**Views**

-- Views are pseudo-tables.

-- That is, they are not real tables; nevertheless, appear as ordinary tables to SELECT.

-- A view can represent a subset of a real table, selecting certain columns or certain rows from an ordinary table.

-- A view can even represent joined tables.

-- Because views are assigned separate permissions, you can use them to restrict table access so that the users see only specific rows or columns of a table.

-- A view can contain all rows of a table or selected rows from one or more tables.

-- A view can be created from one or many tables, which depends on the written PostgreSQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following −

* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data such that a user can only see limited data instead of complete table.
* Summarize data from various tables, which can be used to generate reports.

-- Since views are not ordinary tables, you may not be able to execute a DELETE, INSERT, or UPDATE statement on a view.

-- However, you can create a RULE to correct this problem of using DELETE, INSERT or UPDATE on a view.

Creating Views

-- The PostgreSQL views are created using the **CREATE VIEW** statement.

-- PostgreSQL views can be created from a single table, multiple tables, or another view.

-- The basic CREATE VIEW syntax is as follows −

CREATE [TEMP | TEMPORARY] VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE [condition];

-- You can include multiple tables in your SELECT statement in very similar way as you use them in normal PostgreSQL SELECT query.

-- If the optional TEMP or TEMPORARY keyword is present, the view will be created in the temporary space.

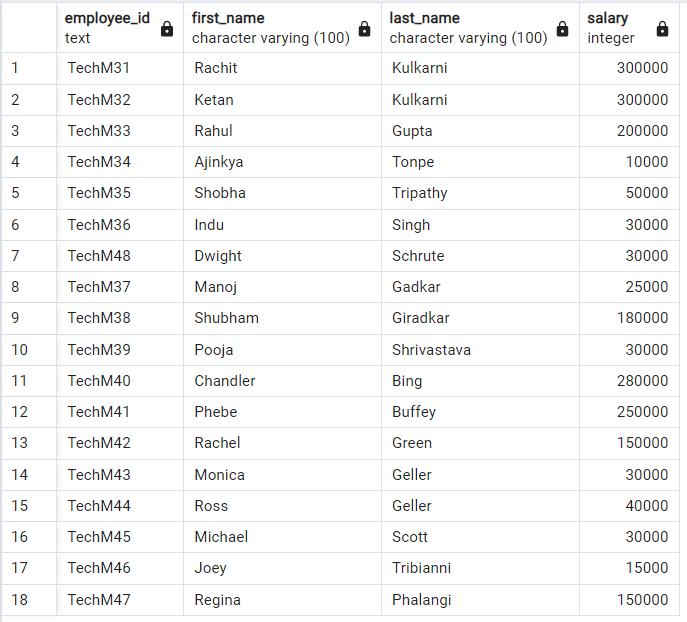
-- Temporary views are automatically dropped at the end of the current session.

**-- View for employees and their salaries**

CREATE VIEW dev\_schema.emp\_salary\_view AS

SELECT employee\_id, first\_name, last\_name, salary FROM dev\_schema.employee;

SELECT \* FROM dev\_schema.emp\_salary\_view;



SELECT \* FROM dev\_schema.emp\_salary\_view ORDER BY salary DESC;

SELECT employee\_id, salary,

DENSE\_RANK() OVER (ORDER BY salary DESC)

FROM dev\_schema.emp\_salary\_view;

**-- View for employee and their department**

CREATE VIEW dev\_schema.emp\_dept\_view AS

SELECT \* FROM

dev\_schema.employee AS emp INNER JOIN dev\_schema.department as dept

ON emp.fk\_department\_id = dept.department\_id;

SELECT employee\_id, first\_name, last\_name, salary, department\_name FROM dev\_schema.emp\_dept\_view;

