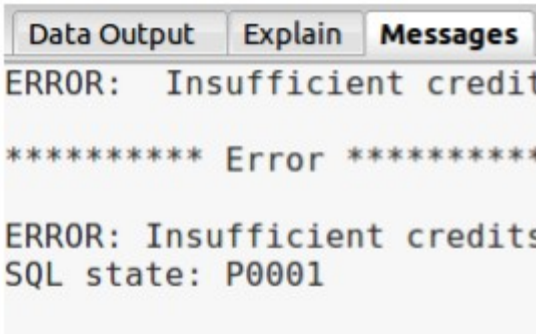
```
--FUNCTION TO CHECK CREDIT REQUIREMENTS
CREATE OR REPLACE FUNCTION credit_check(cr int,sem int) RETURNS boolean AS
$$
DECLARE
        credits int;
        res boolean;
BEGIN
        credits:=0;
        IF sem =1 then
        credits:=0;
        ELSIF sem=2 then
        credits:=22;
        ELSIF sem=3 then
        credits:=44;
        ELSIF sem=4 then
        credits:=88;
        ELSE
        res:=false;
        END IF;
        IF cr=credits then
        res=true;
        ELS e
```

```
        res:=false;
        END IF;
        return res;
        END;
$$
LANGUAGE plpgsql;


--select credit_check(22,3);
--FUNCTION TO REGISTER

CREATE OR REPLACE FUNCTION register_for_course(sid1 int,course
Registersample.course%type,semester int)   RETURNS boolean AS
$$
DECLARE
         res varchar(10);
        b boolean;
        credits int;
        current_year int;
BEGIN
        res:='Success';
        credits:=credits_earned from Studentsample where Studentsample.sid=sid1;
        b:= credit_check (credits,semester);
        current_year:=year1 from Studentsample where Studentsample.sid=sid1;
        IF b=true THEN
        Insert into Registersample values(sid1,semester, current_year, course);
        return b;
        END IF;
        IF b=false then
        RAISE EXCEPTION 'Insufficient credits';
        END IF;
        END;
$$
LANGUAGE plpgsql;


select register_for_course(3,'DBMS',2);
```



select register_for_course(1,'DBMS',2);
select register_for_course(2,'DBMS',2);

	sid integer	sem integer	year integer	course character varying(30)
1	1	2	2012	DBMS
2	2	2	2012	DBMS

4. Write a function to register a student to an elective course if the current number of students has not exceeded the limits.

The present Elective_Course is as follows:

	cid integer	ctitle character varying(30)	no_of_st integer	curr_st integer	sem integer
1	1	Organic Chemist	3	0	2
2	1	Electro Chemist	2	0	2
3	1	Medical Physics	20	0	2
4	1	E M Waves	5	0	2

--ASSUMING ALL ELECTIVE COURSES ARE AVAILABLE IN SEM 2 and MINIMUM CREDITS REQUIREMENT NEED NOT BE MET

```
CREATE OR REPLACE FUNCTION Elective_Reg(sid1 int,elective Registersample.course
%type) RETURNS text AS
$$
DECLARE
        res int;
        curr int;
        current_year int;
        semester int;
        result text;
BEGIN

        result='Seats are not available';
        res:=no_of_st from Elective_Course where ctitle=elective;
        curr:=curr_st from Elective_Course where ctitle=elective;
        current_year:=year1 from Studentsample where Studentsample.sid=sid1;
```

```
        semester:=sem from Elective_Course where ctitle=elective;
        IF curr<res then
        update Elective_Course set curr_st = curr_st+1 where ctitle=elective;
        IF curr<res then
        update Elective_Course set curr_st = curr_st+1 where ctitle=elective;
        insert into Registersample values(sid1,semester,current_year,elective);
        result='true';
        END IF;
        return result;
        END;
$$
LANGUAGE plpgsql;
```

select Elective_reg(1,'Electro Chemistry');
select Elective_reg(2,'Electro Chemistry');

	elective_reg text
1	true

select Elective_reg(3,'Electro Chemistry');

	elective_reg text
1	Seats are not available

now the Elective_Course table is as follows:

	cid integer	ctitle character varying(30)	no_of_st integer	curr_st integer	sem intege
1	1	Organic Chemist	3	0	2
2	1	Medical Physics	20	0	2
3	1	E M Waves	5	0	2
4	1	Electro Chemist	2	2	2

5. Write a function to find the avg marks of students in each course. If this average is more than 30, insert these course info to another table called course_above_avg.

creating table marks for recording all students marks in their respective subjects.

SURYA SEETHARAMAN AM.EN.U4CSE12501

```
create table marks
(
        sid varChar(30) references student,
        ccode varChar(15) references course,
        mark int, foreign key(sid, ccode) references table_register,
        primary key(sid, ccode)
);

select * from marks;
```

After inserting more records into table_register and marks, the tables are as follows:

table_register:

Output pane

Data Output | Explain | Messages | History

	sid character varying(30)	ccode character varying(15)	credits integer
1	u4cse12020	CSE300	4
2	u4cse12020	CSE310	3
3	u4cse12001	CSE300	4
4	u4cse12001	CSE340	3
5	u4cse12023	CSE340	3
6	u4cse12019	CSE340	3
7	u4cse12019	CSE300	4
8	u4cse12019	CSE310	3
9	u4cse12019	CSE320	4
10	u4cse12020	CSE320	4
11	u4cse12020	CSE340	3
12	u4cse12001	CSE310	3
13	u4cse12001	CSE320	4

table marks:

Output pane

Data Output | Explain | Messages | History

	sid character varying(30)	ccode character varying(15)	mark integer
1	u4cse12001	CSE300	35
2	u4cse12019	CSE300	40
3	u4cse12020	CSE300	38
4	u4cse12023	CSE300	25
5	u4cse12001	CSE310	30
6	u4cse12019	CSE310	35
7	u4cse12020	CSE310	25
8	u4cse12023	CSE310	40
9	u4cse12001	CSE320	40
10	u4cse12019	CSE320	38
11	u4cse12020	CSE320	35
12	u4cse12023	CSE320	30
13	u4cse12001	CSE340	35

average marks of each course:

select ccode, avg(mark) from marks group by ccode;

Output pane

Data Output | Explain | Messages | Histo

	ccode character varying(15)	avg numeric
1	CSE320	35.7500
2	CSE300	34.5000
3	CSE310	32.5000

Function to find the average marks of a particular course.

```
CREATE OR REPLACE FUNCTION FIND_AVG(text) RETURNS float AS
$$
	DECLARE
		COURSE_CODE ALIAS FOR $1;

	BEGIN
		return avg(mark) from marks where ccode = COURSE_CODE group by ccode;
	END;
$$
LANGUAGE plpgsql;
```

select FIND_AVG('CSE300') as Average;

Output pane

Data Output | Exp

	average text

creating table course_above_avg

create table course_above_avg(ccode varChar(15) references course primary key, avg_mark float);

inserting into the table course_above_avg the courses whose average is above 30.

insert into course_above_avg (select distinct ccode, FIND_AVG(ccode) from marks where FIND_AVG(ccode) > 30);

select * from course_above_avg;

SURYA SEETHARAMAN AM.EN.U4CSE12501

Output pane

Data Output | Explain | Messages | History

	ccode character varying(15)	avg_mark double precision
1	CSE300	34.5
2	CSE310	32.5