



PostgreSQL Tutorial

Using pgAdmin4

by Feni Ismiati





Background and Objective

PostgreSQL that is managed via pgAdmin 4, does not have a built-in limit on how long data is stored in your database. But Google BigQuery is cloud-based services which might have data retention policies or limitations based on your billing or usage tier. PostgreSQL gives us control over data retention through its configuration, queries, and management.

In PostgreSQL, we can manage data retention manually, using SQL commands and procedures to archive or delete data as needed. This flexibility allows to keep data for as long as we want, assuming there is sufficient storage resources.

However, you are responsible for setting up and managing any data retention policies or practices that you need. This tutorial explain how to setting up the database in PostgreSQL via pgAdmin 4.

Table of Content

So, here are a few steps to save your database in PostgreSQL through pgAdmin 4

1

Create Table &
Columns

2

Import Data

3

Data Type
Adjustment

4

Date Format
Adjustment

5

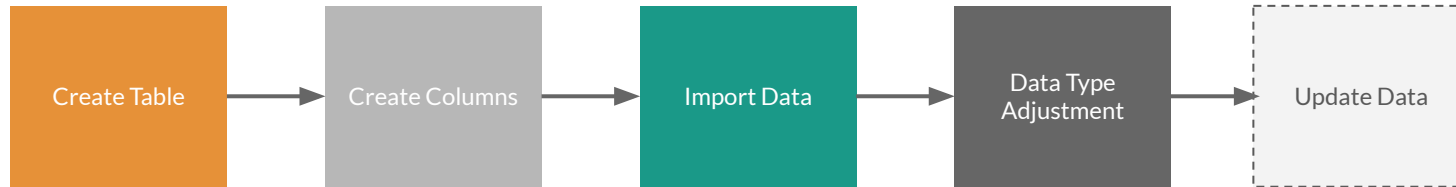
Add New Data

6

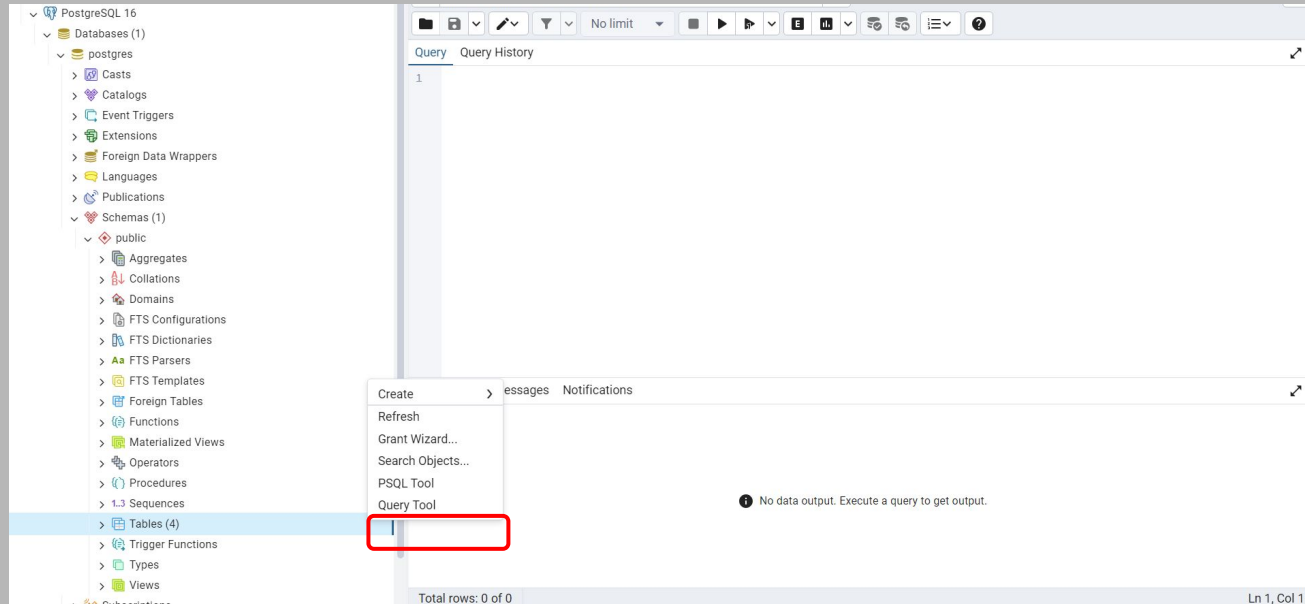
Adjust Existing
Data



Flow Process



Create Table & Columns



On the Tables section, click
“Query Tool” to make the table
and corresponding columns
through SQL script

Query Query History

```
1 create table transaction_detail (po_number text,transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text,seller_category text, buyer_id text)  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 62 msec.

Total rows: 0 of 0 Query complete 00:00:00.062 Ln 1, Col 1

Copy and paste all the column name in query + the data type, then run the query

```
CREATE TABLE transaction_detail (po_number text,transaction_date text, order_id text,  
order_status text, payment_group text, payment_method text, shipping_agency text,  
shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
revenue integer, seller_id text,seller_category text, buyer_id text)
```


The screenshot displays a database management tool interface. On the left, a tree view shows the database structure, including Schemas, public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (6), Trigger Functions, Types, Views, Subscriptions, and Login/Group Roles. The 'Tables (6)' folder is selected, and a context menu is open with the 'Refresh' option highlighted. The main pane shows a SQL query in the 'Query' tab, which is a CREATE TABLE statement for 'transaction_detail'. The query is highlighted in blue. Below the query, the 'Data Output' tab is active, showing the message 'Query returned successfully in 62 msec.'.

```
1 create table transaction_detail (po_number text, transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text, seller_category text, buyer_id text)  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

CREATE TABLE

Query returned successfully in 62 msec.

After running, make sure the query is success. Then right click on the Tables, and click "Refresh" to show the new table

The screenshot shows a database management tool interface. On the left is a tree view of the database schema, including 'public', 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (7)'. The 'transaction_detail' table is highlighted in the 'Tables' section.

The main area displays a SQL query in the 'Query' tab:

```
1 create table transaction_detail (po_number text,transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text,seller_category text, buyer_id text)  
5  
6  
7 select *  
8 from transaction_detail  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

The 'Data Output' tab shows the result of the query, which is a table with the following columns:

po_number	transaction_date	order_id	order_status	payment_group	payment_method	shipping_agency	shipping_cost	total_project_value
text	text	text	text	text	text	text	text	integer

A red box highlights the 'select * from transaction_detail' query, and a red arrow points from it to the 'Data Output' tab.

Then run the script to show all the columns in the corresponding table using select syntax

This is a close-up of the 'Data Output' tab from the previous screenshot. It shows the table structure with columns and their data types, each with a lock icon:

po_number	transaction_date	order_id	order_status	payment_group	payment_method	shipping_agency	shipping_cost	total_project_value
text	text	text	text	text	text	text	text	integer

A red box highlights the entire table structure.

Import Data

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including 'Publications', 'Schemas (1)', and 'public'. Under 'public', there are various database objects like 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (7)'. The 'transaction_detail' table is selected. A context menu is open over the table, with the 'Import/Export Data...' option highlighted. The main query editor shows the following SQL code:

```
1 create table transaction_detail (po_number text,transaction_date text, order_id text,  
2 order_status text,payment_group text,payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text,seller_category text, buyer_id text)  
5  
6  
7 select *  
8 from transaction_detail
```

Below the query editor, there is a table schema view showing the columns and their data types:

po_number	transaction_date	order_id	order_status	payment_group	payment_method	shipping_agency	shipping_cost	total_project_value
text	text	text	text	text	text	text	text	integer

The dialog box is titled 'Import/Export data - table 'transaction_detail''. It has three tabs: 'General', 'Options', and 'Columns'. The 'General' tab is active. It contains the following fields and controls:

- Import/Export:** Two buttons, 'Import' (selected) and 'Export'.
- Filename:** A text field containing 'order_transaction.csv' and a file selection icon.
- Format:** A dropdown menu set to 'csv'.
- Encoding:** A dropdown menu set to 'Select an item...'.
- Buttons:** 'Close', 'Reset', and 'OK' at the bottom right.

The create table syntax is only to create the table including the corresponding columns. After that, we still need to import dataset on the columns in csv format. So the step is click the “Import/Export Data” and import the csv file from your computer.

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including 'Publications', 'Schemas (1)', and 'public'. Under 'public', there are 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (7)'. The 'transaction_detail' table is selected. A right-click context menu is open, with 'Import/Export Data...' highlighted. The main window shows a SQL query editor with the following code:

```
1 create table transaction_detail (po_number text, transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text, seller_category text, buyer_id text)  
5  
6  
7 select *  
8 from transaction_detail
```

Below the query editor, a table structure is displayed with columns: po_number (text), transaction_date (text), order_id (text), order_status (text), payment_group (text), payment_method (text), shipping_agency (text), shipping_cost (text), total_project_value (integer), voucher_val (text), voucher_code (text), revenue (integer), seller_id (text), seller_category (text), and buyer_id (text).

The screenshot shows the 'Import/Export data - table 'transaction_detail'' dialog box. The 'Columns' tab is selected. The 'Columns to import' section lists the following columns: po_number, transaction_date, order_id, order_status, payment_group, payment_method, shipping_agency, shipping_cost, total_project_value, voucher_val, voucher_code, revenue, seller_id, seller_category, and buyer_id. Below this, there is a section for 'NOT NULL columns' with a dropdown menu set to 'Not null columns...'. At the bottom, there are buttons for 'Close', 'Reset', and 'OK'.

Before clicking “OK”, check the “Columns” section to ensure that all the columns needed are already match. Then click “OK”

The screenshot shows a database management interface. At the top is a toolbar with icons for file operations, filters, and execution. Below the toolbar is a 'Query' tab with a text editor containing the following SQL code:

```
1 create table transaction_detail (po_number text, transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text, seller_category text, buyer_id text)  
5  
6  
7 select *  
8 from transaction_detail  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

Below the query editor is a 'Data Output' tab showing a table with three columns: 'po_number' (text), 'transaction_date' (text), and 'order_id' (text). The table is currently empty. Overlaid on the interface are two notifications:

- Process failed** (red background):
Copying table data 'public.transaction_detail' on database 'postgres' and server 'PostgreSQL 16 (localhost:5432)'
[View Processes]
- Process started** (green background):
Copying table data 'public.transaction_detail' on database 'postgres' and server 'PostgreSQL 16 (localhost:5432)'
[View Processes]

Three red exclamation marks are positioned above the 'Process failed' notification.

Check the process result, if the process failed, we should checking the reason by clicking **“View Processes”**

Data Type Adjustment

Search

<input type="checkbox"/>		PID	Type	Server	Object	Start Time ▾	Status	Time
<input type="checkbox"/>		27312	Import Data	PostgreSQL 16 (loca...	postgres/public.tran...	8/30/2024, 5:13:56 ...	Failed	0.4

Process Watcher - Import - Copying table data

✕

Copying table data 'public.transaction_detail' on database 'postgres' and server 'PostgreSQL 16 (localhost:5432)'
Running command:

```
--command " \copy public.transaction_detail (po_number, transaction_date, order_id, order_status, payment_group, payment_method, shipping_agency, shipping_cost, total_project_value, voucher_val, voucher_code, revenue, seller_id, seller_category, buyer_id) FROM 'C:/Users/Lenovo/DOCUME~1/REVOUX~1/ORDER_~2.CSV' DELIMITER ',' CSV HEADER QUOTE '\"' ESCAPE '\n', '\r', '\t' "
```

Start time: Fri Aug 30 2024 17:13:56 GMT+0700 (Indochina Time) Stop Process

ERROR: value "2969805000" is out of range for type integer
CONTEXT: COPY transaction_detail, line 383, column total_project_value: "2969805000"

Failed (exit code: 1). Execution time: 0.4 seconds

Click on the file icon to see the error details. In the error prompt it mentioned that the one value in the data 2,969,805,000 which is out of range for type integer

SOLUTION!

The "total_project_value" column needs to store larger values, we should change data type to **bigint**. Because **bigint** type can hold values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

Query Query History

```
1 create table transaction_detail (po_number text, transaction_date text, order_id text,
2 order_status text, payment_group text, payment_method text, shipping_agency text,
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,
4 revenue integer, seller_id text, seller_category text, buyer_id text)
5
6
7 select *
8 from transaction_detail
9
10 --change data type from "integer" to "bigint"
11 alter table transaction_detail
12 alter column total_project_value type bigint
13
14
15
16
17
18
```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 103 msec.

✓ Query returned successfully in 103 msec. ✕

Run the script to change the data type from integer to bigint

alter table transaction_detail
alter column total_project_value type bigint

Date Format Adjustment

Change the data type from "text" to "date". In this case, "MM/DD/YYYY" is used for format date

Query

```
1 create table transaction_detail (po_number text, transaction_date text, order_id text,  
2 order_status text, payment_group text, payment_method text, shipping_agency text,  
3 shipping_cost text, total_project_value integer, voucher_val text, voucher_code text,  
4 revenue integer, seller_id text, seller_category text, buyer_id text)  
5  
6 select *  
7 from transaction_detail  
8  
9 --change data type from "integer" to "bigint"  
10 alter table transaction_detail  
11 alter column total_project_value type bigint  
12  
13  
14 ---Change to date format  
15 alter table transaction_detail  
16 alter column transaction_date type date  
17 using to_date(transaction_date, 'MM/DD/YYYY');  
18  
19
```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 1 secs 210 msec.

	po_number text	transaction_date text	order_id text
17	PO-2022-07-14-140490	2022-07-14	173117
18	PO-2022-04-08-105769	2022-04-08	58732
19	PO-2022-03-26-101168	2022-03-26	232921

	po_number text	transaction_date date	order_id text
1	PO-2022-12-12-8804434	2022-12-12	463776
2	PO-2022-02-02-83227	2022-02-02	84817
3	PO-2022-01-21-80758	2022-01-21	70055

```
alter table transaction_detail  
alter column transaction_date type date  
using to_date(transaction_date, 'MM/DD/YYYY');
```

Update Data



There are 2 types of updating data:

1. **Add a new data records**

Adding new data records in the existing columns by importing the CSV file into a temporary or staging table that has the same structure / columns as the main table

2. **Adjust the existing data**

1. Add a new data records

```
21
22 --UPDATE DATA
23 --1. add new records
24 create table temp_table (po_number text, transaction_date date, order_id text,
25 order_status text, payment_group text, payment_method text, shipping_agency text,
26 shipping_cost text, total_project_value bigint, voucher_val text, voucher_code text,
27 revenue integer, seller_id text, seller_category text, buyer_id text)
28
29 select *
30 from temp_table
31
32
33
34
35
36
37
38
39
40
```

	po_number text	transaction_date date	order_id text	order_status text	payment_group text	payment_method text	shipping_agency text	shipping_cost text	total_project_value bigint
1	PO-2023-01-01-1234567	2023-01-01	463776	Selesai	TOP	mandiriva	Kurir Pribadi	0	
2	PO-2023-01-02-1234568	2023-02-01	463777	Selesai	TOP	mandiriva	Kurir Pribadi	0	
3	PO-2023-01-03-1234569	2023-03-01	463778	Selesai	TOP	mandiriva	Kurir Pribadi	0	
4	PO-2023-01-04-1234570	2023-04-01	463779	Selesai	TOP	mandiriva	Kurir Pribadi	0	

In this case, create new table which contains the new records is needed by using create table script. Create the table by naming temporary table “temp_table” from the csv file that you wanted to add to main table. **Please note that the structure or columns must be same with the main table**

Repeat the step by importing the data, until the data are shown in the temporary table


```
32
33
34 --add to main table
35 insert into transaction_detail (
36     po_number, transaction_date, order_id,
37     order_status, payment_group, payment_method, shipping_agency,
38     shipping_cost, total_project_value, voucher_val, voucher_code,
39     revenue, seller_id, seller_category, buyer_id
40 )
41 select
42     po_number, transaction_date, order_id,
43     order_status, payment_group, payment_method, shipping_agency,
44     shipping_cost, total_project_value, voucher_val, voucher_code,
45     revenue, seller_id, seller_category, buyer_id
46 from temp_table
47 where not exists (
48     select 1 from transaction_detail
49     where transaction_detail.po_number = temp_table.po_number
50 );
51
```

Data Output Messages Notifications

INSERT 0 0

Query returned successfully in 137 msec.

Update the main table using the data from the temporary table is depending on your needs, you might perform an “UPDATE”, “INSERT”, or “MERGE” operation. In this case, because we want to add new records to main table “transaction_detail”, so we use “INSERT” operations.

```
INSERT INTO transaction_detail (
    po_number, transaction_date, order_id,
    order_status, payment_group, payment_method, shipping_agency,
    shipping_cost, total_project_value, voucher_val, voucher_code,
    revenue, seller_id, seller_category, buyer_id
)
SELECT
    po_number, transaction_date, order_id,
    order_status, payment_group, payment_method, shipping_agency,
    shipping_cost, total_project_value, voucher_val, voucher_code,
    revenue, seller_id, seller_category, buyer_id
FROM temp_table
WHERE NOT EXISTS (
    SELECT 1 FROM transaction_detail
    WHERE transaction_detail.po_number = temp_table.po_number
);
```

The screenshot shows a SQL IDE interface. The top toolbar contains icons for file operations, query execution, and settings. The main editor displays a SQL query with line numbers 40 to 58. The query inserts data into a table and then checks for existing records. A red box highlights the check query. Below the editor, the 'Data Output' tab shows a table with 10 columns and 4 rows of data. A red box highlights the first four rows of the table.

```
40 )
41 select
42     po_number, transaction_date, order_id,
43     order_status, payment_group, payment_method, shipping_agency,
44     shipping_cost, total_project_value, voucher_val, voucher_code,
45     revenue, seller_id, seller_category, buyer_id
46 from temp_table
47 where not exists (
48     select 1 from transaction_detail
49     where transaction_detail.po_number = temp_table.po_number
50 );
51
52
53 --checking
54 select *
55 from transaction_detail
56 where transaction_date between '2023/01/01' and '2023/12/31'
57
58
```

Data Output

	po_number text	transaction_date date	order_id text	order_status text	payment_group text	payment_method text	shipping_agency text	shipping_cost text	total_project bigint
1	PO-2023-01-01-1234567	2023-01-01	463776	Selesai	TOP	mandiriva	Kurir Pribadi	0	
2	PO-2023-01-02-1234568	2023-02-01	463777	Selesai	TOP	mandiriva	Kurir Pribadi	0	
3	PO-2023-01-03-1234569	2023-03-01	463778	Selesai	TOP	mandiriva	Kurir Pribadi	0	
4	PO-2023-01-04-1234570	2023-04-01	463779	Selesai	TOP	mandiriva	Kurir Pribadi	0	

After the operation is success, checking the “transaction_detail” is needed to see whether the new records has already been added

As of the new records are the transaction date on 2023, then we will use the script “WHERE” the transaction date was made from January to December 2023

2. Adjust the existing data

Object Explorer

postgres 16.sql* X

postgres/postgres@PostgreSQL 16

Query Query History

```
56  
57 ---2. ADJUST DATA (update the existing)  
58 create table updated_table (po_number text, transaction_date date, order_id text,  
59 order_status text, payment_group text, payment_method text, shipping_agency text,  
60 shipping_cost text, total_project_value bigint, voucher_val text, voucher_code text,  
61 revenue integer, seller_id text, seller_category text, buyer_id text)  
62  
63 select *  
64 from updated_table  
65  
66  
67  
68  
69  
70  
71  
72  
73
```

Data Output Messages Notifications

	po_number text	transaction_date date	order_id text	order_status text	payment_group text	payment_method text	shipping_agency text	shipping_cost text	total_project_value bigint
1	PO-2022-01-05-76470	2022-01-05	165689	Selesai	Direct	mips-bniva	JNE	30000	24
2	PO-2022-01-05-76480	2022-01-05	165618	Selesai	Direct	QREN	self	0	0
3	PO-2022-01-18-79546	2022-01-18	255059	Selesai	Direct	bnl_ecolL_va	self	0	1

✓ Successfully run. Total query runtime: 99 msec. 3 rows affected. ✕

OBJECTIVE

In this step, we are trying to update the data from **updated_table** to main table (**transaction_detail**) which contains the updated value in column **order_status** from “Permintaan Perubahan” to “Selesai” in January 2022.

The steps have similar process, first you need to create a new table and import the updated data

postgres/postgres@PostgreSQL 16

Query Query History

```
56
57 ---2. ADJUST DATA (update the existing)
58 create table updated_table (po_number text, transaction_date date, order_id text,
59 order_status text, payment_group text, payment_method text, shipping_agency text,
60 shipping_cost text, total_project_value bigint, voucher_val text, voucher_code text,
61 revenue integer, seller_id text, seller_category text, buyer_id text)
62
63 select *
64 from updated_table
65
66
67 --checking 'Permintaan Perubahan' value in transaction_detail table
68 select *
69 from transaction_detail
70 where transaction_date between '2022/01/01' and '2022/01/31'
71 and order_status = 'Permintaan Perubahan'
72
73
```

Data Output Messages Notifications

	po_number text	transaction_date date	order_id text	order_status text	payment_group text	payment_method text	shipping_agency text	shipping_cost text	total_p bigint
1	PO-2022-01-05-76470	2022-01-05	165689	Permintaan Perubahan	Direct	mps-bniva	JNE	30000	
2	PO-2022-01-05-76480	2022-01-05	165618	Permintaan Perubahan	Direct	QREN	self	0	
3	PO-2022-01-18-79546	2022-01-18	255059	Permintaan Perubahan	Direct	bnl_ecoll_va	self	0	

✓ Successfully run. Total query runtime: 157 msec. 3 rows affected. ✕

In main table “transaction_detail”, there are 3 values of ‘Permintaan Perubahan’ in order_status column, in January 2022

postgres/postgres@PostgreSQL 16

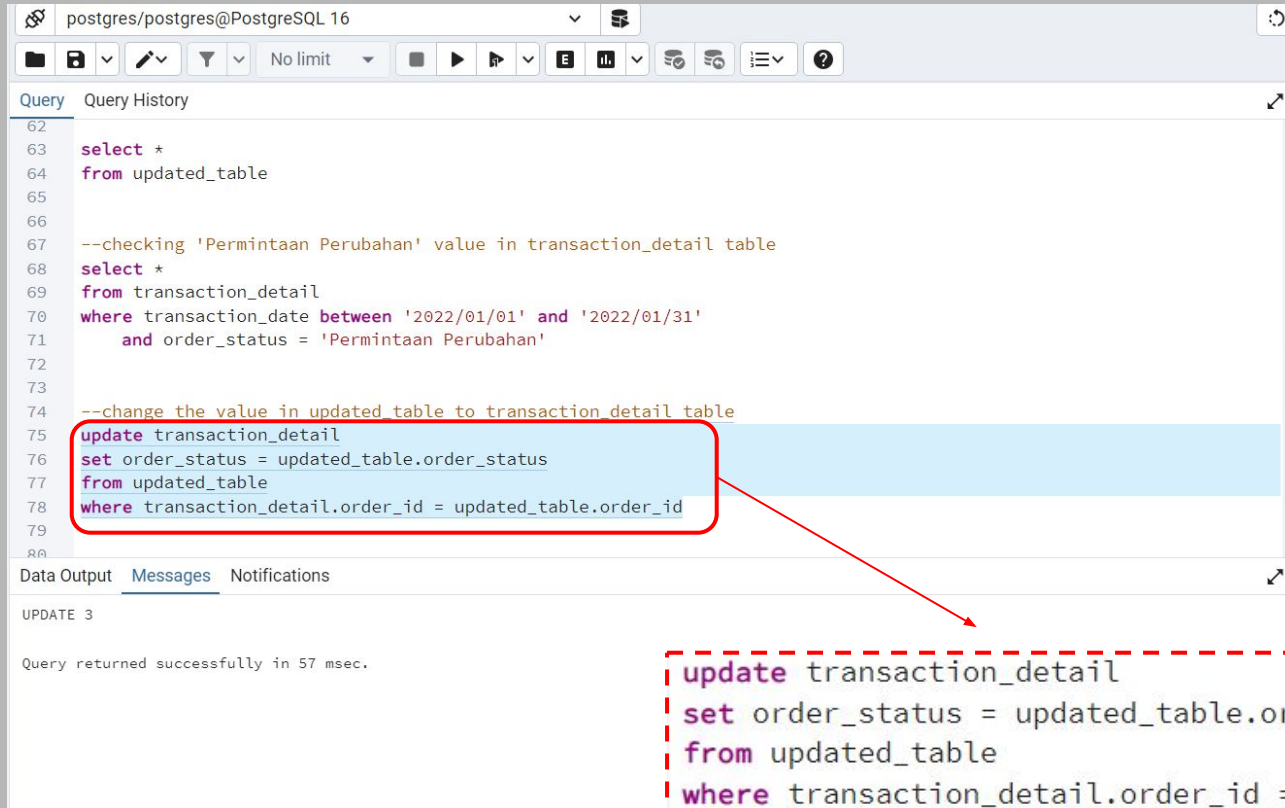
Query History

```
62
63 select *
64 from updated_table
65
66 --checking 'Permintaan Perubahan' value in transaction_detail table
67 select *
68 from transaction_detail
69 where transaction_date between '2022/01/01' and '2022/01/31'
70 and order_status = 'Permintaan Perubahan'
71
72
73
74 --change the value in updated table to transaction detail table
75 update transaction_detail
76 set order_status = updated_table.order_status
77 from updated_table
78 where transaction_detail.order_id = updated_table.order_id
79
80
```

Data Output Messages Notifications

UPDATE 3

Query returned successfully in 57 msec.



Update the value in updated_table to transaction detail, by running the following syntax

```
update transaction_detail
set order_status = updated_table.order_status
from updated_table
where transaction_detail.order_id = updated_table.order_id
```

postgres/postgres@PostgreSQL 16

Query Query History

```
72
73
74 --change the value in updated_table to transaction_detail table
75 update transaction_detail
76 set order_status = updated_table.order_status
77 from updated_table
78 where transaction_detail.order_id = updated_table.order_id
79
80
81 --checking 'Permintaan Perubahan' value
82 select *
83 from transaction_detail
84 where transaction_date between '2022/01/01' and '2022/01/31'
85 and order_status = 'Permintaan Perubahan'
86
87
88
89
90
```

Data Output Messages Notifications

po_number	transaction_date	order_id	order_status	payment_group	payment_method	shipping_agency	shipping_cost	total_project_value
text	date	text	text	text	text	text	text	bigint

✓ Successfully run. Total query runtime: 148 msec. 0 rows affected. ✕

Run the syntax to check if there is still 'Permintaan Perubahan' value. The result shows that there is no value of 'Permintaan Perubahan' in the January 2022

postgres/postgres@PostgreSQL 16

No limit

Query Query History

```
81 --checking 'Permintaan Perubahan' value
82 select *
83 from transaction_detail
84 where transaction_date between '2022/01/01' and '2022/01/31'
85       and order_status = 'Permintaan Perubahan'
86
87
88 --create groupby to count the order_status in January 2022
89 select
90     order_status,
91     count(distinct order_id) as total_status
92 from transaction_detail
93 group by order_status
94 order by count(distinct order_id) asc
95
96
97
98
```

Data Output Messages Notifications

	order_status text	total_status bigint
1	Tagihan Diterima	3
2	Permintaan Upload Document	3
3	Permintaan Perubahan	22
4	Menunggu Persetujuan	27
5	Pembayaran Terverifikasi	34
6	Dibayar	74

Total rows: 22 of 22 Query complete 00:00:01.780 Ln 88, Col 59

Double checking with group by function. The result shows that there is no 'Permintaan Perubahan' value in the order_status column. Sort by ascending value

THANK YOU!

Feni Ismiati | 6282210023177 | ismatifeni6@gmail.com

