Tugas 5 : Manajemen Basis Data

View, Log, Funtion, Trigger

DDL & DML

CREATE DATABASE CostumersActive

CREATE TABLE customers(

    ID INTEGER PRIMARY KEY NOT NULL,

    NAME VARCHAR(50) NOT NULL,

    AGE INTEGER NOT NULL,

    ADDRESS VARCHAR(50) NOT NULL,

    SALARY DECIMAL(6,2) NOT NULL

);

INSERT INTO customers VALUES

(1,'Ramesh',32, 'Ahmedabad',2000.00),

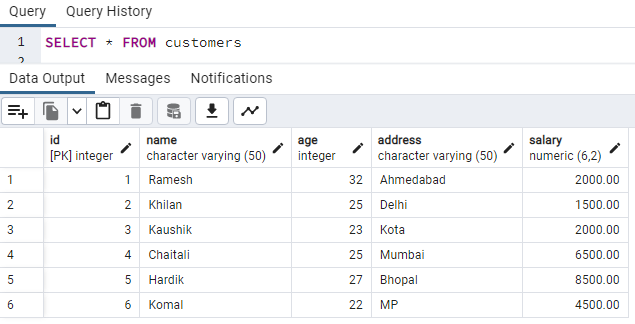
(2,'Khilan',25,'Delhi',1500.00),

(3,'Kaushik',23,'Kota',2000.00),

(4,'Chaitali',25,'Mumbai',6500.00),

(5,'Hardik',27,'Bhopal',8500.00),

(6,'Komal',22,'MP',4500.00)



1. Create **a view** to displays name, salary and age of each customer who has salary more than 2000.00

-- Create a view to displays name, salary and age of each customer who has salary more than 2000.00

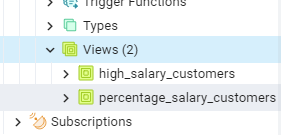
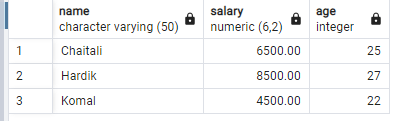
CREATE VIEW high\_salary\_customers AS

SELECT name, salary, age

FROM customers

WHERE salary > 2000.00;

SELECT \* FROM high\_salary\_customers



1. Create **a view** to displays the name and percentage of each customer's salary to the total salary and is sorted from the largest percentage to the smallest percentage

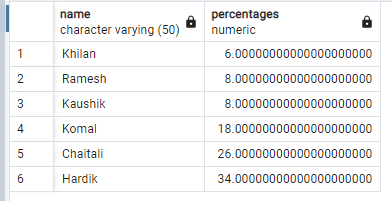
CREATE VIEW percentage\_salary\_customers AS

SELECT name, (salary / (SELECT SUM(salary) FROM customers)) \* 100 AS PERCENTAGES

FROM Customers

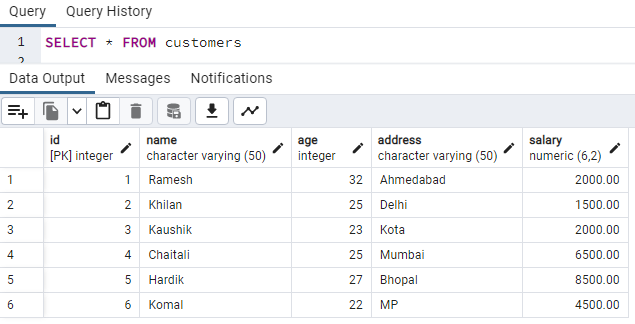
ORDER BY percentages

SELECT \* FROM percentage\_salary\_customers DESC

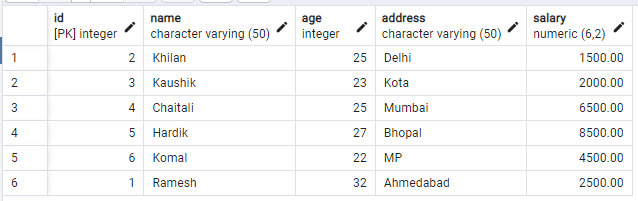


1. Create **a trigger** to record a log that shows salary changes if there is a change in the salary (previous salary is different from the next salary). Columns in the log : IDLog, date, previous salary, next salary.
2. CREATE TABLE customers(
3. ID INTEGER PRIMARY KEY NOT NULL,
4. NAME VARCHAR(50) NOT NULL,
5. AGE INTEGER NOT NULL,
6. ADDRESS VARCHAR(50) NOT NULL,
7. SALARY DECIMAL(6,2) NOT NULL
8. );
9. CREATE TABLE customers\_log(
10. IDlog INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
11. Date\_Change DATE NOT NULL,
12. Previous\_Salary DECIMAL(6,2) NOT NULL,
13. Next\_Salary DECIMAL(6,2) NOT NULL
14. );
15. CREATE OR REPLACE FUNCTION user\_change\_salary()
16. RETURNS TRIGGER
17. LANGUAGE PLPGSQL
18. AS $$
19. BEGIN
20. IF OLD.salary != NEW.salary THEN
21. INSERT INTO customers\_log (Date\_Change,Previous\_Salary,Next\_Salary)
22. VALUES (NOW(),OLD.salary,NEW.salary);
23. END IF;
24. RETURN NEW;
25. END $$
26. CREATE OR REPLACE TRIGGER change\_salary
27. AFTER UPDATE ON Customers
28. FOR EACH ROW
29. EXECUTE FUNCTION user\_change\_salary();
30. -- Update test
31. UPDATE Customers SET salary = 2500.00 WHERE ID = 1;
32. select \* from customers
33. select \* from customers\_log

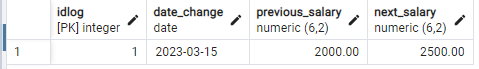
Before Update :



After Update :



Customer\_log :



4. Create **a trigger** to ensure that the salary range entered in the customer table is 1500 - 8500 (if there is an insert / update salary > 8500, then salary = 8500 and if there is an insert / update salary < 1500 then salary = 1500)

CREATE OR REPLACE FUNCTION salary\_range()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS $$

BEGIN

    IF NEW.salary > 8500.00 THEN

        NEW.salary = 8500.00;

    ELSIF NEW.salary < 1500.00 THEN

        NEW.salary = 1500.00;

    END IF;

    RETURN NEW;

END $$;

CREATE OR REPLACE TRIGGER salary\_range\_fixed

BEFORE UPDATE ON Customers

FOR EACH ROW

EXECUTE FUNCTION salary\_range();

DROP TRIGGER salary\_range\_fixed ON Customers;

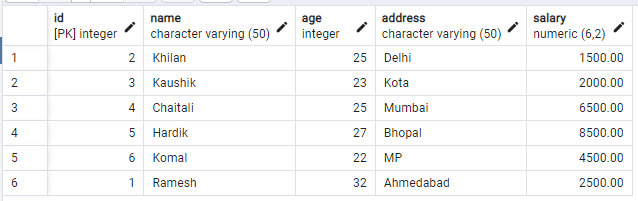
DROP FUNCTION salary\_range();

select \* from customers

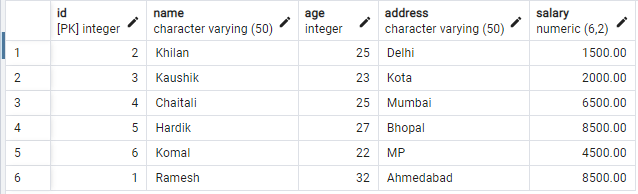
select \* from customers\_log

UPDATE Customers SET salary = 9200.00 WHERE ID = 1;

Before Update :



After Update :



Costumer log

