

# View

## What you will learn

---



Describe views



Create and use  
a view



Explain how  
materialized  
views differ  
from regular  
views



Create and use  
a materialized  
view

create and use a

## What is a view?

---

- Presents data from one or more tables or additional views
  - Enables interaction with data as with tables
  - Restricts access to sensitive data
  - Simplifies data retrieval
  - Reduces access to underlying tables
- 
- A view is an alternative method of presenting information sourced from one or more tables or additional views. You can interact with views in the same way as you interact with tables. Inserting, updating, and deleting data as required. Views are a useful way of limiting access to sensitive data, simplifying data retrieval, and reducing access to underlying tables.

# What is a view?

empid	firstname	lastname	startdate	salary
1001	John	Thomas	2002-07-31	100000
1002	Alice	James	2010-08-10	80000
1003	Steve	Wells	2015-07-20	50000
1004	Santosh	Kumar	2011-01-04	60000
1005	Ahmed	Hussain	2008-02-06	70000
1006	Nancy	Allen	2005-05-05	90000
1007	Mary	Thomas	2005-05-06	65000
1008	Bharath	Gupta	2020-07-09	65000
1009	Andrea	Jones	2002-03-30	70000
1010	Ann	Jacobs	2006-09-01	70000

empid	email
1001	johnt@
1002	alicej@
1003	steve@
1004	santoshk@
1005	ahmedh@
1006	nancya@
1007	maryt@
1008	bharathg@
1009	andrea@
1010	annj@

For example, you could

# What is a view?

empid	firstname	lastname	startdate	salary
1001	John	Thomas	2002-07-31	100000
1002	Alice	James	2010-08-10	80000
1003	Steve	Wells	2015-07-20	50000
1004	Santosh	Kumar	2011-01-04	60000
1005	Ahmed	Hussain	2008-02-06	70000
1006	Nancy	Allen	2005-05-05	90000
1007	Mary	Thomas	2005-05-06	65000
1008	Bharath	Gupta	2020-07-09	65000
1009	Andrea	Jones	2002-03-30	70000
1010	Ann	Jacobs	2006-09-01	70000

empid	email
1001	johnt@
1002	alicej@
1003	steve@
1004	santoshk@
1005	ahmedh@
1006	nancya@
1007	maryt@
1008	bharathg@
1009	andrea@
1010	annj@

the name and email columns

- For example, you could create a view to include the name and email columns from these two tables

## What is a view?

firstname	lastname	email
John	Thomas	johnt@
Alice	James	alicej@
Steve	Wells	stevew@
Santosh	Kumar	santoshk@
Ahmed	Hussain	ahmedh@
Nancy	Allen	nancya@
Mary	Thomas	maryt@
Bharath	Gupta	bharathg@
Andrea	Jones	andreaaj@

stored in two different tables and without being

kills Network

- Users can then easily access this data without needing to know that it is stored in two different tables and without being given access to the sensitive salary information

## Creating a view

```
Create - View
General Definition Code Security SQL
1: SELECT firstname, lastname, email
2: FROM employee_details
3: INNER JOIN employee_contact_info
4: ON employee_details.empid = employee_contact_info.empid
```

## Using a view

```
Query Editor Query History
1 SELECT * FROM public.emp_email_view;
```

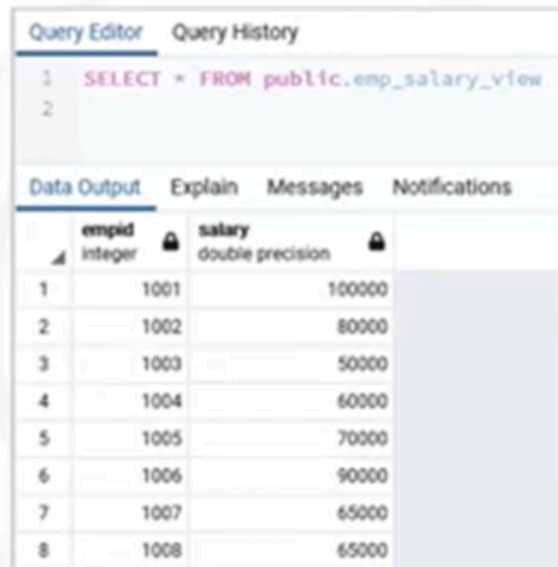
- Example View with PostgreSQL

## Materialized views

---

- Behave differently to regular views
  - Result set is materialized, or saved, for future use
  - Cannot insert, update, or delete rows
  - Can improve performance
- 
- PostgreSQL also supports another type of view, a materialized view. When you refresh a materialized view for the first time, the result set is materialized or saved for future use. This materialization does mean that you can only query the data and cannot update or delete it. However, it also enhances performance for future querying of the view because the result set is readily available, often stored in memory.

# Using a materialized view

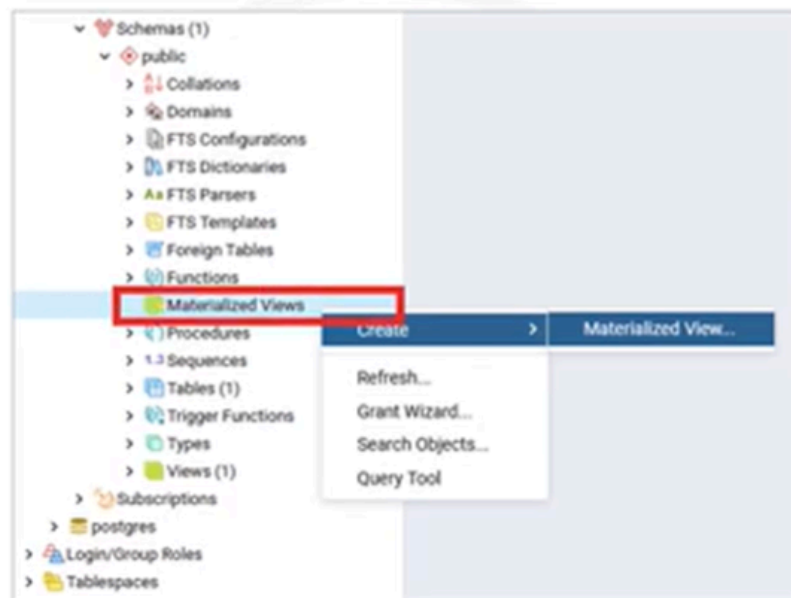


The screenshot shows a database interface with a 'Query Editor' tab. The query entered is `SELECT * FROM public.emp_salary_view`. Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'empid' (integer) and 'salary' (double precision). The table contains 8 rows of data.

	empid integer	salary double precision
1	1001	100000
2	1002	80000
3	1003	50000
4	1004	60000
5	1005	70000
6	1006	90000
7	1007	65000
8	1008	65000

to update it with the data in the underlying tables.

## Creating a materialized view



but you must ensure you start in

# Creating a materialized view



This example includes just the employee ID

- Example Materialized View with PostgreSQL

## Recap

In this video, you learned that:

- Views are another way of data presentation from one or more tables
- You can use views to restrict access to sensitive data and for easier data retrieval
- Materialized views store the result set for fast access
- You cannot insert, update, or delete rows using a materialized view

# Understanding Views in Databases: A Simple Explanation

A view is like a window into your data. Imagine you have a big library filled with books (which represent your data tables). Instead of letting everyone wander through the entire library, you create a special reading room (the view) where only certain books are available. This way, people can read what they need without getting lost in the whole library. Views help simplify data access and can also protect sensitive information by only showing what's necessary.

## Example to Illustrate

For instance, if you have a table with employee details that includes names, emails, and salaries, you can create a view that only shows names and emails. This way, users can access the information they need without seeing the sensitive salary data. It's like giving someone a menu at a restaurant that only lists the dishes they can order, rather than showing them the entire kitchen!

## What is a view in a database?

A **view** in a database is a virtual table that presents data from one or more tables in a specific way. Here are some key points about views:

- **Virtual Table:** A view does not store data itself; it displays data stored in other tables. It acts like a window to the data.
- **Simplifies Data Access:** Views can simplify complex queries by allowing users to access data without needing to understand the underlying table structures.
- **Data Security:** Views can restrict access to sensitive data. For example, you can create a view that only shows certain columns (like names and emails) while hiding others (like salaries).

- **Interactivity:** You can interact with views similarly to how you interact with tables, allowing you to select, insert, update, or delete data, depending on the view's definition.

In summary, views are useful for presenting data in a more manageable and secure way.

## What is the purpose of using views in databases?

The purpose of using views in databases includes several important benefits:

- **Simplified Data Access:** Views can present complex data in a more understandable format, making it easier for users to retrieve the information they need without dealing with complex queries.
- **Data Security:** By restricting access to certain columns or rows, views can protect sensitive information. For example, a view can show employee names and emails while hiding their salaries.
- **Data Abstraction:** Views provide a way to abstract the underlying table structure. Users can interact with the view without needing to know how the data is organized in the database.
- **Consistency:** Views can ensure that users see a consistent representation of data, even if the underlying tables change. This is useful for reporting and analysis.
- **Reusability:** Once a view is created, it can be reused in multiple queries, reducing redundancy and improving efficiency.

In summary, views enhance data management by simplifying access, improving security, and providing a consistent way to interact with data.



# What are the advantages of using materialized views over regular views?

Materialized views offer several advantages over regular views:

- **Performance Improvement:** Materialized views store the result set of a query physically, which means that accessing the data is faster compared to regular views that compute the data on-the-fly each time they are queried.
- **Reduced Load on Database:** Since the data is pre-computed and stored, materialized views can reduce the load on the database, especially for complex queries that require significant processing time.
- **Data Snapshot:** Materialized views provide a snapshot of the data at a specific point in time. This can be useful for reporting and analysis, as users can work with a stable dataset without being affected by ongoing changes in the underlying tables.
- **Query Optimization:** Materialized views can be indexed, which further enhances query performance. This allows for quicker access to the data when running queries against the materialized view.
- **Refresh Options:** Materialized views can be refreshed at specific intervals or on-demand, allowing users to control how up-to-date the data is while still benefiting from improved performance.

In summary, materialized views are beneficial for performance and efficiency, especially when dealing with large datasets or complex queries.

## What is a materialized view?

A **materialized view** is a database object that stores the result of a query physically, unlike a regular view, which generates the data dynamically each time it is accessed. Here are some key points about materialized views:

- **Stored Data:** Materialized views save the result set of a query, allowing for faster access to the data since it doesn't need to be recalculated every time.
- **Performance Benefits:** Because the data is pre-computed and stored, querying a materialized view is generally much faster, especially for complex queries or large datasets.
- **Snapshot of Data:** Materialized views provide a snapshot of the data at the time they were last refreshed. This means users can work with a stable dataset without being affected by changes in the underlying tables.
- **Refresh Mechanism:** Materialized views can be refreshed periodically or on-demand to update the stored data with the latest information from the underlying tables.
- **Use Cases:** They are particularly useful in reporting and data warehousing scenarios where performance and quick access to aggregated data are critical.

In summary, materialized views enhance performance and efficiency by storing query results, making data retrieval faster and more manageable.

## In what scenarios would you prefer using a materialized view over a regular view?

You might prefer using a **materialized view** over a regular view in the following scenarios:

- **Performance-Critical Applications:** When you have complex queries that take a long time to execute, materialized views can significantly speed up data retrieval since they store the results of the query.
- **Large Datasets:** In cases where you are working with large datasets, materialized views can help reduce the load on the database by pre-computing and storing the results, making it easier to access the data quickly.

- **Reporting and Analytics:** For reporting purposes, where users need to access aggregated or summarized data frequently, materialized views provide a stable snapshot of the data, allowing for faster report generation.
- **Data Warehousing:** In data warehousing scenarios, where data is often read but not frequently updated, materialized views can optimize query performance and improve user experience.
- **Static Data Requirements:** When you need to work with a consistent dataset that does not change frequently, materialized views can provide a reliable snapshot without being affected by ongoing changes in the underlying tables.

In summary, materialized views are ideal for scenarios where performance, quick access to data, and stable datasets are essential.