**22AIE303 – DBMS**

**CLASS WORKSHEET – 09-dec-2024**

Name : Anuvind M P

Roll no : AM.EN.U4AIE22010

1. Write a function that accepts two ints and returns its sum

**CODE :**

CREATE function intsum(a int, b int) returns int AS

$$

Declare

c int;

begin

c:=a+b;

return c;

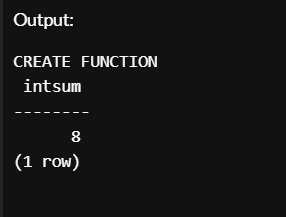
end;

$$

language 'plpgsql';

select intsum(5,3);

**OUTPUT :**

****

1. Create a function that accepts eno and returns salary

**CODE :**

CREATE table emp(eno int primary key, ename varchar(10), salary int);

INSERT INTO emp (eno, ename, salary) VALUES

(1, 'Alice', 50000),

(2, 'Bob', 60000),

(3, 'Charlie', 55000),

(4, 'Diana', 70000),

(5, 'Eve', 45000);

CREATE function salaryfetch(enum emp.eno%type, OUT sal emp.salary%type) AS

$$

begin

select salary into sal from emp where enum = eno;

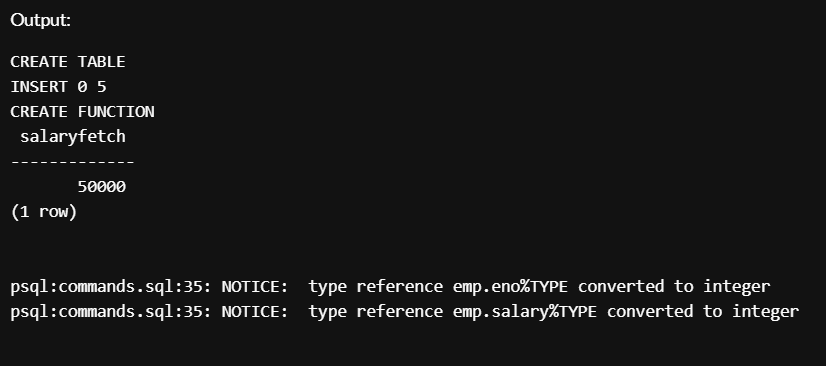
end;

$$

language 'plpgsql';

select salaryfetch(1);

**OUTPUT :**

****

1. Write a function which accepts salesperson number and check if the salesperson is eligible for getting commission. A salesperson is eligible for commission only if he has sold a total of 1000 quantity or more for all products together. If salesperson is eligible update the commission amount for the corresponding salesperson in the salesperson table. Initially commission is null

**CODE :**

CREATE TABLE SalesPerson (

slno INT PRIMARY KEY,

slname VARCHAR(10),

commission CHAR(1)

);

CREATE TABLE Product (

pno INT PRIMARY KEY,

pname VARCHAR(10),

unitprice INT

);

CREATE TABLE Sales (

sno INT PRIMARY KEY,

slno INT REFERENCES SalesPerson(slno),

pno INT REFERENCES Product(pno),

qtysold int

);

INSERT INTO SalesPerson (slno, slname, commission) VALUES

(1, 'Alice', null),

(2, 'Bob', null),

(3, 'Charlie', null);

INSERT INTO Product (pno, pname, unitprice) VALUES

(101, 'Laptop', 800),

(102, 'Tablet', 500),

(103, 'Phone', 300);

INSERT INTO Sales (sno, slno, pno, qtysold) VALUES

(1, 1, 101, 6501),

(2, 2, 103, 540),

(3, 3, 102, 1200);

CREATE function commission\_eligible(sales\_no SalesPerson.slno%type, OUT msg varchar) AS

$$

Declare

tot\_qty Sales.qtysold%type;

begin

select sum(qtysold) into tot\_qty from Sales where slno = sales\_no;

if tot\_qty > 1000 then

update SalesPerson set commission = 'y' where slno = sales\_no;

msg := 'Eligible';

else

update SalesPerson set commission = 'n' where slno = sales\_no;

msg := 'Not Eligible';

end if;

end;

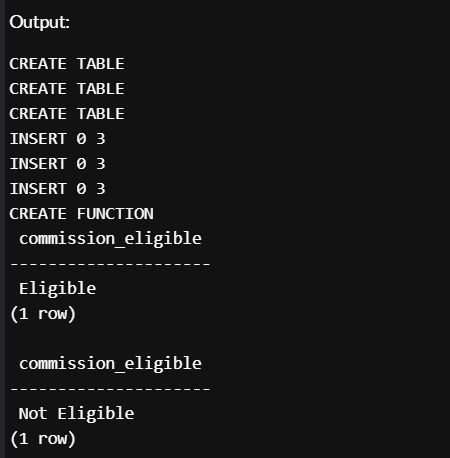
$$

language 'plpgsql';

select commission\_eligible(1);

select commission\_eligible(2);

**OUTPUT :**



1. For all courses maximum sixty students can be registered write a function to register a student for a particular course only if current Number of students registered for that course is not exceeding the limit

CODE :

-- Create the student table

CREATE TABLE Student (

sno INT PRIMARY KEY,

sname VARCHAR(15)

);

-- Create the course table

CREATE TABLE Course (

cno INT PRIMARY KEY,

cname VARCHAR(15)

);

-- Create the stud\_course table to represent the many-to-many relationship

CREATE TABLE Stud\_Course (

sno INT REFERENCES Student(sno),

cno INT REFERENCES Course(cno),

PRIMARY KEY (sno, cno)

);

-- Insert sample students

INSERT INTO Student (sno, sname) VALUES

(1, 'Alice'),

(2, 'Bob'),

(3, 'Charlie'),

(4, 'Diana'),

(5, 'Eve');

-- Insert sample courses

INSERT INTO Course (cno, cname) VALUES

(101, 'Mathematics'),

(102, 'Physics'),

(103, 'Chemistry');

CREATE OR REPLACE FUNCTION register\_student(s\_id INT, c\_id INT)

RETURNS TEXT AS $$

DECLARE

current\_count INT;

BEGIN

SELECT COUNT(\*) INTO current\_count

FROM Stud\_Course

WHERE cno = c\_id;

-- Check if the limit is exceeded

IF current\_count >= 60 THEN

RETURN 'Registration failed: Maximum student limit (60) reached for this course.';

ELSE

-- Insert the student into the course

INSERT INTO Stud\_Course (sno, cno) VALUES (s\_id, c\_id);

RETURN 'Registration successful!';

END IF;

END;

$$ LANGUAGE plpgsql;

SELECT register\_student(1, 101);

SELECT register\_student(2, 101);

SELECT \* FROM Stud\_Course;

OUTPUT :

