

## Assignment on Function

SET A (Number of Slots – 2)

1) Project-Employee Database

Consider the following Entities and their Relationships for Project-Employee 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.

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".

Ans

Certainly! Below are the stored functions for the given requirements:

1. \*\*Number of employees whose joining date is before '01/01/2007'.\*\*

```
```sql
CREATE OR REPLACE FUNCTION EmployeeCountBeforeDate()
RETURNS INTEGER AS $$
DECLARE
    emp_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO emp_count
    FROM Employee
    WHERE joining_date < '2007-01-01';

    RETURN emp_count;
END;
$$ LANGUAGE PLPGSQL;
```
```

2. \*\*Number of projects on which a specific employee is working (Accepts eno as input parameter).\*\*

```
```sql
CREATE OR REPLACE FUNCTION ProjectCountForEmployee(emp_id INTEGER)
RETURNS INTEGER AS $$
DECLARE
    proj_count INTEGER;
BEGIN
```

```

SELECT COUNT(*)
INTO proj_count
FROM ProjectEmployee
WHERE eno = emp_id;

RETURN proj_count;
END;
$$ LANGUAGE PLPGSQL;
```

```

3. **\*\*Employee details who worked more than 2000 hours on a specific project (Accepts project name).\*\***

```

```sql
CREATE OR REPLACE FUNCTION HighHourEmployeesForProject(proj_name VARCHAR)
RETURNS TABLE (eno INTEGER, ename CHAR(20), hours_worked INTEGER) AS $$
BEGIN
    RETURN QUERY
    SELECT e.eno, e.ename, pe.no_of_hours_worked
    FROM Employee e
    JOIN ProjectEmployee pe ON e.eno = pe.eno
    JOIN Project p ON pe.pno = p.pno
    WHERE p.pname = proj_name AND pe.no_of_hours_worked > 2000;
END;
$$ LANGUAGE PLPGSQL;
```

```

4. **\*\*Projects started after date '01/01/2019'.**

```

```sql
CREATE OR REPLACE FUNCTION ProjectsAfterDate()
RETURNS TABLE (pno INTEGER, pname CHAR(30), ptype CHAR(20), start_date DATE) AS
$$
BEGIN
    RETURN QUERY
    SELECT pno, pname, ptype, start_date
    FROM Project
    WHERE start_date > '2019-01-01';
END;
$$ LANGUAGE PLPGSQL;
```

```

You can use these functions in your PostgreSQL environment. Example usage:

1. `SELECT EmployeeCountBeforeDate();`
2. `SELECT ProjectCountForEmployee(123);`
3. `SELECT \* FROM HighHourEmployeesForProject('ProjectABC');`
4. `SELECT \* FROM ProjectsAfterDate();`

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.

## 2) Person-Area Database

Consider the following Entities and their Relationships for Person-Area 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'.

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.

Ans

Certainly! Below are the stored functions for the given requirements:

1. \*\*Total number of persons of a particular area (Accepts area name as an input parameter).\*\*

```
```sql
CREATE OR REPLACE FUNCTION PersonCountForArea(area_name VARCHAR)
RETURNS INTEGER AS $$
DECLARE
    person_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO person_count
    FROM Person p
    JOIN Area a ON p.area_type = a.area_type
    WHERE a.aname = area_name;

    RETURN person_count;
END;
$$ LANGUAGE PLPGSQL;
```
```

2. \*\*Update the income of all persons living in urban areas by 20%.\*\*

```
```sql
CREATE OR REPLACE FUNCTION UpdateIncomeUrbanArea()
RETURNS VOID AS $$
BEGIN
    UPDATE Person
    SET income = income * 1.20
    WHERE area_type = 'urban';
END;
$$ LANGUAGE PLPGSQL;
```
```

3. \*\*Display person's details area-wise (Accepts area\_type as an input parameter).\*\*

```

```sql
CREATE OR REPLACE FUNCTION PersonDetailsForArea(area_type_param VARCHAR)
RETURNS TABLE (pno INTEGER, pname VARCHAR, birthdate DATE, income MONEY) AS
$$
BEGIN
    RETURN QUERY
    SELECT pno, pname, birthdate, income
    FROM Person
    WHERE area_type = area_type_param;
END;
$$ LANGUAGE PLPGSQL;
```

```

4. \*\*Display all persons having age more than 60 in a specific area (Accepts area name as an input parameter).\*\*

```

```sql
CREATE OR REPLACE FUNCTION ElderlyPersonsForArea(area_name VARCHAR)
RETURNS TABLE (pno INTEGER, pname VARCHAR, birthdate DATE, income MONEY) AS
$$
BEGIN
    RETURN QUERY
    SELECT pno, pname, birthdate, income
    FROM Person p
    JOIN Area a ON p.area_type = a.area_type
    WHERE a.aname = area_name AND EXTRACT(YEAR FROM AGE(NOW(), p.birthdate)) >
60;
END;
$$ LANGUAGE PLPGSQL;
```

```

You can use these functions in your PostgreSQL environment. Example usage:

1. `SELECT PersonCountForArea('UrbanArea');`
2. `SELECT UpdateIncomeUrbanArea();`
3. `SELECT \* FROM PersonDetailsForArea('urban');`
4. `SELECT \* FROM ElderlyPersonsForArea('RuralArea');`

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.

#### 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).

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.

Ans

Certainly! Below are the stored functions for the given requirements:

1. **\*\*Display bus information running on a specific route (Accepts route number as an input parameter).\*\***

```
```sql
CREATE OR REPLACE FUNCTION BusInfoForRoute(route_number INT)
RETURNS TABLE (bus_no INT, b_capacity INT, depot_name VARCHAR) AS $$
BEGIN
    RETURN QUERY
    SELECT b.bus_no, b.b_capacity, b.depot_name
    FROM Bus b
    JOIN Route r ON b.route_no = r.route_no
    WHERE r.route_no = route_number;
END;
$$ LANGUAGE PLPGSQL;
```
```

2. **\*\*Display driver details for a specific shift and depot name after '01/07/2020'.**

```
```sql
CREATE OR REPLACE FUNCTION DriversForShiftAndDepot(shift_value INT,
depot_name_param VARCHAR)
RETURNS TABLE (driver_no INT, driver_name VARCHAR, license_no INT, address
VARCHAR, d_age INT, salary FLOAT) AS $$
BEGIN
    RETURN QUERY
    SELECT DISTINCT d.*
    FROM Driver d
    JOIN BusDriver bd ON d.driver_no = bd.driver_no
    WHERE bd.shift = shift_value AND bd.date_of_duty_allotted > '2020-07-01' AND
b.depot_name = depot_name_param;
END;
$$ LANGUAGE PLPGSQL;
```
```

3. **\*\*Count of buses running from a specific source place (Accepts source name as an input parameter).\*\***

```
```sql
CREATE OR REPLACE FUNCTION BusCountFromSource(source_name VARCHAR)
```

```

RETURNS INTEGER AS $$
DECLARE
    bus_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO bus_count
    FROM Bus b
    JOIN Route r ON b.route_no = r.route_no
    WHERE r.source = source_name;

    RETURN bus_count;
END;
$$ LANGUAGE PLPGSQL;

```

4. **\*\*Display driver details having age more than 50 for a specific depot name.\*\***

```

```sql
CREATE OR REPLACE FUNCTION SeniorDriversForDepot(depot_name_param VARCHAR)
RETURNS TABLE (driver_no INT, driver_name VARCHAR, license_no INT, address
VARCHAR, d_age INT, salary FLOAT) AS $$
BEGIN
    RETURN QUERY
    SELECT *
    FROM Driver
    WHERE d_age > 50 AND depot_name = depot_name_param;
END;
$$ LANGUAGE PLPGSQL;

```

You can use these functions in your PostgreSQL environment. Example usage:

1. ``SELECT * FROM BusInfoForRoute(123);``
2. ``SELECT * FROM DriversForShiftAndDepot(1, 'DepotABC');``
3. ``SELECT BusCountFromSource('CityXYZ');``
4. ``SELECT * FROM SeniorDriversForDepot('DepotXYZ');``

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.

## 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.

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.

.

Ans

Certainly! Below are the stored functions for the given requirements:

1. \*\*Display customer details whose loan amount required is more than loan approved for a specific branch.\*\*

```
```sql
CREATE OR REPLACE FUNCTION
OverRequestedLoanDetailsForBranch(branch_name_param VARCHAR)
RETURNS TABLE (cno INT, c_name VARCHAR, lno INT, l_amt_required MONEY,
l_amt_approved MONEY, l_date DATE) AS $$
BEGIN
    RETURN QUERY
    SELECT cno, c_name, lno, l_amt_required, l_amt_approved, l_date
    FROM Ternary
    WHERE br_name = branch_name_param AND l_amt_required > l_amt_approved;
END;
$$ LANGUAGE PLPGSQL;
```
```

2. \*\*Display customer name, loan number, and loan amount approved on or after '01/06/2019' for a specific branch.\*\*

```
```sql
CREATE OR REPLACE FUNCTION
ApprovedLoanDetailsAfterDateForBranch(branch_name_param VARCHAR)
RETURNS TABLE (c_name VARCHAR, lno INT, l_amt_approved MONEY) AS $$
BEGIN
    RETURN QUERY
    SELECT c.c_name, t.lno, t.l_amt_approved
    FROM Ternary t
    JOIN Customer c ON t.cno = c.cno
    WHERE t.br_name = branch_name_param AND t.l_date >= '2019-06-01';
END;
$$ LANGUAGE PLPGSQL;
```
```

3. \*\*Display total loan amount approved by all branches after date '30/05/2019'.\*\*

```
```sql
CREATE OR REPLACE FUNCTION TotalLoanAmountApprovedAfterDate()
RETURNS MONEY AS $$
DECLARE
```

```

        total_approved MONEY;
BEGIN
    SELECT SUM(l_amt_approved)
    INTO total_approved
    FROM Ternary
    WHERE l_date > '2019-05-30';

    RETURN total_approved;
END;
$$ LANGUAGE PLPGSQL;
```

```

4. **\*\*Display customer details who have applied for a loan at more than one branch.\*\***

```

```sql
CREATE OR REPLACE FUNCTION CustomersWithMultipleLoanApplications()
RETURNS TABLE (cno INT, c_name VARCHAR, num_branches INT) AS $$
BEGIN
    RETURN QUERY
    SELECT c.cno, c.c_name, COUNT(DISTINCT t.br_id) AS num_branches
    FROM Customer c
    JOIN Ternary t ON c.cno = t.cno
    GROUP BY c.cno, c.c_name
    HAVING COUNT(DISTINCT t.br_id) > 1;
END;
$$ LANGUAGE PLPGSQL;
```

```

You can use these functions in your PostgreSQL environment. Example usage:

1. `SELECT \* FROM OverRequestedLoanDetailsForBranch('BranchXYZ');`
2. `SELECT \* FROM ApprovedLoanDetailsAfterDateForBranch('BranchABC');`
3. `SELECT TotalLoanAmountApprovedAfterDate();`
4. `SELECT \* FROM CustomersWithMultipleLoanApplications();`

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.

#### 1) Business trip database

Consider Business trip database that keeps track of the business trips of salesman in an office. Following are the tables:

Salesman (sno integer, s\_name char (30), start\_year integer, dept\_no varchar(10))

Trip(tno integer, from\_city char (20), to\_citychar (20),departure\_date date, return\_date date)

Dept(dept\_no varchar (10), dept\_name char(20))

Expense(eid integer, amount money)

Relationships:

Dept-Salesman: 1 to M

Salesman-Trip: 1 to M

Trip-Expense: 1 to 1

Execute the following stored functions.

- a) Write a stored function to find a business trip having maximum expenses.



b) Write a stored function to count the total number of business trips from 'Pune' to 'Mumbai'.

Ans

Certainly! Below are the stored functions for the given requirements:

a) \*\*Find a business trip having the maximum expenses.\*\*

```
```sql
CREATE OR REPLACE FUNCTION MaxExpenseBusinessTrip()
RETURNS TABLE (tno INTEGER, from_city CHAR(20), to_city CHAR(20), departure_date
DATE, return_date DATE, max_expense MONEY) AS $$
BEGIN
    RETURN QUERY
    SELECT t.tno, t.from_city, t.to_city, t.departure_date, t.return_date, e.amount AS
max_expense
    FROM Trip t
    JOIN Expense e ON t.tno = e.tno
    ORDER BY e.amount DESC
    LIMIT 1;
END;
$$ LANGUAGE PLPGSQL;
```
```

b) \*\*Count the total number of business trips from 'Pune' to 'Mumbai'.\*\*

```
```sql
CREATE OR REPLACE FUNCTION CountTripsFromPuneToMumbai()
RETURNS INTEGER AS $$
DECLARE
    trip_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO trip_count
    FROM Trip
    WHERE from_city = 'Pune' AND to_city = 'Mumbai';

    RETURN trip_count;
END;
$$ LANGUAGE PLPGSQL;
```
```

You can use these functions in your PostgreSQL environment. Example usage:

a) `SELECT \* FROM MaxExpenseBusinessTrip();`

b) `SELECT CountTripsFromPuneToMumbai();`

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.

## 2) Railway Reservation Database

Consider a Railway reservation system for passengers. The bogie capacity of all the bogies of a train

is same.

TRAIN (train\_no int, train\_name varchar(20), depart\_time time , arrival\_time time, source\_stn varchar (20),dest\_stn varchar (20), no\_of\_res\_bogies int ,bogie\_capacity int)

PASSENGER (passenger\_id int, passenger\_name varchar(20), address varchar(30), age int ,gender char)

Relationships:

Train \_Passenger: M-M relationship named ticket with descriptive attributes as follows:

TICKET( train\_no int, passenger\_id int, ticket\_no int ,bogie\_no int, no\_of\_berths int ,tdate date , ticket\_amt decimal(7,2),status char)

Constraints:

The status of a berth can be 'W' (waiting) or 'C' (confirmed).

Execute the following stored functions.

1. Write a stored function to calculate the ticket amount paid by all the passengers on 12/12/2019 for all the trains.
2. Write a stored function to update the status of the ticket from 'waiting' to 'confirm' for passenger named "Mr.Mohite".

Ans

Certainly! Below are the stored functions for the given requirements:

1. **\*\*Calculate the total ticket amount paid by all passengers on 12/12/2019 for all trains.\*\***

```
```sql
CREATE OR REPLACE FUNCTION TotalTicketAmountOnDate(date_param DATE)
RETURNS DECIMAL(7,2) AS $$
DECLARE
    total_amount DECIMAL(7,2);
BEGIN
    SELECT COALESCE(SUM(ticket_amt), 0)
    INTO total_amount
    FROM TICKET
    WHERE tdate = date_param;

    RETURN total_amount;
END;
$$ LANGUAGE PLPGSQL;
```
```

2. **\*\*Update the status of the ticket from 'waiting' to 'confirm' for a passenger named "Mr.Mohite".\*\***

```
```sql
CREATE OR REPLACE FUNCTION
UpdateTicketStatusForPassenger(passenger_name_param VARCHAR)
RETURNS VOID AS $$
BEGIN
    UPDATE TICKET
    SET status = 'C'

```

```
WHERE passenger_id IN (SELECT passenger_id FROM PASSENGER WHERE
passenger_name = passenger_name_param)
AND status = 'W';
END;
$$ LANGUAGE PLPGSQL;
```
```

You can use these functions in your PostgreSQL environment. Example usage:

1. `SELECT TotalTicketAmountOnDate('2019-12-12');`
2. `CALL UpdateTicketStatusForPassenger('Mr.Mohite');`

Ensure that your database schema aligns with the provided entities and relationships for accurate results. If you encounter any issues, double-check your database setup.