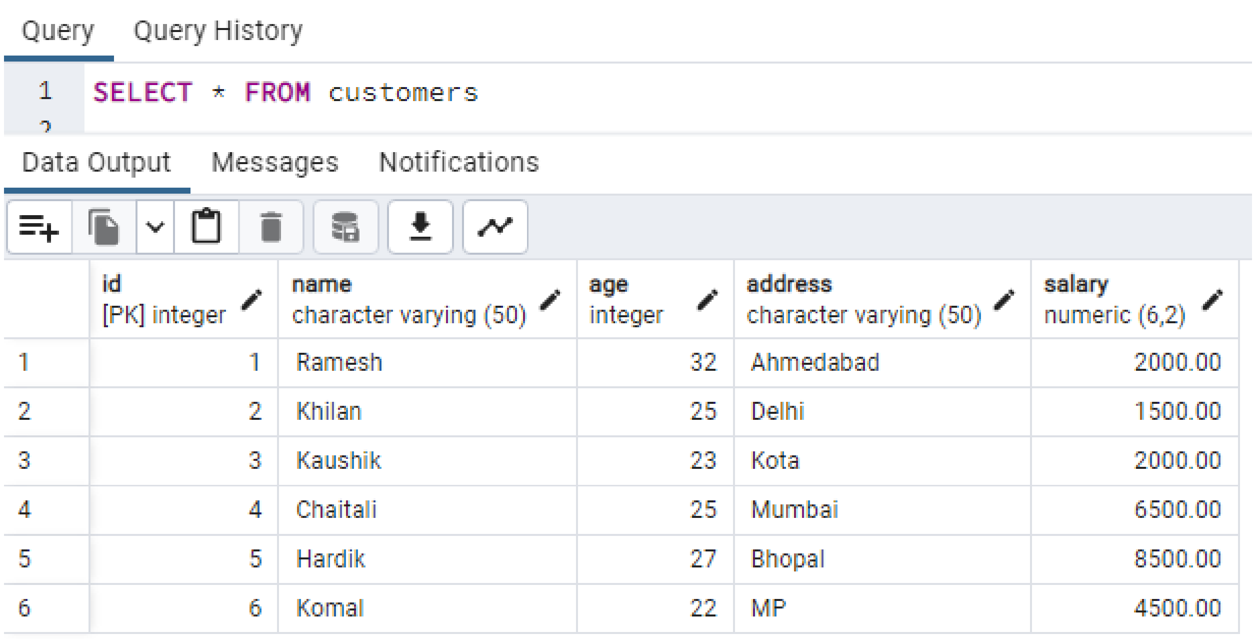
Tugas 5 : Manajemen Basis Data

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)
```

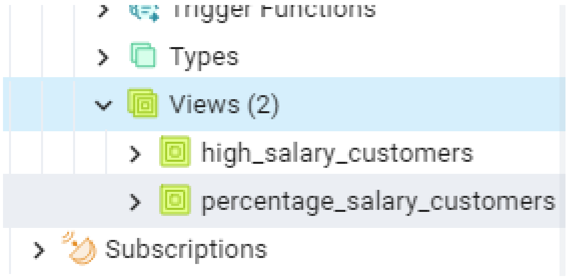


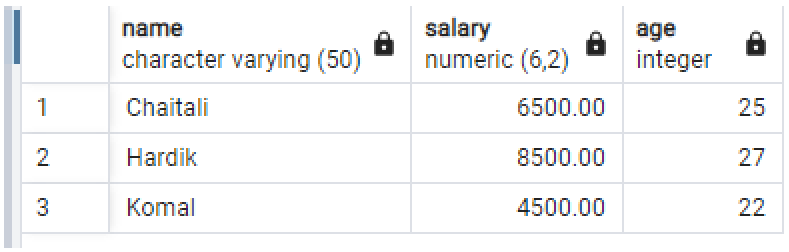
```
SELECT * FROM customers
```

	id [PK] integer	name character varying (50)	age integer	address character varying (50)	salary numeric (6,2)
1	1	Ramesh	32	Ahmedabad	2000.00
2	2	Khilan	25	Delhi	1500.00
3	3	Kaushik	23	Kota	2000.00
4	4	Chaitali	25	Mumbai	6500.00
5	5	Hardik	27	Bhopal	8500.00
6	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
```

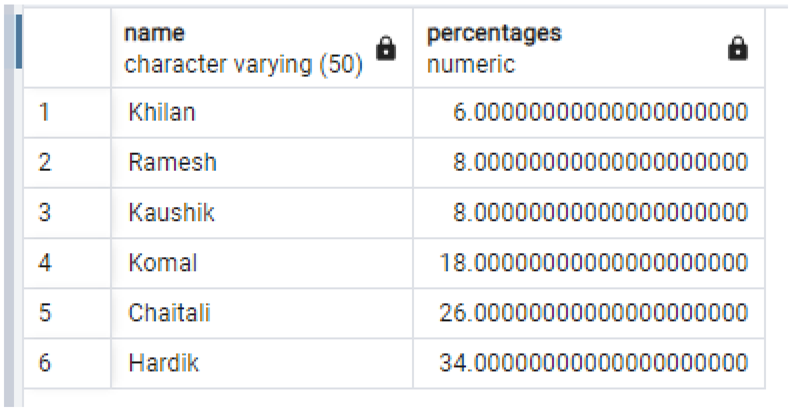




	name character varying (50)	salary numeric (6,2)	age integer
1	Chaitali	6500.00	25
2	Hardik	8500.00	27
3	Komal	4500.00	22

2. 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

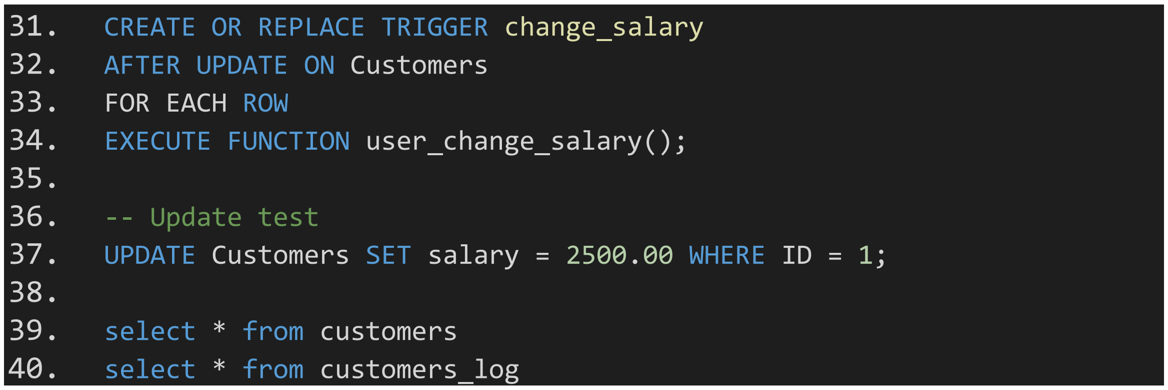
```
CREATE VIEW percentage_salary_customers AS
SELECT name, (salary / (SELECT SUM(salary) FROM customers)) * 100 AS PERCENTAGES
FROM Customers
ORDER BY percentages

SELECT * FROM percentage_salary_customers DESC
```



	name character varying (50)	percentages numeric
1	Khilan	6.00000000000000000000
2	Ramesh	8.00000000000000000000
3	Kaushik	8.00000000000000000000
4	Komal	18.00000000000000000000
5	Chaitali	26.00000000000000000000
6	Hardik	34.00000000000000000000

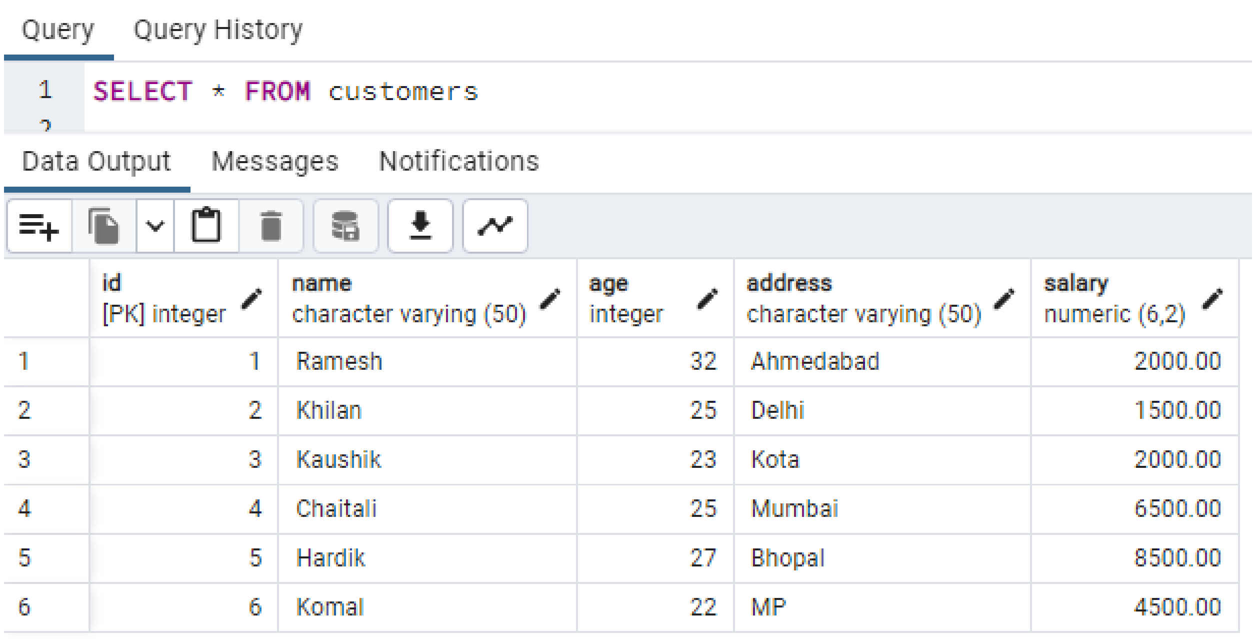
3. Create **a trigger** to record a log that shows salary changes if there is a change in the salary (previous salary is different from the next salary). Columns in the log : IDLog, date, previous salary, next salary.

```sql
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
);

CREATE TABLE customers_log(
    IDlog INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    Date_Change DATE NOT NULL,
    Previous_Salary DECIMAL(6,2) NOT NULL,
    Next_Salary DECIMAL(6,2) NOT NULL
);

CREATE OR REPLACE FUNCTION user_change_salary()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
BEGIN
    IF OLD.salary != NEW.salary THEN
    INSERT INTO customers_log (Date_Change,Previous_Salary,Next_Salary)
    VALUES (NOW(),OLD.salary,NEW.salary);
    END IF;
    RETURN NEW;
END $$

```



```sql
CREATE OR REPLACE TRIGGER change_salary
AFTER UPDATE ON Customers
FOR EACH ROW
EXECUTE FUNCTION user_change_salary();

-- Update test
UPDATE Customers SET salary = 2500.00 WHERE ID = 1;

select * from customers
select * from customers_log
```

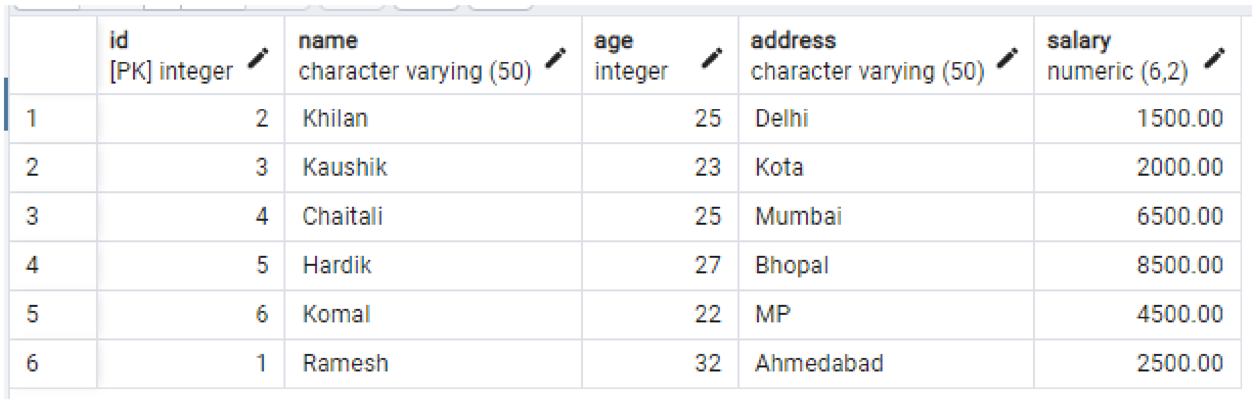
Before Update :



```sql
SELECT * FROM customers
```

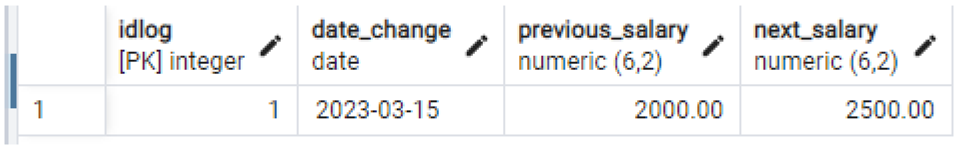
	id [PK] integer	name character varying (50)	age integer	address character varying (50)	salary numeric (6,2)
1	1	Ramesh	32	Ahmedabad	2000.00
2	2	Khilan	25	Delhi	1500.00
3	3	Kaushik	23	Kota	2000.00
4	4	Chaitali	25	Mumbai	6500.00
5	5	Hardik	27	Bhopal	8500.00
6	6	Komal	22	MP	4500.00

After Update :



	id [PK] integer	name character varying (50)	age integer	address character varying (50)	salary numeric (6,2)
1	2	Khilan	25	Delhi	1500.00
2	3	Kaushik	23	Kota	2000.00
3	4	Chaitali	25	Mumbai	6500.00
4	5	Hardik	27	Bhopal	8500.00
5	6	Komal	22	MP	4500.00
6	1	Ramesh	32	Ahmedabad	2500.00

Customer_log :

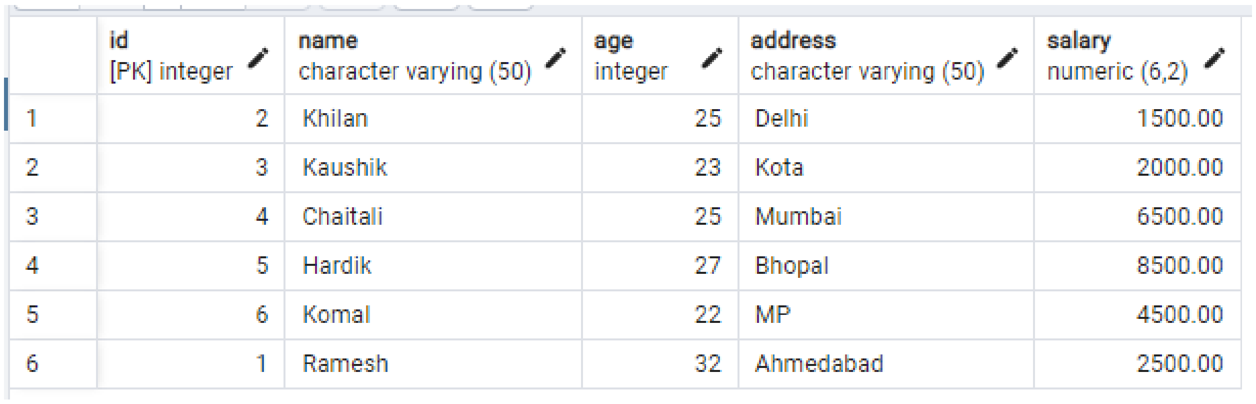


	idlog [PK] integer	date_change date	previous_salary numeric (6,2)	next_salary numeric (6,2)
1	1	2023-03-15	2000.00	2500.00

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)

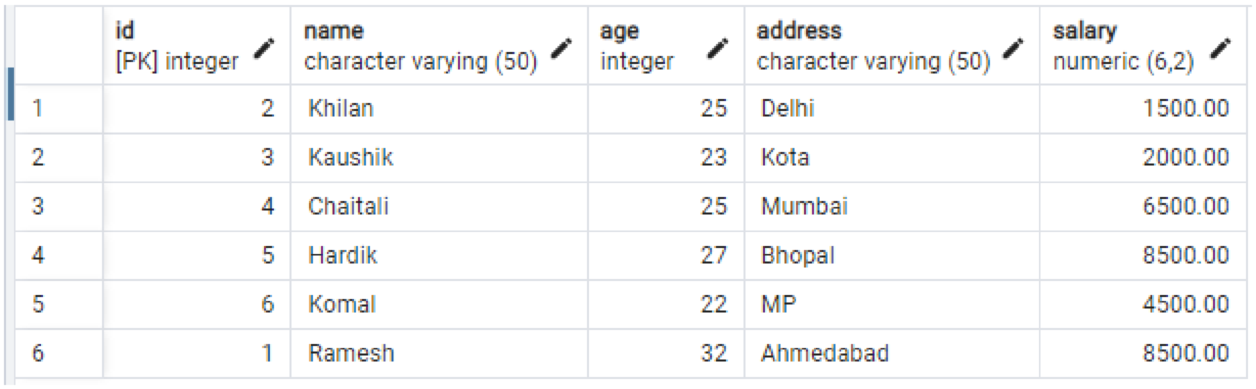
```
CREATE OR REPLACE FUNCTION salary_range()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
BEGIN
    IF NEW.salary > 8500.00 THEN
        NEW.salary = 8500.00;
    ELSIF NEW.salary < 1500.00 THEN
        NEW.salary = 1500.00;
    END IF;
    RETURN NEW;
END $$;


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 :



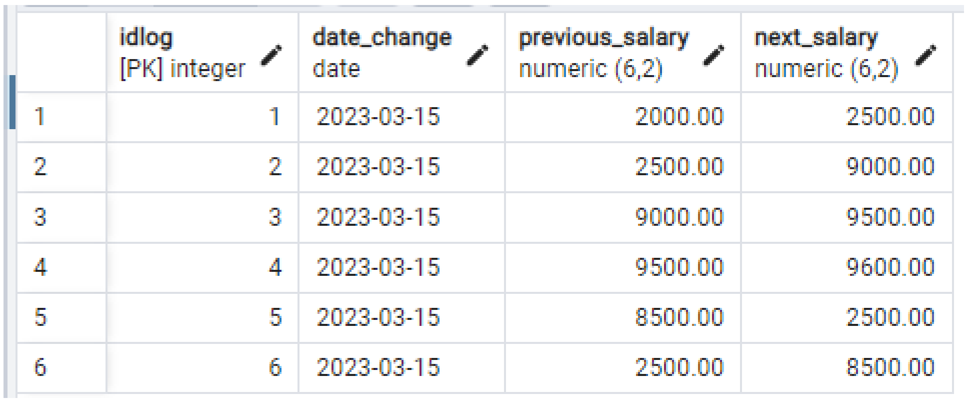
	id [PK] integer	name character varying (50)	age integer	address character varying (50)	salary numeric (6,2)
1	2	Khilan	25	Delhi	1500.00
2	3	Kaushik	23	Kota	2000.00
3	4	Chaitali	25	Mumbai	6500.00
4	5	Hardik	27	Bhopal	8500.00
5	6	Komal	22	MP	4500.00
6	1	Ramesh	32	Ahmedabad	2500.00

After Update :



	id [PK] integer	name character varying (50)	age integer	address character varying (50)	salary numeric (6,2)
1	2	Khilan	25	Delhi	1500.00
2	3	Kaushik	23	Kota	2000.00
3	4	Chaitali	25	Mumbai	6500.00
4	5	Hardik	27	Bhopal	8500.00
5	6	Komal	22	MP	4500.00
6	1	Ramesh	32	Ahmedabad	8500.00

Costumer log



	idlog [PK] integer	date_change date	previous_salary numeric (6,2)	next_salary numeric (6,2)
1	1	2023-03-15	2000.00	2500.00
2	2	2023-03-15	2500.00	9000.00
3	3	2023-03-15	9000.00	9500.00
4	4	2023-03-15	9500.00	9600.00
5	5	2023-03-15	8500.00	2500.00
6	6	2023-03-15	2500.00	8500.00