Project Title

Students Attendance and Marks Calculator

Name: Ansh Patel Pragneshkumar

Enrolment No: 22002170110104

Roll Number: 12

Batch: D1

Branch: CE

Creating Tables:

1. Marks -

CREATE TABLE marks (

s\_id INT PRIMARY KEY,

s\_name VARCHAR(15),

s\_branch VARCHAR(10),

total INT,

maths\_marks NUMERIC(5,2),

dbms\_marks NUMERIC(5,2),

java\_marks NUMERIC(5,2),

marks\_without\_bonus NUMERIC(5,2),

marks\_with\_bonus NUMERIC(5,2),

percentage NUMERIC(5,2)

);

1. Attendance -

CREATE TABLE attendance (

s\_id INT PRIMARY KEY,

s\_name VARCHAR(15),

maths\_attended\_lec INT,

maths\_total\_lec INT,

java\_attended\_lec INT,

java\_total\_lec INT,

dbms\_attended\_lec INT,

dbms\_total\_lec INT,

attended\_lec INT,

total\_lec INT,

attendance\_percentage NUMERIC(5,2),

bonus INT

);

Creating Procedures to insert data:

1. Insert in Marks Table:

CREATE OR REPLACE PROCEDURE insert\_in\_marks(s\_id INT, s\_name VARCHAR(15), s\_branch VARCHAR(10) ,total INT, maths\_marks NUMERIC, dbms\_marks NUMERIC, java\_marks NUMERIC) AS $$

BEGIN

INSERT INTO marks(s\_id, s\_name, s\_branch, total, maths\_marks, dbms\_marks,java\_marks)

VALUES (s\_id, s\_name, s\_branch ,total ,maths\_marks ,dbms\_marks , java\_marks);

END;

$$

LANGUAGE plpgsql;

1. Insert in Attendance Table:

CREATE OR REPLACE PROCEDURE insert\_in\_attendance(s\_id INT, s\_name VARCHAR(15),maths\_attended\_lec INT, maths\_total\_lec INT, java\_attended\_lec INT, java\_total\_lec INT, dbms\_attended\_lec INT, dbms\_total\_lec INT) AS $$

BEGIN

INSERT INTO attendance(s\_id, s\_name, maths\_attended\_lec, maths\_total\_lec, java\_attended\_lec, java\_total\_lec, dbms\_attended\_lec, dbms\_total\_lec)

VALUES (s\_id, s\_name, maths\_attended\_lec, maths\_total\_lec, java\_attended\_lec, java\_total\_lec, dbms\_attended\_lec, dbms\_total\_lec);

END;

$$

LANGUAGE plpgsql;

Creating Triggers:

CREATE OR REPLACE FUNCTION calculate\_attendance() RETURNS TRIGGER AS $$

DECLARE

total\_lec\_attended INT = 0;

total NUMERIC(5,2) = 0;

att\_per NUMERIC(5,2) = 0;

BEGIN

total\_lec\_attended = NEW.maths\_attended\_lec + NEW.java\_attended\_lec + NEW.dbms\_attended\_lec;

UPDATE attendance SET attended\_lec = total\_lec\_attended WHERE s\_id = NEW.s\_id;

total = NEW.maths\_total\_lec + NEW.java\_total\_lec + NEW.dbms\_total\_lec;

UPDATE attendance SET total\_lec=total WHERE s\_id = NEW.s\_id;

att\_per = (total\_lec\_attended / total) \* 100 ;

UPDATE attendance SET attendance\_percentage = att\_per WHERE s\_id = NEW.s\_id;

CALL calculate\_bonus(NEW.s\_id,att\_per);

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER insert\_attendance AFTER INSERT ON attendance FOR EACH ROW EXECUTE FUNCTION calculate\_attendance();

CREATE OR REPLACE PROCEDURE calculate\_bonus(id IN INT, att\_per IN NUMERIC) AS $$

BEGIN

IF att\_per >= 90 then

UPDATE attendance SET bonus = 5 WHERE s\_id = id;

ELSIF att\_per >= 80 then

UPDATE attendance SET bonus = 4 WHERE s\_id = id;

ELSIF att\_per >= 70 then

UPDATE attendance SET bonus = 3 WHERE s\_id = id;

ELSIF att\_per >= 60 then

UPDATE attendance SET bonus = 2 WHERE s\_id = id;

ELSIF att\_per >= 50 then

UPDATE attendance SET bonus = 1 WHERE s\_id = id;

ELSE

UPDATE attendance SET bonus = 0 WHERE s\_id = id;

END IF;

END;

$$

LANGUAGE plpgsql;

2.

CREATE OR REPLACE FUNCTION calculate\_marks() RETURNS TRIGGER AS $$

DECLARE

total\_marks NUMERIC(5,2)=0;

percentage NUMERIC(5,2) =0;

BEGIN

total\_marks = NEW.maths\_marks + NEW.java\_marks + NEW.dbms\_marks;

UPDATE marks SET marks\_without\_bonus = total\_marks WHERE s\_id = NEW.s\_id;

UPDATE marks SET percentage = total\_marks / 3 WHERE s\_id = NEW.s\_id;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER insert\_marks AFTER INSERT ON marks FOR EACH ROW EXECUTE FUNCTION calculate\_marks();

Creating Procedures And Functions:

CREATE OR REPLACE PROCEDURE add\_bonus() AS $$

DECLARE

total\_without\_bonus marks.marks\_without\_bonus%TYPE;

bonus\_marks attendance.bonus%TYPE;

id marks.s\_id%TYPE;

c1 CURSOR FOR SELECT marks.s\_id, marks\_without\_bonus, bonus FROM marks, attendance WHERE marks.s\_id = attendance.s\_id;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO id,total\_without\_bonus,bonus\_marks;

EXIT WHEN NOT FOUND;

IF total\_without\_bonus + bonus\_marks > 300 THEN

UPDATE marks SET marks\_with\_bonus = 300 WHERE s\_id = id;

ELSE

UPDATE marks SET marks\_with\_bonus = total\_without\_bonus + bonus\_marks WHERE s\_id = id;

END IF;

UPDATE marks SET percentage = marks\_with\_bonus / 3 WHERE s\_id = id;

END LOOP;

CLOSE c1;

END;

$$

LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE display\_branch\_topper(branch VARCHAR) AS $$

DECLARE

var marks%ROWTYPE;

BEGIN

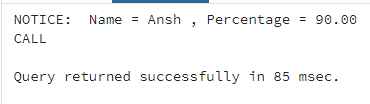
SELECT \* FROM marks WHERE s\_branch = branch ORDER BY percentage DESC INTO var;

RAISE NOTICE 'Name = % , Percentage = %',var.s\_name,var.percentage;

END;

$$

LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION getUserDetailsByid(id INT) RETURNS TABLE(s\_id INT, s\_name VARCHAR(15), s\_branch VARCHAR(10),

total INT,maths\_marks NUMERIC(5,2),dbms\_marks NUMERIC(5,2),java\_marks NUMERIC(5,2),marks\_without\_bonus NUMERIC(5,2),

marks\_with\_bonus NUMERIC(5,2),percentage NUMERIC(5,2))AS $$

BEGIN

RETURN QUERY

SELECT \* FROM marks WHERE s\_id = id;

END

$$

LANGUAGE plpgsql;

