Assignment 3 – Stored Functions

Set A

1) Project-Employee Database

Consider the following Entities and their Relationships for ProjectEmployee database.

Project (pno integer, pname char (30), ptype char (20), duration integer)

Employee (eno integer, ename char (20), qualification char (15), joining_date date)

Relationship between Project and Employee is many to many with

descriptive attribute start_date date, no_of_hours_worked integer.

Constraints: Primary Key,

duration should be greater than zero, pname should not be null.

Queries:

1. Write a stored function to find the number of employees whose joining date is before '01/01/2007'.

2. Write a stored function to accept eno as input parameter and count number of projects on which that employee is working.

3. Write a stored function to accept project name and display employee details who worked more than 2000 hours.

4. Write a stored function to display all projects started after date "01/01/2019".

Solution

Creating DB:

```
create table project(
 pno int primary key,
 pname varchar(30) NOT NULL,
 ptype varchar(20),
 duration int check (duration > 0)
);
insert into project values (1,'ERP','Management',10);
insert into project values (2,'SAP','Civil',30);
insert into project values (3,'TTT','Manufacturing',4);
insert into project values (4,'TOM','Civil',5);
insert into project values (5,'HTI','Management',7);
```

```
insert into project values (6,'X-T','Civil',50);
create table employee(
 eno int primary key,
 ename varchar(20),
 qualification varchar(15),
 joining_date date
);
insert into employee values (11,'John','MCA','2020-09-01');
insert into employee values (12,'Smith','MCS','1998-10-01');
insert into employee values (13,'Ron','BBA-CA','2000-03-01');
insert into employee values (14,'Roy','MCA','2003-12-01');
insert into employee values (15,'Alex','MCS','2000-11-10');
insert into employee values (16,'David','BBA-CA','2001-08-01');
insert into employee values (17,'Ben','MCS','1990-10-10');
insert into employee values (18,'Sam','MCA','1995-12-12');
create table pro_emp(
 pno int references project(pno)
on delete cascade on update cascade,
 eno int references employee(eno)
on delete cascade on update cascade,
 start_date date,
 no_of_hrs int
);
insert into pro_emp values (1,11,'2020-09-10',20);
insert into pro_emp values (2,12,'2000-10-12',700);
insert into pro_emp values (3,13,'2010-06-06',400);
insert into pro_emp values (4,14,'2000-10-16',350);
insert into pro_emp values (5,15,'1999-01-20',250);
insert into pro_emp values (1,16,'2000-10-12',25);
insert into pro_emp values (2,12,'2000-10-12',800);
insert into pro_emp values (3,13,'2010-06-06',320);
```

```
insert into pro_emp values (6,17,'1995-12-23',2010);
insert into pro_emp values (6,18,'1995-12-23',2010);
```

DATABASE:

```
select * from employee;
 eno | ename | qualification | joining_date
-----+-------+---------------+--------------
 11 | John | MCA | 2020-09-01
 12 | Smith | MCS | 1998-10-01
 13 | Ron | BBA-CA | 2000-03-01
 14 | Roy | MCA | 2003-12-01
 15 | Alex | MCS | 2000-11-10
 16 | David | BBA-CA | 2001-08-01
 17 | Ben | MCS | 1990-10-10
 18 | Sam | MCA | 1995-12-12
(8 rows)
select * from project;
 pno | pname | ptype | duration
-----+-------+---------------+----------
 1 | ERP | Management | 10
 2 | SAP | Civil | 30
 3 | TTT | Manufacturing | 4
 4 | TOM | Civil | 5
 5 | HTI | Management | 7
 6 | X-T | Civil | 50
(6 rows)
select * from pro_emp;
 pno | eno | start_date | no_of_hrs
-----+-----+------------+-----------
 1 | 11 | 2020-09-10 | 20
 2 | 12 | 2000-10-12 | 700
 3 | 13 | 2010-06-06 | 400
```

4 | 14 | 2000-10-16 | 350

5 | 15 | 1999-01-20 | 250

1 | 16 | 2000-10-12 | 25

2 | 12 | 2000-10-12 | 800

3 | 13 | 2010-06-06 | 320

6 | 17 | 1995-12-23 | 2010

6 | 18 | 1995-12-23 | 2010

(10 rows)

1. Write a stored function to find the number of employees whose joining date is before '01/01/2007'.

```
create or replace function noofemp() returns int as
'
 declare
 cnt int;
 begin
 select count(*) into cnt
 from employee
 where joining_date < ''2007-01-01'';
 return cnt;
 end;
'language 'plpgsql';
select noofemp();
 noofemp
---------
 7
(1 row)
```

2. Write a stored function to accept eno as input parameter and count number of projects on which that employee is working.

```
create or replace function countp(n int) returns int as
'
 declare
```

```
 cnt int;
 begin
 select count(*) into cnt
 from pro_emp
 where eno = n;
 return cnt;
 end;
'language 'plpgsql';
select countp(12);
 countp
--------
 2
(1 row)
```

3. Write a stored function to accept project name and display employee details who worked more than 2000 hours.

```
create or replace function empwork(name text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''ENO || Name || Qualification ||
 Joining Date'';

 for rec in
 select employee.eno,ename,qualification,joining_date
 from employee,project,pro_emp
 where employee.eno = pro_emp.eno
 and project.pno = pro_emp.pno
 and pname = name loop
 raise notice ''% || % || % ||
```

```
 %'',rec.eno,rec.ename,rec.qualification,rec.joining_date;
 end loop;
 end;
'language 'plpgsql';
select empwork('X-T');
NOTICE: ENO||Name||Qualification||Joining Date
NOTICE: 17||Ben||MCS||1990-10-10
NOTICE: 18||Sam||MCA||1995-12-12
 empwork
---------
(1 row)
```

4. Write a stored function to display all projects started after date “01/01/2019”.

```
create or replace function pstart() returns void as
'
 declare
 name text;
 begin
 raise notice ''Project Name'';
 for name in
 select pname
 from project,pro_emp
 where project.pno = pro_emp.pno
 and start_date > ''2019-01-01'' loop
 raise notice ''%'',name;
 end loop;
 end;
'language 'plpgsql';
select pstart();
NOTICE: Project Name
NOTICE: ERP
 pstart
```

(1 row)

2) Person-Area Database

Consider the following Entities and their Relationships for PersonArea database.

Person (pno integer, pname varchar (20), birthdate date, income money)

Area (aname varchar (20), area_type varchar (5))

An area can have one or more persons living in it, but a person belongs to

exactly one area.

Constraints: Primary Key,area_type can be either ‘urban’ or ‘rural’.

Queries:

1. Write a stored function to print total number of persons of a particular area. Accept area name as input parameter.

2. Write a stored function to update the income of all persons living in urban area by 20%.

3. Write a stored function to accept area_type and display person’s details area wise.

4. Write a stored function to accept area name and display all persons having age more than 60.

Solution

Creating DB:

```
create table area(
 aname varchar(20) primary key,
 area_type varchar(5) check (area_type in('rural','urban'))
);
insert into area values ('Hadapsar','urban');
insert into area values ('Mandai','urban');
insert into area values ('Wagholi','rural');
insert into area values ('Loni','rural');
create table person(
 pno int primary key,
 pname varchar(20),
 birthdate date,
```

```
 income numeric(10,2),
 age int,
 aname varchar(20) references area(aname)
on delete cascade on update cascade
);
insert into person values (1,'Narayan Tupe','1990-08-11',20000,30,'Hadapsar');
insert into person values (2,'Anil Wagh','1958-01-20',30000,62,'Mandai');
insert into person values (3,'Sumit Patil','1955-12-25',20000,65,'Wagholi');
insert into person values (4,'Kavya Chavan','1998-04-18',40000,22,'Hadapsar');
insert into person values (5,'Deep Das','1994-12-30',40000,26,'Loni');
insert into person values (6,'Ashok Deshmukh','1992-08-23',30000,28,'Loni');
```

DATABASE:

```
select * from area;
 aname | area_type
----------+-----------
 Hadapsar | urban
 Mandai | urban
 Wagholi | rural
 Loni | rural
(4 rows)
select * from person;
 pno | pname | birthdate | income | age | aname
-----+---------------+------------+----------+-----+----------
 1 | Narayan Tupe | 1990-08-11 | 20000.00 | 30 | Hadapsar
 2 | Anil Wagh | 1958-01-20 | 30000.00 | 62 | Mandai
 3 | Sumit Patil | 1955-12-25 | 20000.00 | 65 | Wagholi
 4 | Kavya Chavan | 1998-04-18 | 40000.00 | 22 | Hadapsar
 5 | Deep Das | 1994-12-30 | 40000.00 | 26 | Loni
 6 | Ashok Deshmukh | 1992-08-23 | 30000.00 | 28 | Loni
(6 rows)
```

1. Write a stored function to print total number of persons of a particular area. Accept area name as input parameter.

```
create or replace function cntperson(name text) returns int as
'
 declare
 cnt int;
 begin
 select count(*) into cnt
 from person
 where aname = name;

 return cnt;
 end;
'language 'plpgsql';
select cntperson('Hadapsar');
 cntperson
-----------
 2
(1 row)
```

2. Write a stored function to update the income of all persons living in urban area by 20%.

```
create or replace function updateincome() returns void as
'
 begin
 update person
 set income = income+(income*20/100)
 where aname in
 (
 select aname
 from area
 where area_type = ''urban''
 );
```

```
 raise notice ''Database Updated'';
 end;
'language 'plpgsql';
select updateincome();
NOTICE: Database Updated
 updateincome
--------------
(1 row)
select * from person;
 pno | pname | birthdate | income | age | aname
-----+---------------+------------+----------+-----+----------
 3 | Sumit Patil | 1955-12-25 | 20000.00 | 65 | Wagholi
 5 | Deep Das | 1994-12-30 | 40000.00 | 26 | Loni
 6 | Ashok Deshmukh | 1992-08-23 | 30000.00 | 28 | Loni
 1 | Narayan Tupe | 1990-08-11 | 24000.00 | 30 | Hadapsar
 2 | Anil Wagh | 1958-01-20 | 36000.00 | 62 | Mandai
 4 | Kavya Chavan | 1998-04-18 | 48000.00 | 22 | Hadapsar
(6 rows)
```

3. Write a stored function to accept area_type and display person's details area wise.

```
create or replace function printperson(atype text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''PNO || Name || Birthdate ||
 Income || Age'';
 for rec in
 select person.pno,pname,birthdate,income,age
 from person,area
 where person.aname = area.aname
 and area_type = atype loop
```

```
 raise notice ''% || % || % || % ||
 %'',rec.pno,rec.pname,rec.birthdate,rec.income,rec.age;
 end loop;
 end;
'language 'plpgsql';
select printperson('urban');
NOTICE: PNO||Name||Birthdate||Income||Age
NOTICE: 1||Narayan Tupe||1990-08-11||24000.00||30
NOTICE: 2||Anil Wagh||1958-01-20||36000.00||62
NOTICE: 4||Kavya Chavan||1998-04-18||48000.00||22
 printperson
-------------
(1 row)
```

4. Write a stored function to accept area name and display all persons having age more than 60.

```
create or replace function findperson(name text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''PNO || Name || Birthdate ||
 Income || Age'';
 for rec in
 select *
 from person
 where aname = name
 and age > 60 loop
 raise notice ''% || % || % || % ||
 %'',rec.pno,rec.pname,rec.birthdate,rec.income,rec.age;
 end loop;
 end;
'language 'plpgsql';
```

select findperson('Mandai');

NOTICE: PNO||Name||Birthdate||Income||Age

NOTICE: 2||Anil Wagh||1958-01-20||36000.00||62

findperson

(1 row)

SET B

1) Bus Transport Database

Consider the following Entities and their Relationships for Bus Transport database.

Bus (bus_no int ,b_capacity int , depot_name varchar(20))

Route (route_no int, source char (20), destination char (20), no_of_stations int)

Driver (driver_no int ,driver_name char(20), license_no int, address char(20), d_age int , salary float)

Relationship between Bus and Route is many to one and relationship between Bus and Driver is many to many with descriptive attributes date_of_duty_allotted and shift.

Constraints: Primary Key, license_no is unique, b_capacity should not be

null, shift can be 1 (Morning) or 2(Evening).

Queries:

1. Write a stored function to accept route no and display bus information running on that route.

2. Write a stored function to accept shift and depot name and display driver details who having duty allocated after '01/07/2020'.

3. Write a stored function to accept source name and display count of buses running from source place.

4. Write a stored function to accept depot name and display driver details having age more than 50.

Solution

Creating DB:

```
create table driver(
 driver_no int primary key,
 driver_name char(20),
 license_no int unique,
 address char(20),
 driver_age int,
```

```
 salary float
);
insert into driver values(1,'ganesh',101,'satara_road',34,10000);
insert into driver values(2,'sunil',102,'swarget',55,12000);
insert into driver values(3,'sagar',103,'anandnagar',52,15000);
create table route(
 route_no int primary key,
 source char(20),
 destination char(20),
 no_of_stations int
);
insert into route values(11,'deccan','katraj',6);
insert into route values(12,'vadgoan','mandai',7);
insert into route values(13,'dhayari','shanipar',5);
create table bus(
 bus_no int primary key,
 capacity int not null,
 depot_name varchar(20),
 route_no int references route(route_no)
on delete cascade on update cascade
);
insert into bus values(10,45,'kothrud',11);
insert into bus values(56,44,'corporation',11);
insert into bus values(57,43,'Katraj',12);
create table bus_driver(
 bus_no int references bus,
 driver_no int references driver
on delete cascade on update cascade,
 date_of_duty date,
 shift int check(shift in(1,2))
);
```

```
insert into bus_driver values(10,1,'2020/07/02',2);
insert into bus_driver values(56,2,'2016/06/06',1);
insert into bus_driver values(57,3,'2016/10/06',2);
```

DATABASE:

```
select * from driver;
 driver_no | driver_name | license_no | address |
driver_age | salary
-----------+---------------------+------------+---------------------+----
--------+--------
 1 | ganesh | 101 | satara_road |
34 | 10000
 2 | sunil | 102 | swarget |
55 | 12000
 3 | sagar | 103 | anandnagar |
52 | 15000
(3 rows)
select * from route;
 route_no | source | destination | no_of_stations
----------+---------------------+---------------------+----------------
 11 | deccan | katraj | 6
 12 | vadgoan | mandai | 7
 13 | dhayari | shanipar | 5
(3 rows)
select * from bus;
 bus_no | capacity | depot_name | route_no
--------+----------+-------------+----------
 10 | 45 | kothrud | 11
 56 | 44 | corporation | 11
 57 | 43 | Katraj | 12
(3 rows)
```

```
select * from bus_driver;
 bus_no | driver_no | date_of_duty | shift
--------+-----------+--------------+-------
 10 | 1 | 2020-07-02 | 2
 56 | 2 | 2016-06-06 | 1
 57 | 3 | 2016-10-06 | 2
(3 rows)
```

1. Write a stored function to accept route no and display bus information running on that route.

```
create or replace function findbus(routeno int) returns void as
'
 declare
 rec record;
 begin
 raise notice ''Bus No || Capacity || Depo Name'';

 for rec in
 select *
 from bus
 where route_no = routeno loop
 raise notice ''% || % ||
 %'',rec.bus_no,rec.capacity,rec.depot_name;
 end loop;
 end;
'language 'plpgsql';
select findbus(11);
NOTICE: Bus No||Capacity||Depo Name
NOTICE: 10||45||kothrud
NOTICE: 56||44||corporation
 findbus
---------
```

(1 row)

2. Write a stored function to accept shift and depot name and display driver details who having duty allocated after '01/07/2020'.

```
create or replace function finddriver(sh int,name text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''Driver No || Name || License No ||
 Address || Age || Salary'';
 for rec in
 select
driver.driver_no,driver_name,license_no,address,driver_age,salary
 from bus,bus_driver,driver
 where bus.bus_no = bus_driver.bus_no
 and driver.driver_no = bus_driver.driver_no
 and date_of_duty > ''2020-07-01''
 and shift = sh
 and depot_name = name loop
 raise notice ''% || % || % || % ||
 % || %''
 ,rec.driver_no,rec.driver_name,rec.license_no,rec.address,rec.driver_
age,rec.salary;
 end loop;
 end;
'language 'plpgsql';
select finddriver(2,'kothrud');
NOTICE: Driver No||Name||License No||Address||Age||Salary
NOTICE: 1||ganesh ||101||satara_road ||34||10000
 finddriver
```

(1 row)

3. Write a stored function to accept source name and display count of buses running from source place.

```
create or replace function countbus(sname text) returns int as
'
 declare
 cnt int;
 begin
 select count(*) into cnt
 from bus
 where route_no in(
 select route_no
 from route
 where source = sname
 );
 return cnt;
 end;
'language 'plpgsql';
select countbus('deccan');
 countbus
----------
 2
(1 row)
```

4. Write a stored function to accept depot name and display driver details having age more than 50.

```
create or replace function getdriver(dname text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''Driver No || Name || License No ||
```

```
 Address || Age || Salary'';
  for rec in
 select driver.driver_no,driver_name,license_no,address,driver_age,salary
 from bus,driver,bus_driver where bus.bus_no = bus_driver.bus_no
 and driver.driver_no = bus_driver.driver_no
 and depot_name = dname
 and driver_age > 50 loop
 raise notice ''% || % || % || % || % || %''
,rec.driver_no,rec.driver_name,rec.license_no,rec.address,rec.driver_
age,rec.salary;
 end loop;
 end;
'language 'plpgsql';
select getdriver('Katraj');
NOTICE: Driver No||Name||License No||Address||Age||Salary
NOTICE: 3||sagar ||103||anandnagar ||52||15000
 getdriver
-----------
(1 row)
```

2) Bank Database

Consider the following Entities and their Relationships for Bank database.

Branch (br_id integer, br_name char (30), br_city char (10))

Customer (cno integer, c_name char (20), caddr char (35), city char (20))

Loan_application(lno integer, l_amt_required money, l_amt_approved money, l_date date)

Relationship between Branch, Customer and Loan_application is Ternary.

Ternary (br_id integer, cno integer, lno integer)

Constraints: Primary Key,

l_amt_required should be greater than zero.

Queries:

1. Write a stored function to accept branch name and display customer

details whose loan amount required is more than loan approved.

2. Write a stored function to accept branch name and display customer name, loan number, loan amount approved on or after 01/06/2019.

3. Write a stored function to display total loan amount approved by all branches after date 30/05/2019.

4. Write a stored function to display customer details who have applied for loan more than one branches.

Solution

Creating DB:

```
create table branch(
 bid int primary key,
 brname char(30),
 brcity char(10)
);
insert into branch values(1,'mg Road','pune');
insert into branch values(2,'deccan','mumbai');
insert into branch values(3,'aundh','solapur');
insert into branch values(4,'swarget','nagpur');
insert into branch values(5,'parvati','konkan');
create table customer(
 cno int primary key,
 cname char(20),
 caddr char(35),
 city char(20)
);
insert into customer values(101,'raj','satara road','pune');
insert into customer values(102,'shreya','nagar road','solapur');
insert into customer values(103,'jaya','sahakar nagar','konkan');
insert into customer values(105,'sunny','kothrud','pune');
insert into customer values(106,'joya','swarget','Nagpur');
```

```
insert into customer values(107,'taran','parvati','konkan');
insert into customer values(108,'neha','aundh','solapur');
create table loan_application(
 lno int primary key,
 lamtrequired numeric(10,2),
 lamtapproved numeric(10,2),
 l_date date
);
insert into loan_application values(1111,300000,250000,'2019-01-02');
insert into loan_application values(2222,400000,300000,'2019-12-15');
insert into loan_application values(3333,200000,200000,'2019-05-23');
insert into loan_application values(4444,10000,15000,'2020-02-02');
insert into loan_application values(5555,900000,900000,'2018-01-01');
insert into loan_application values(6666,500000,200000,'2020-01-01');
create table ternary(
 bid int references branch(bid)
on delete cascade on update cascade,
 cno int references customer(cno)
on delete cascade on update cascade,
 lno int references loan_application(lno)
on delete cascade on update cascade
);
insert into ternary values(1,101,1111);
insert into ternary values(2,103,3333);
insert into ternary values(3,102,5555);
insert into ternary values(4,105,4444);
insert into ternary values(5,101,2222);
insert into ternary values(5,102,6666);
DATABASE:
select * from branch;
 bid | brname | brcity
```

-----+---------------------------------+------------

1 | mg Road | pune

2 | deccan | mumbai

3 | aundh | solapur

4 | swarget | nagpur

5 | parvati | konkan

(5 rows)

select * from customer;

cno | cname | caddr |

city

-----+---------------------+-----------------------------------+--------

----

101 | raj | satara road | pune

102 | shreya | nagar road | solapur

103 | jaya | sahakar nagar | konkan

105 | sunny | kothrud | pune

106 | joya | swarget | Nagpur

107 | taran | parvati | konkan

108 | neha | aundh | solapur

(7 rows)

select * from loan_application;

lno | lamtrequired | lamtapproved | l_date

------+--------------+--------------+------------

1111 | 300000.00 | 250000.00 | 2019-01-02

2222 | 400000.00 | 300000.00 | 2019-12-15

3333 | 200000.00 | 200000.00 | 2019-05-23

4444 | 10000.00 | 15000.00 | 2020-02-02

5555 | 900000.00 | 900000.00 | 2018-01-01

6666 | 500000.00 | 200000.00 | 2020-01-01

(6 rows)

select * from ternary;

```
 bid | cno | lno
-----+-----+------
 1 | 101 | 1111
 2 | 103 | 3333
 3 | 102 | 5555
 4 | 105 | 4444
 5 | 101 | 2222
 5 | 102 | 6666
(6 rows)
```

1. Write a stored function to accept branch name and display customer details whose loan amount required is more than loan approved.

```
create or replace function getdetails(name text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''Cust.No || Name || Address ||
 City'';
 for rec in
 select customer.cno, cname, caddr, city
 from customer,branch,ternary,loan_application
 where customer.cno = ternary.cno
 and branch.bid = ternary.bid
 and loan_application.lno = ternary.lno
 and lamtrequired > lamtapproved
 and brname = name loop
 raise notice ''% || % || % ||
 %'',rec.cno,rec.cname,rec.caddr,rec.city;
 end loop;
 end;
```

'language 'plpgsql';

select getdetails('parvati');

NOTICE: Cust.No||Name||Address||City

NOTICE: 101||raj ||satara road

||pune

NOTICE: 102||shreya ||nagar road

||solapur

getdetails

(1 row)

2. Write a stored function to accept branch name and display customer name, loan number, loan amount approved on or after 01/06/2019.

```
create or replace function getcname(name text) returns void as
'
 declare
 rec record;
 begin
 raise notice ''Name || Loan No. || Loan Amt.
Approved'';
 for rec in
 select cname, ternary.lno, lamtapproved
 from customer,ternary,loan_application,branch
 where customer.cno = ternary.cno
 and loan_application.lno = ternary.lno
 and branch.bid = ternary.bid
 and l_date > ''2019-06-01''
 and brname = name loop
 raise notice ''% || % ||
 %'',rec.cname,rec.lno,rec.lamtapproved;
 end loop;
```

```
 end;
'language 'plpgsql';
select getcname('parvati');
NOTICE: Name||Loan No.||Loan Amt. Approved
NOTICE: raj ||2222||300000.00
NOTICE: shreya ||6666||200000.00
 getcname
----------
(1 row)
```

3. Write a stored function to display total loan amount approved by all branches after date 30/05/2019.

```
create or replace function gettotal() returns numeric as
'
 declare
 total loan_application.lamtapproved%type;
 begin
 select sum(lamtapproved) into total
 from branch,loan_application,ternary
 where branch.bid = ternary.bid
 and loan_application.lno = ternary.lno
 and l_date > ''2019-05-30'';
 return total;
 end;
'language 'plpgsql';
select gettotal();
 gettotal
-----------
515000.00
(1 row)
```