

Viewlar bilan ishlash

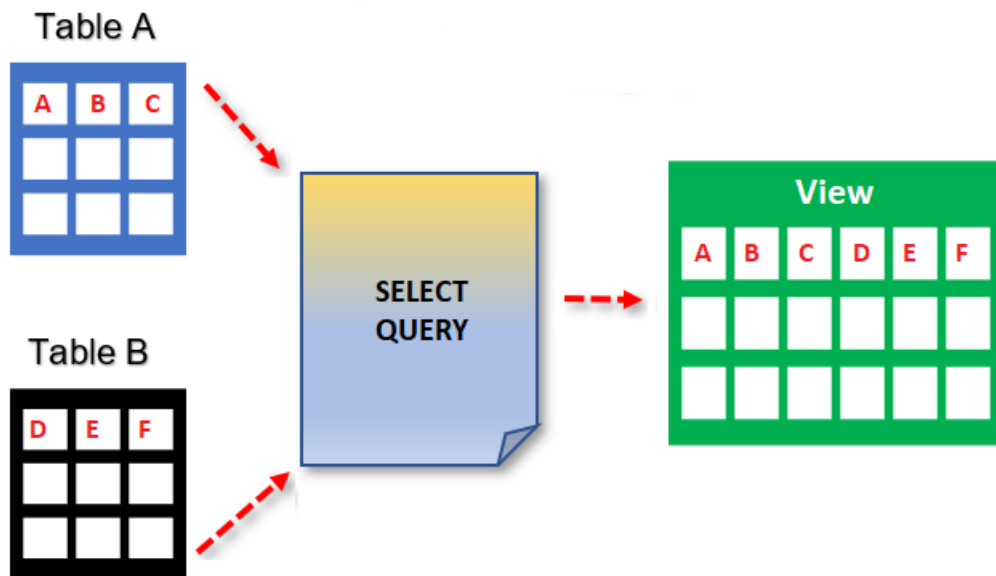
Reja:

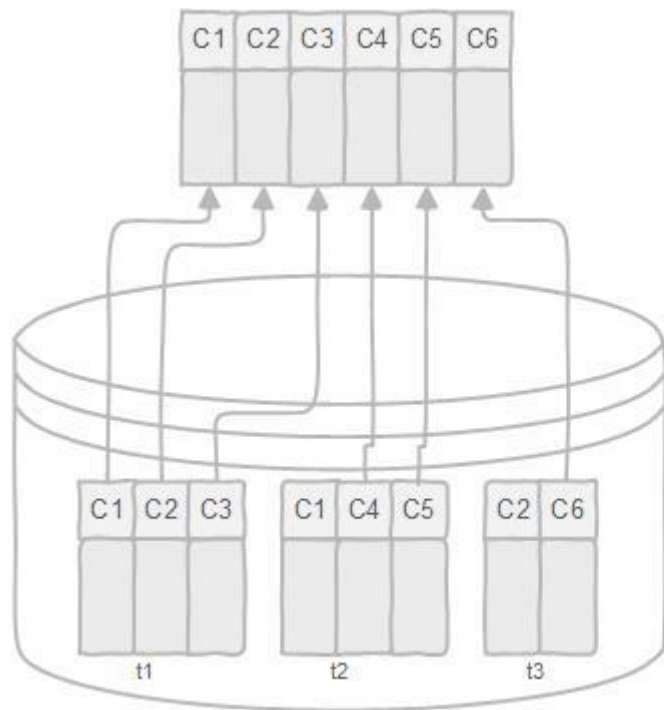
1. Views(Simple, Complex, Inline, Materialized)
2. Managing PostgreSQL Views
(CREATE, DROP, UPDATABLE)

View

View- bu ma`lumotlar ombori jadvalidagi ma`lumotlarni boshqacha tarzda ko`rsatuvchi so`rovdir. View bir yoki bir nechta jadvallarga asoslangan holda yaratiladi. Viewlar odatda murakkab so`rovlarni yaxlit bitta obe`kt sifatida o`rash uchun qo`llaniladi.

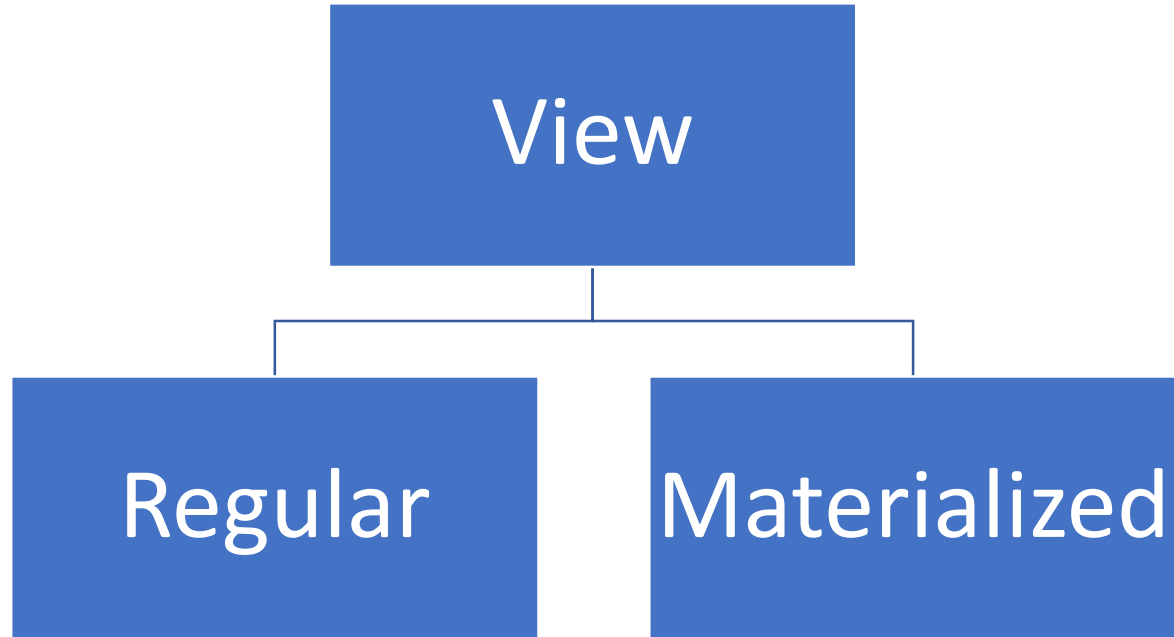
- PostgreSQLda **view** psevdo-jadvaldir, ya`ni u haqiqiy jadval emas.
- **View** bir yoki bir nechta jadvallardan yaratilishi mumkin.
- **View** yaratilgan jadvallar asosiy jadvallar deb nomlanadi.
- **View** bir nechta asosiy jadvallarning ayrim ustunlarini o`zida birlashtirgan jadvaldir.





```
SELECT * FROM view
```

```
SELECT c1,c2,c3,c4,c5,c6  
FROM t1  
JOIN t2 USING (c1)  
JOIN t3 USING (c2)
```



PostgreSqllda Simple View

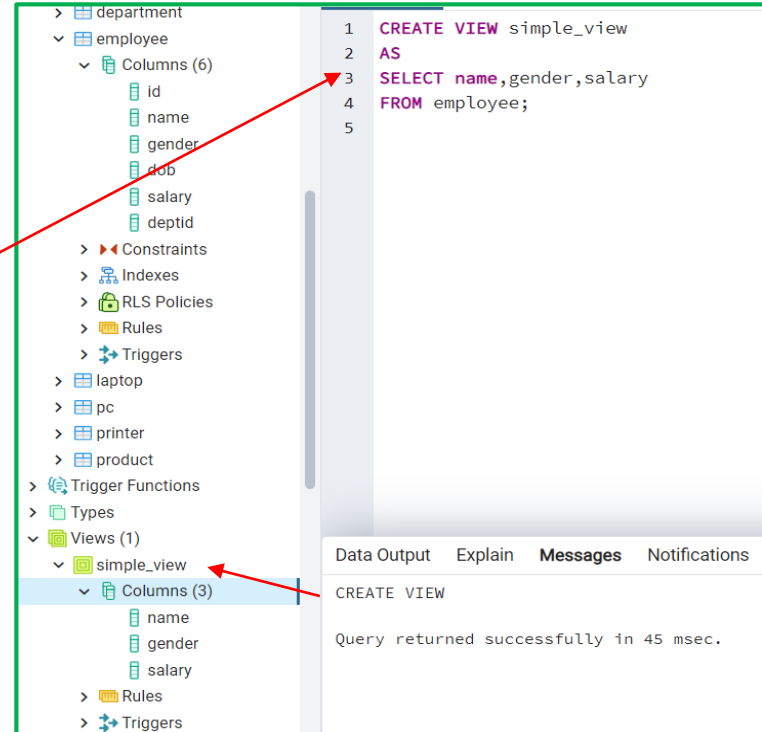
View bir bitta jadval asosida yaratilganda, u PostgreSQLda **simple (sodda) view** hisoblanadi. PostgreSQLdagi simple viewlarni tushunish uchun biz quyidagi **Employee** jadvalidan foydalanamiz.

Employee

ID	Name	Gender	DOB	Salary	DeptID
1	Pranaya	Male	1996-02-29 10:53:27.060	25000.00	1
2	Priyanka	Female	1995-05-25 10:53:27.060	30000.00	2
3	Anurag	Male	1995-04-19 10:53:27.060	40000.00	2
4	Preety	Female	1996-03-17 10:53:27.060	35000.00	3
5	Sambit	Male	1997-01-15 10:53:27.060	27000.00	1
6	Hina	Female	1995-07-12 10:53:27.060	33000.00	2

PostgreSQL Simple View

```
CREATE VIEW simple_view  
AS  
SELECT name,gender,salary  
FROM employee;
```



The screenshot displays the PostgreSQL Enterprise Console interface. On the left, a tree view shows the database structure, including tables like 'employee' and 'simple_view'. The 'simple_view' table is selected, showing its columns: 'name', 'gender', and 'salary'. On the right, the SQL editor shows the following code:





```
1 CREATE VIEW simple_view  
2 AS  
3 SELECT name,gender,salary  
4 FROM employee;  
5
```

Below the SQL editor, the 'Messages' tab is active, displaying the message: 'Query returned successfully in 45 msec.'

Two red arrows highlight the workflow: one points from the 'Columns (6)' of the 'employee' table to the 'SELECT' statement in the SQL editor, and another points from the 'Columns (3)' of the 'simple_view' table to the 'CREATE VIEW' statement in the SQL editor.

PostgreSQLda Simple View

```
SELECT * FROM simple_view;
```

	Data Output	Explain	Messages	Notifications
	 name character varying (50) 	gender character varying (50) 	salary numeric (18,2) 	
1	Pranaya	Male	25000.00	
2	Priyanka	Female	30000.00	
3	Anurag	Male	40000.00	
4	Preety	Female	35000.00	
5	Sambit	Male	27000.00	
6	Hina	Female	33000.00	

PostgreSqllda Complex (*murakkab*) View

View bir nechta jadvallar asosida yaratilganda, u PostgreSQLda **complex (*murakkab*) view** sifatida yaratiladi. PostgreSQLdagi complex viewlarni tushunish uchun biz quyidagi **Department** va **Employee** jadvallaridan foydalanamiz.

Department

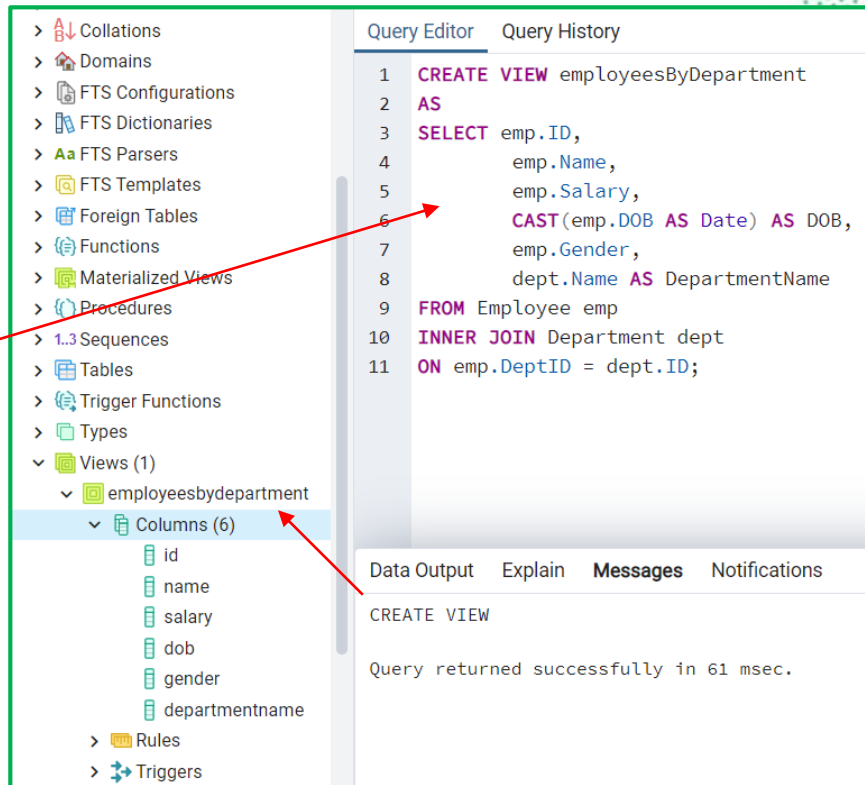
ID	Name
1	IT
2	HR
3	Sales

Employee

ID	Name	Gender	DOB	Salary	DeptID
1	Pranaya	Male	1996-02-29 10:53:27.060	25000.00	1
2	Priyanka	Female	1995-05-25 10:53:27.060	30000.00	2
3	Anurag	Male	1995-04-19 10:53:27.060	40000.00	2
4	Preety	Female	1996-03-17 10:53:27.060	35000.00	3
5	Sambit	Male	1997-01-15 10:53:27.060	27000.00	1
6	Hina	Female	1995-07-12 10:53:27.060	33000.00	2

Yuqoridagi ikkita jadval asosida complex view yaratamiz.

```
CREATE VIEW employeesByDepartment
AS
SELECT emp.ID,
       emp.Name,
       emp.Salary,
       CAST(emp.DOB AS Date) AS DOB,
       emp.Gender,
       dept.Name AS DepartmentName
FROM Employee emp
INNER JOIN Department dept
ON emp.DeptID = dept.ID;
```



The screenshot displays a database management tool interface. On the left, a tree view shows the database structure, with 'Views (1)' expanded to show 'employeesbydepartment'. The 'Columns (6)' for this view are listed: id, name, salary, dob, gender, and departmentname. On the right, the 'Query Editor' tab is active, showing the SQL code for creating the view. The code is as follows:

```
1 CREATE VIEW employeesByDepartment
2 AS
3 SELECT emp.ID,
4        emp.Name,
5        emp.Salary,
6        CAST(emp.DOB AS Date) AS DOB,
7        emp.Gender,
8        dept.Name AS DepartmentName
9 FROM Employee emp
10 INNER JOIN Department dept
11 ON emp.DeptID = dept.ID;
```

Below the query editor, the 'Messages' tab is active, displaying the following message:

```
CREATE VIEW
Query returned successfully in 61 msec.
```

Two red arrows point from the SQL code in the text box to the corresponding lines in the query editor: one from the `CAST(emp.DOB AS Date) AS DOB` line to line 6, and another from the `dept.Name AS DepartmentName` line to line 8.

Complex (*murakkab*) View

```
SELECT * FROM  
employeesByDepartment;
```

Department

ID	Name
1	IT
2	HR
3	Sales

Employee

ID	Name	Gender	DOB	Salary	DeptID
1	Pranaya	Male	1996-02-29 10:53:27.060	25000.00	1
2	Priyanka	Female	1995-05-25 10:53:27.060	30000.00	2
3	Anurag	Male	1995-04-19 10:53:27.060	40000.00	2
4	Preety	Female	1996-03-17 10:53:27.060	35000.00	3
5	Sambit	Male	1997-01-15 10:53:27.060	27000.00	1
6	Hina	Female	1995-07-12 10:53:27.060	33000.00	2

ID	Name	Salary	DOB	Gender	DepartmentName
1	Pranaya	25000.00	1996-02-29	Male	IT
2	Priyanka	30000.00	1995-05-25	Female	HR
3	Anurag	40000.00	1995-04-19	Male	HR
4	Preety	35000.00	1996-03-17	Female	Sales
5	Sambit	27000.00	1997-01-15	Male	IT
6	Hina	33000.00	1995-07-12	Female	HR

Inline View

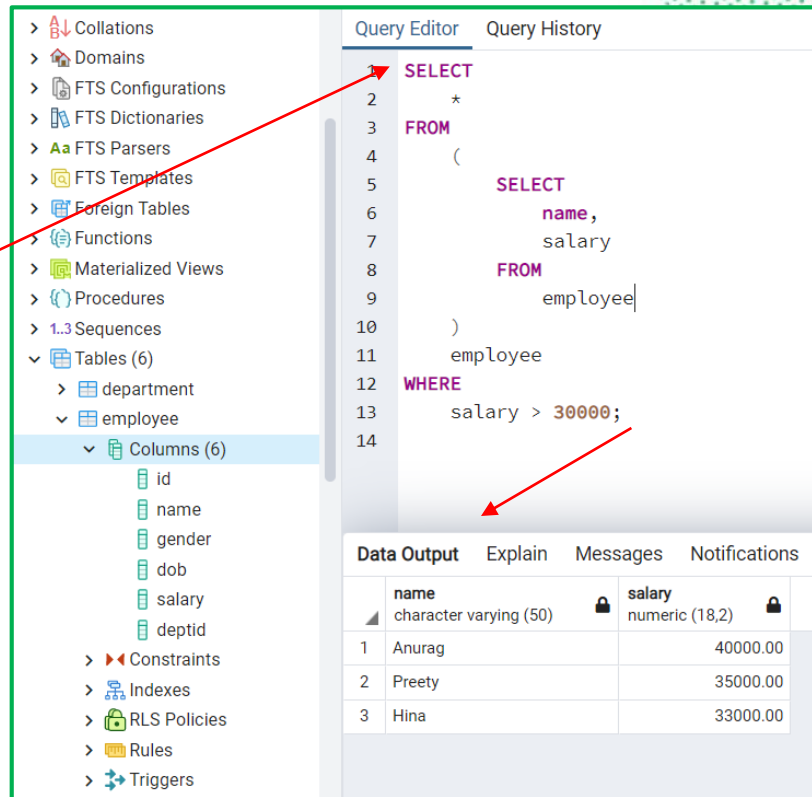
Viewlarni alohida yaratishdan tashqari, SQL so`rovlarning ichida ham view yaratish mumkin. Bu **inline view** deyiladi. Quyidagi SQL so`rovda FROM dan keyin ko`rsatilgan (*ichki so`rov*) inline view deb ataladi. Inline view so`rovdagi jadval o`rnini bosishi mumkinligi sababli, u hosila jadval deb ham ataladi. Ba`zan siz inline view bilan bir xil ma`noga ega bo`lgan subselect atamasini uchratishingiz mumkin.

Inline View

```
SELECT
    column_list
FROM
    (
        SELECT
            *
        FROM
            table_name
    ) t;
```

Inline View

```
SELECT
*
FROM
(
    SELECT
        name,
        salary
    FROM
        employee
)
employee
WHERE
    salary > 30000;
```



The screenshot shows a database query editor with a tree view on the left, a query editor on the right, and a data output table at the bottom.

Tree View:

- > Collations
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Procedures
- > 1.3 Sequences
- > Tables (6)
 - > department
 - > employee
 - > Columns (6)
 - id
 - name
 - gender
 - dob
 - salary
 - deptid
- > Constraints
- > Indexes
- > RLS Policies
- > Rules
- > Triggers

Query Editor:

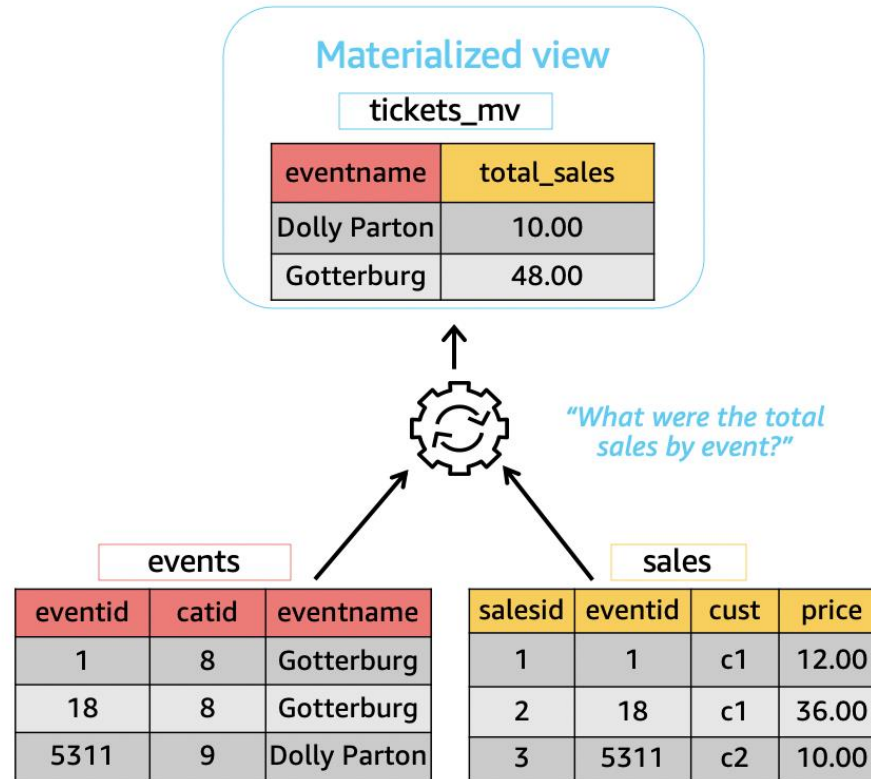
```
1 SELECT
2 *
3 FROM
4 (
5     SELECT
6         name,
7         salary
8     FROM
9         employee
10 )
11 employee
12 WHERE
13     salary > 30000;
14
```

Data Output:

	name	salary
1	Anurag	40000.00
2	Preety	35000.00
3	Hina	33000.00

Materialized View

" **Materialized View** " - bu ma'lumotlar bazasi ob'ekti bo'lib, u oldindan hisoblangan ma'lumotlar bazasi so'rovi natijasini saqlaydi va kerak bo'lganda ushbu natijani yangilashni osonlashtiradi. **Materialized Viewlar** deyarli barcha ilg'or ma'lumotlar bazasi tizimlarining ajralmas xususiyati hisoblanadi. Tabiiyki, PostgreSQL ham **Materialized Viewlarni** qo'llab-quvvatlaydi va foydalanuvchiga ko'proq vaqt talab qiladigan so'rovlarni bajarish uchun kuchli vositani taklif qiladi. PostgreSQLda **Materialized View** so'rov natijalarini saqlash va ma'lumotlarni vaqti-vaqti bilan yangilash imkonini beradi. **Materialized View** ma'lumotlarni tez olishni talab qiladigan ko'p hollarda foydalidir.



Quyida **Materialized View**ni yaratish sintaksisi keltirilgan.

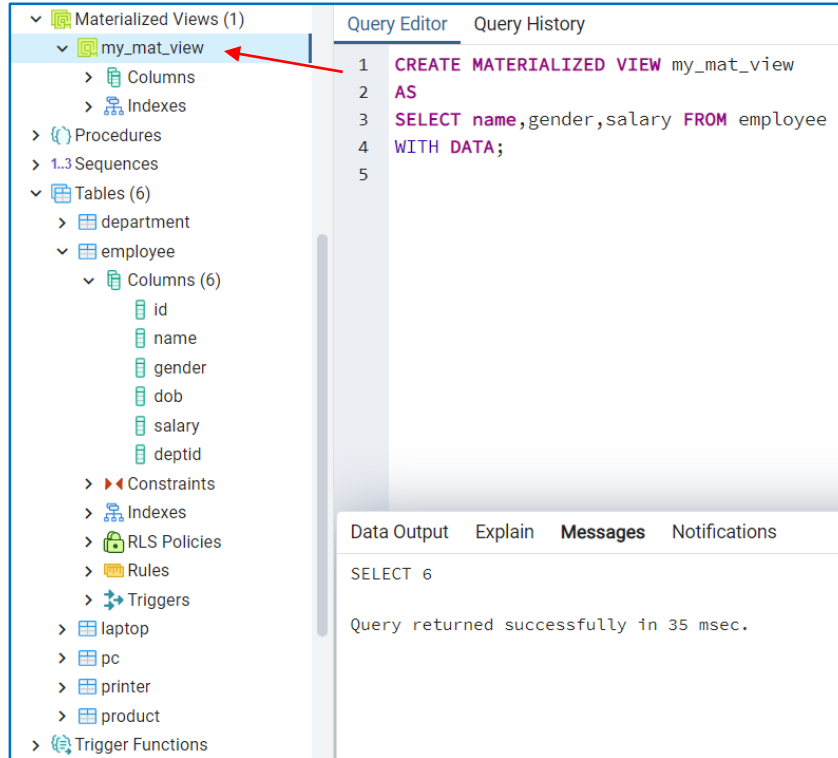
```
CREATE MATERIALIZED VIEW view_name  
AS  
query  
WITH [NO] DATA;
```

- view_name**: ko`rinish nomini belgilaydi, CREATE MATERIALIZED VIEW bandidan so`ng yoziladi.
- query**: Bu AS kalit so`zidan keyin ishlatiladi. Bu jadvallardan ma`lumotlarni oladigan so`rovni belgilaydi.
- With [NO] DATA** : [NO] kalit so`zi ixtiyoriy. Agar u aniqlanmagan bo`lsa, view yaratishda ma`lumotlar viewga yuklangan holda yaratiladi. WITH NO DATA aniqlansa, viewga ma`lumotlar yuklanmagan holda yaratiladi va view o`qilmaydi.

Quyida **Materialized View**ni yangilash sintaksisi keltirilgan.

```
REFRESH MATERIALIZED VIEW view_name;
```

Create Materialized View with data



The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including Materialized Views, Columns, Indexes, Procedures, Sequences, Tables, and Constraints. The 'my_mat_view' materialized view is selected, and a red arrow points to it. The 'Query Editor' tab is active, showing the SQL command to create the materialized view. The 'Query History' tab is also visible.

```
1 CREATE MATERIALIZED VIEW my_mat_view
2 AS
3 SELECT name,gender,salary FROM employee
4 WITH DATA;
5
```

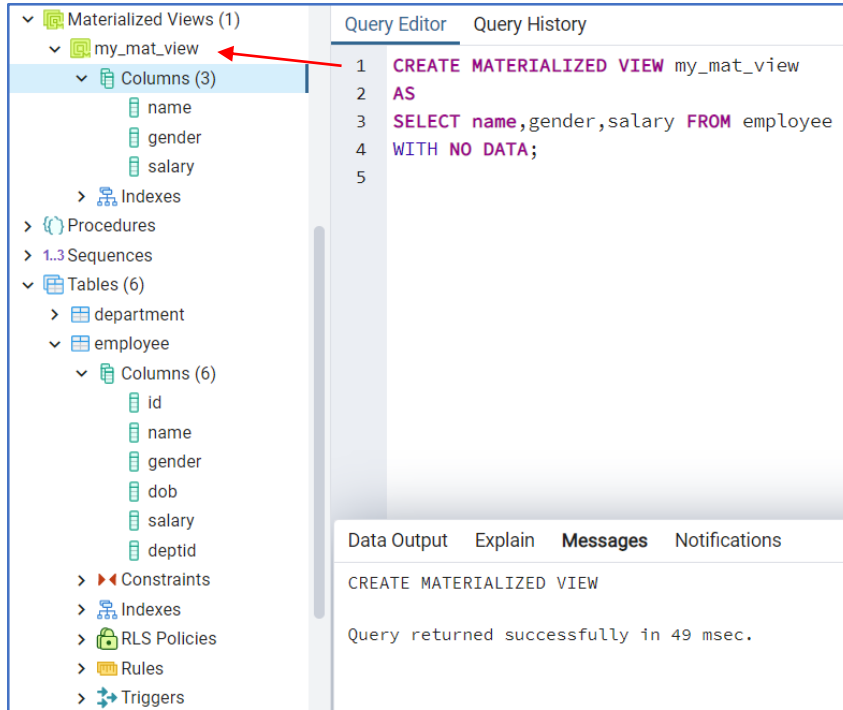
Below the query editor, the 'Data Output' tab is selected, showing the results of the query. The output is a table with 6 rows and 3 columns: name, gender, and salary. The query returned successfully in 35 msec.

	name	gender	salary
1	Pranaya	Male	25000.00
2	Priyanka	Female	30000.00
3	Anurag	Male	40000.00
4	Preety	Female	35000.00
5	Sambit	Male	27000.00
6	Hina	Female	33000.00



Data Output	Explain	Messages	Notifications
	name character varying (50)	gender character varying (50)	salary numeric (18,2)
1	Pranaya	Male	25000.00
2	Priyanka	Female	30000.00
3	Anurag	Male	40000.00
4	Preety	Female	35000.00
5	Sambit	Male	27000.00
6	Hina	Female	33000.00

Create Materialized View with no data



The screenshot shows a database management tool interface. On the left, a tree view displays the database structure. Under 'Materialized Views (1)', 'my_mat_view' is selected, and its columns (name, gender, salary) are visible. The 'Query Editor' tab on the right shows the following SQL code:

```
1 CREATE MATERIALIZED VIEW my_mat_view
2 AS
3 SELECT name,gender,salary FROM employee
4 WITH NO DATA;
5
```

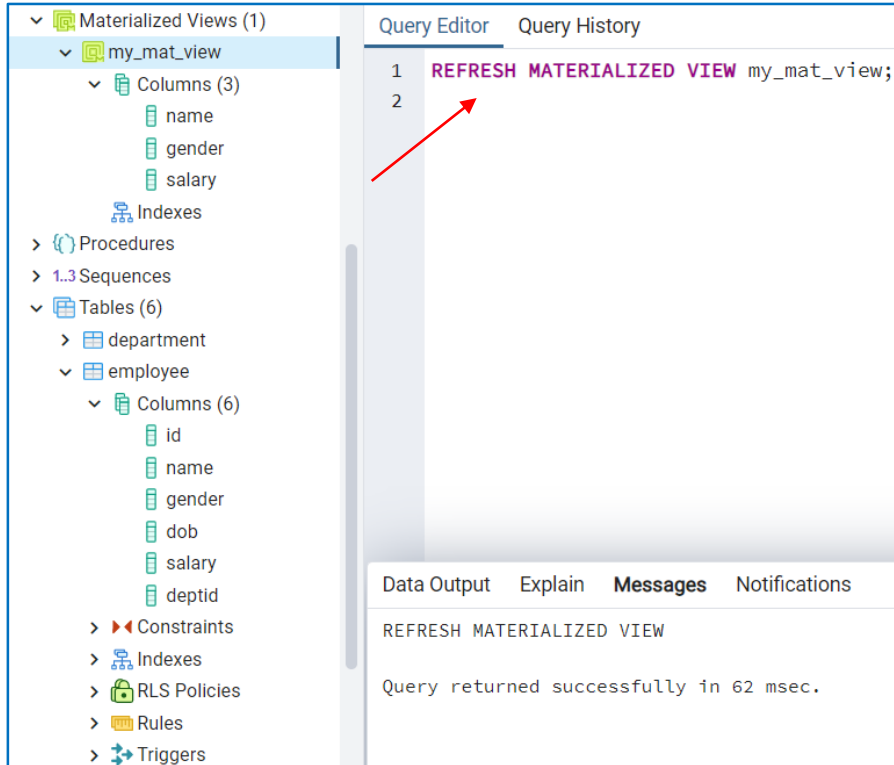
Below the query editor, the 'Messages' tab shows the following output:

```
CREATE MATERIALIZED VIEW
Query returned successfully in 49 msec.
```



Data Output	Explain	Messages	Notifications
ERROR: materialized view "my_mat_view" has not been populated HINT: Use the REFRESH MATERIALIZED VIEW command. SQL state: 55000			

Refresh Materialized View



Materialized Views (1)

- my_mat_view
 - Columns (3)
 - name
 - gender
 - salary
 - Indexes
- Procedures
- Sequences
- Tables (6)
 - department
 - employee
 - Columns (6)
 - id
 - name
 - gender
 - dob
 - salary
 - deptid
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers

Query Editor Query History

```
1 REFRESH MATERIALIZED VIEW my_mat_view;  
2
```

Data Output Explain Messages Notifications

REFRESH MATERIALIZED VIEW

Query returned successfully in 62 msec.



	Data Output	Explain	Messages	Notifications
	name character varying (50)		gender character varying (50)	salary numeric (18,2)
1	Pranaya		Male	25000.00
2	Priyanka		Female	30000.00
3	Anurag		Male	40000.00
4	Preety		Female	35000.00
5	Sambit		Male	27000.00
6	Hina		Female	33000.00

Managing PostgreSQL Views

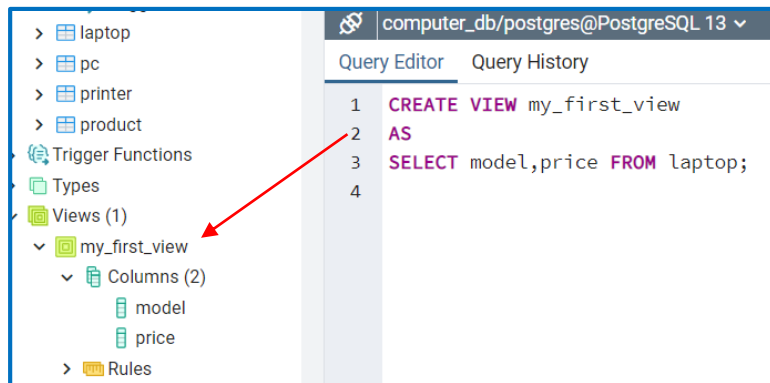
Managing PostgreSQL Views ya`ni PostgreSQLda viewlarni boshqarish. Quyida view qanday yaratilishi, view qanday o`zgartirilishi va view qanday o`chirilishini ko`rib chiqamiz.

CREATE,
UPDATABLE,
DROP

Creating PostgreSQL Views

```
CREATE VIEW view_name AS query;
```

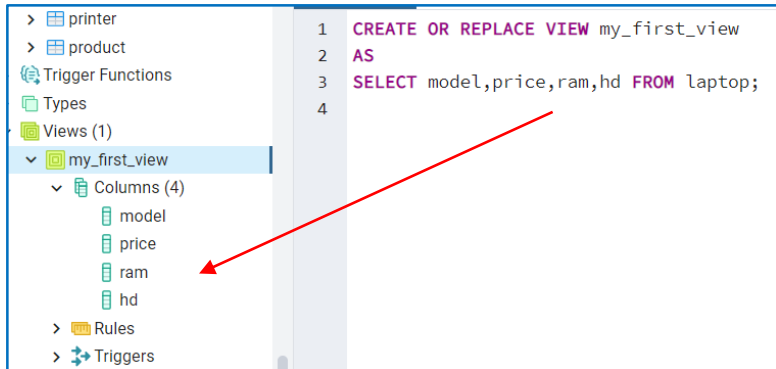
Yangi view **CREATE VIEW** buyrug'i orqali yaratiladi. So'ngra **AS** kalit so'zidan keyin query (so'rov) yoziladi. **Masalan:**



Updating PostgreSQL Views

```
CREATE OR REPLACE view_name  
AS  
query
```

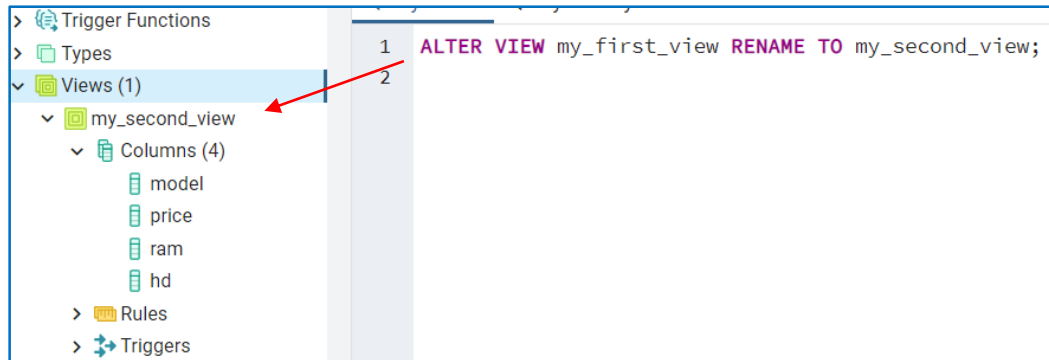
Viewni o'zgartirish uchun **CREATE OR REPLACE VIEW** buyrug'i yoziladi. So'ngra **AS** kalit so'zidan keyin query (so'rov) yoziladi. Bunda view mavjud bo'lsa o'zgartiradi, agar yo'q bo'lsa yangi view yaratadi. **Masalan:**



Updating PostgreSQL Views

```
ALTER VIEW customer_master RENAME TO customer_info;
```

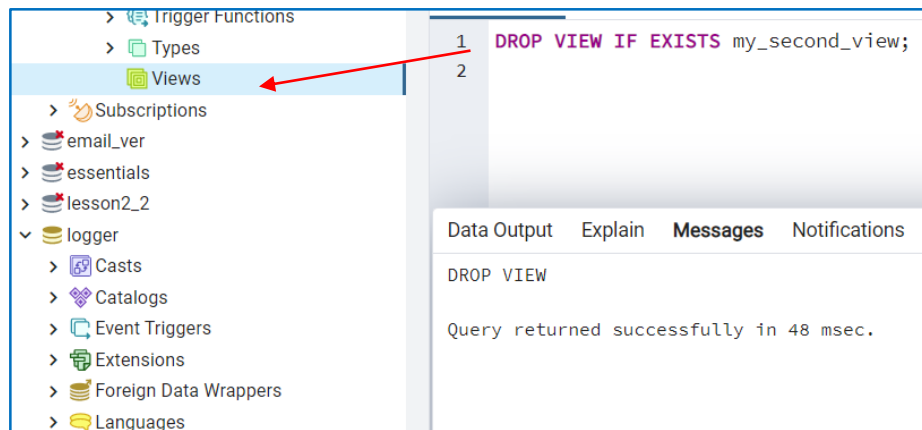
Viewni nomini o'zgartirish uchun **ALTER VIEW** dan keyin eski view nomi va **RENAME TO** dan keyin yangi view nomikiritiladi.
Masalan:



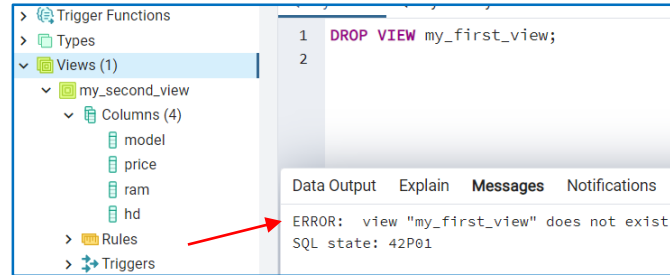
Removing PostgreSQL Views

```
DROP VIEW [ IF EXISTS ] view_name;
```

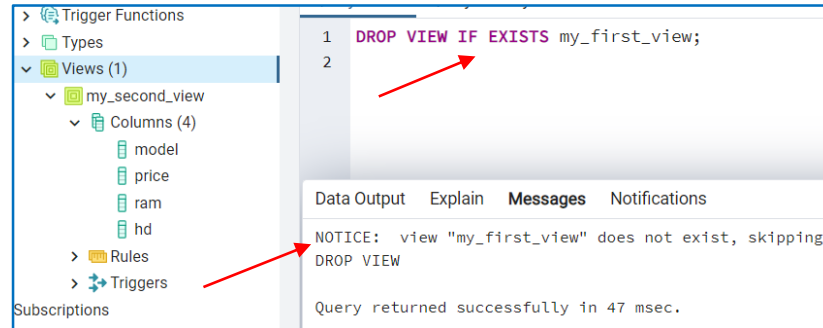
Viewni o`chirish uchun **DROP VIEW** buyrug`idan foydalaniladi, lekin o`chirmoqchi bo`lgan view mavjud bo`lmasa PostgreSQL xatolik beradi. Buning oldini olish uchun **IF EXISTS** dan foydalaniladi. **Masalan:**



Removing PostgreSQL Views



The screenshot shows the PostgreSQL Enterprise Studio interface. On the left, the 'Views (1)' folder is expanded, showing 'my_second_view' with columns 'model', 'price', 'ram', and 'hd'. A red arrow points from the 'Rules' folder to the 'Messages' tab. The 'Messages' tab displays an error message: 'ERROR: view "my_first_view" does not exist SQL state: 42P01'. The SQL editor on the right contains the command: `DROP VIEW my_first_view;`



The screenshot shows the PostgreSQL Enterprise Studio interface. On the left, the 'Views (1)' folder is expanded, showing 'my_second_view' with columns 'model', 'price', 'ram', and 'hd'. A red arrow points from the 'Rules' folder to the 'Messages' tab. The 'Messages' tab displays a notice message: 'NOTICE: view "my_first_view" does not exist, skipping DROP VIEW'. The SQL editor on the right contains the command: `DROP VIEW IF EXISTS my_first_view;`. Below the messages, it states 'Query returned successfully in 47 msec.'

E`TIBORINGIZ UCHUN RAHMAT!