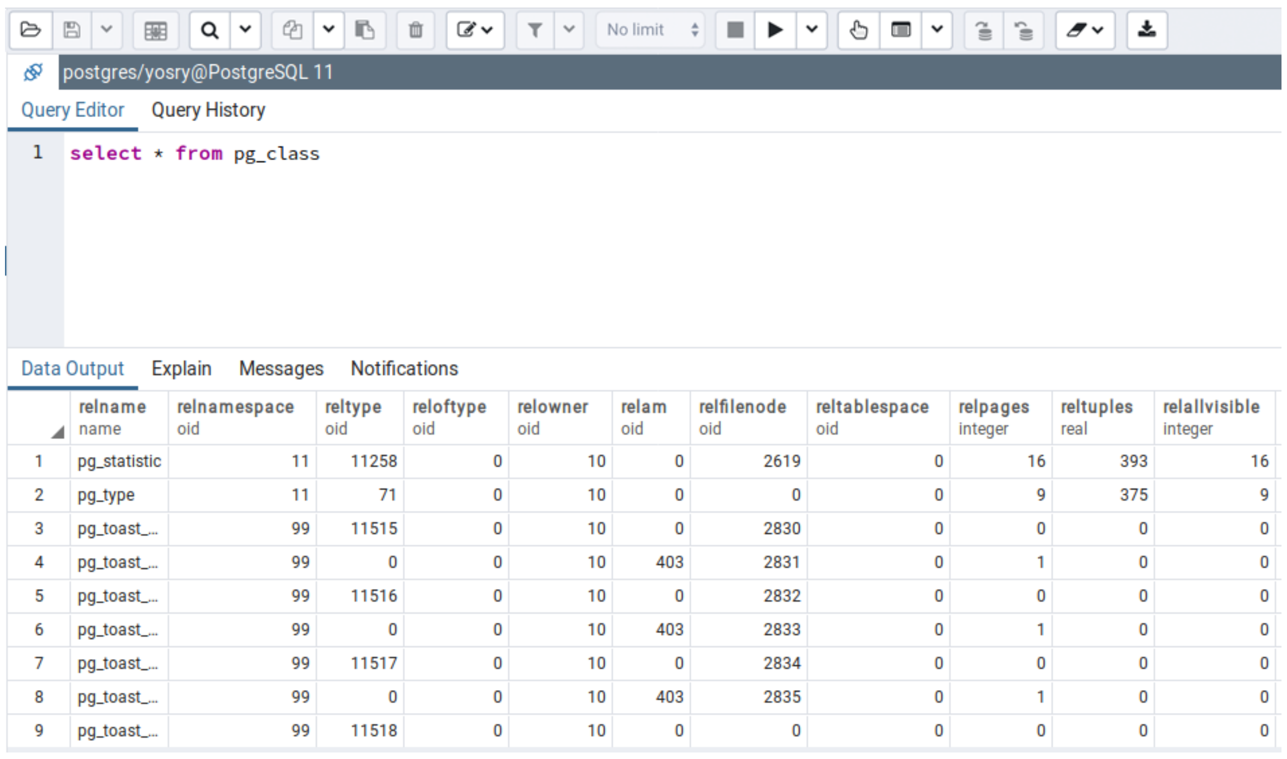
Warning: This documentation is for a pre-release version of pgAdmin 4

Contents

Query Tool ¶

The Query Tool is a powerful, feature-rich environment that allows you to execute arbitrary SQL commands and review the result set. You can access the Query Tool via the *Query Tool* menu option on the *Tools* menu, or through the context menu of select nodes of the Browser tree control. The Query Tool allows you to:

- Issue ad-hoc SQL queries.
- Execute arbitrary SQL commands.
- Edit the result set of a SELECT query if it is updatable.
- Displays current connection and transaction status as configured by the user.
- Save the data displayed in the output panel to a CSV file.
- Review the execution plan of a SQL statement in either a text, a graphical format or a table format (similar to https://explain.depesz.com).
- View analytical information about a SQL statement.



```
select * from pg_class
```

	relname name	relnamespace oid	reltype oid	reloftype oid	relowner oid	relam oid	relfilenode oid	reltablespace oid	relpages integer	reltuples real	relallvisible integer
1	pg_statistic	11	11258	0	10	0	2619	0	16	393	16
2	pg_type	11	71	0	10	0	0	0	9	375	9
3	pg_toast_...	99	11515	0	10	0	2830	0	0	0	0
4	pg_toast_...	99	0	0	10	403	2831	0	1	0	0
5	pg_toast_...	99	11516	0	10	0	2832	0	0	0	0
6	pg_toast_...	99	0	0	10	403	2833	0	1	0	0
7	pg_toast_...	99	11517	0	10	0	2834	0	0	0	0
8	pg_toast_...	99	0	0	10	403	2835	0	1	0	0
9	pg_toast_...	99	11518	0	10	0	0	0	0	0	0

You can open multiple copies of the Query tool in individual tabs simultaneously. To close a copy of the Query tool, click the *X* in the upper-right hand corner of the tab bar.

The Query Tool features two panels:

- The upper panel displays the *SQL Editor*. You can use the panel to enter, edit, or execute a query. It also shows the *History* tab which can be used to view the queries that have been executed in the session, and a *Scratch Pad* which can be used to hold text snippets during editing. If the Scratch Pad is closed, it can be re-opened (or additional ones opened) by right-clicking in the SQL Editor and other panels and adding a new panel.
- The lower panel displays the *Data Output* panel. The tabbed panel displays the result set returned by a query, information about a query's execution plan, server messages related to the query's execution and any asynchronous notifications received from the server.

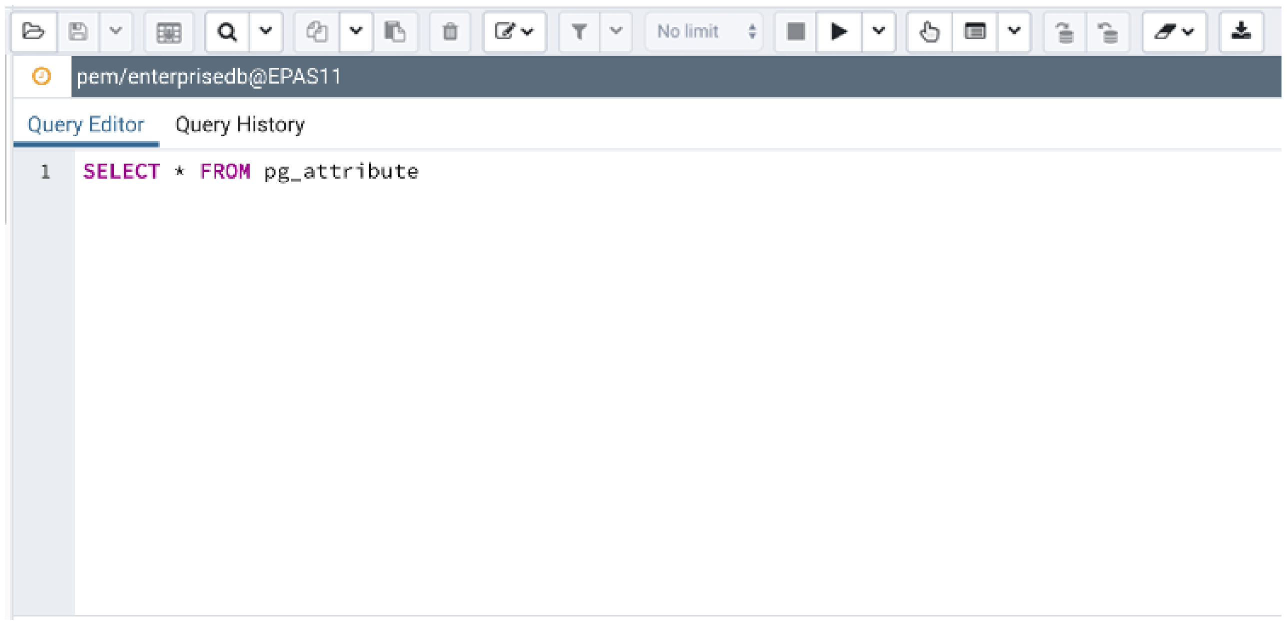
Toolbar ¶

The toolbar is described in the following subsections.

- Query Tool Toolbar
  - File Options
  - Editing Options
  - View/Edit Data Resultset Control
  - Query Execution

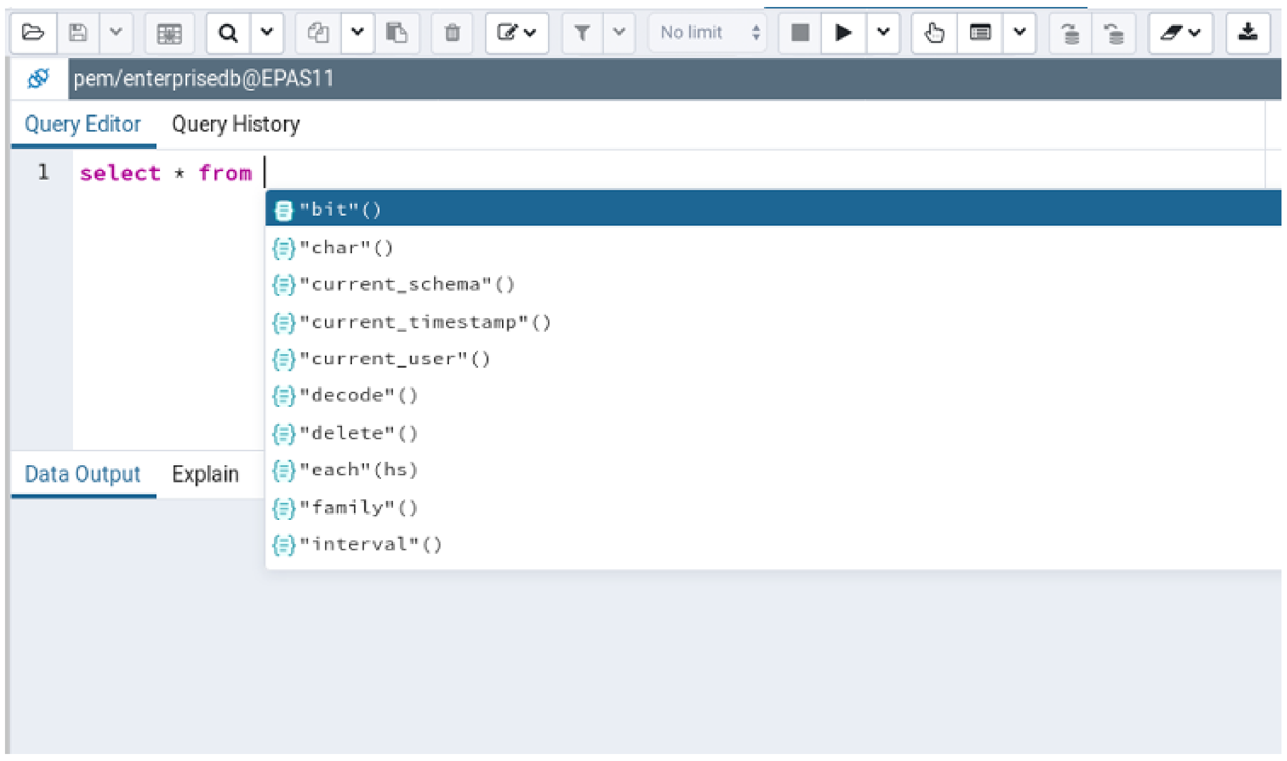
The SQL Editor Panel ¶

The *SQL editor* panel is a workspace where you can manually provide a query, copy a query from another source, or read a query from a file. The SQL editor features syntax coloring and autocompletion.



```
SELECT * FROM pg_attribute
```

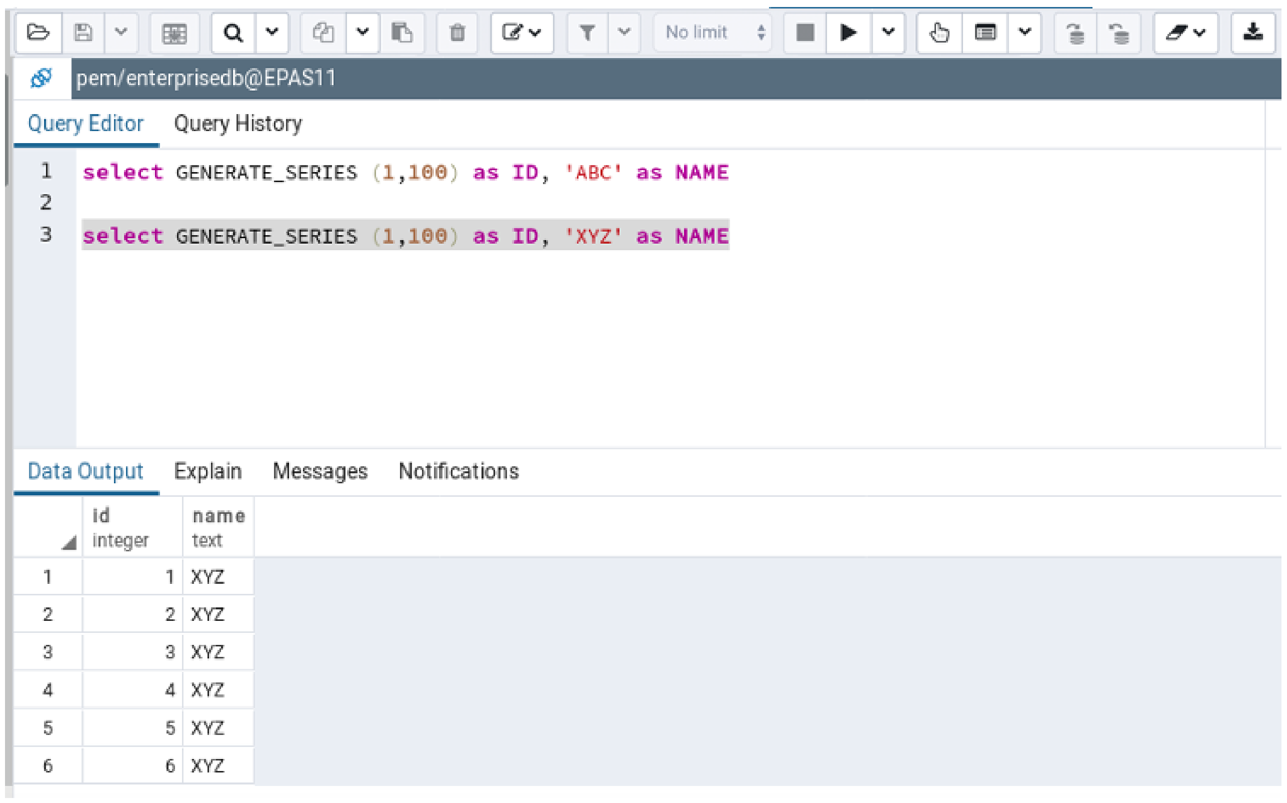
To use autocomplete, begin typing your query; when you would like the Query editor to suggest object names or commands that might be next in your query, press the Control+Space key combination. For example, type "*SELECT * FROM* " (without quotes, but with a trailing space), and then press the Control+Space key combination to select from a popup menu of autocomplete options.



```
select * from
```

```
"bit"()
"char"()
"current_schema"()
"current_timestamp"()
"current_user"()
"decode"()
"delete"()
"each"(hs)
"family"()
"interval"()
```

After entering a query, select the *Execute/Refresh* icon from the toolbar. The complete contents of the SQL editor panel will be sent to the database server for execution. To execute only a section of the code that is displayed in the SQL editor, highlight the text that you want the server to execute, and click the *Execute/Refresh* icon.

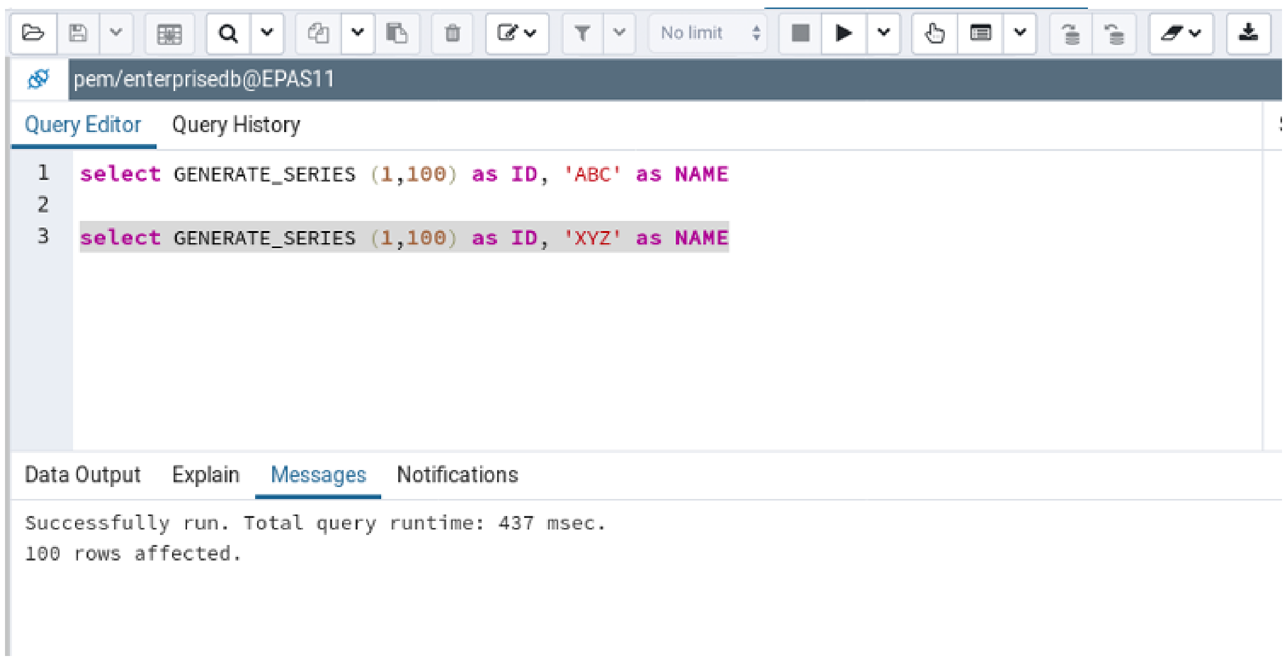


```
select GENERATE_SERIES (1,100) as ID, 'ABC' as NAME

select GENERATE_SERIES (1,100) as ID, 'XYZ' as NAME
```

	id integer	name text
1	1	XYZ
2	2	XYZ
3	3	XYZ
4	4	XYZ
5	5	XYZ
6	6	XYZ

The message returned by the server when a command executes is displayed on the *Messages* tab. If the command is successful, the *Messages* tab displays execution details.



```
select GENERATE_SERIES (1,100) as ID, 'ABC' as NAME

select GENERATE_SERIES (1,100) as ID, 'XYZ' as NAME
```

```
Successfully run. Total query runtime: 437 msec.
100 rows affected.
```

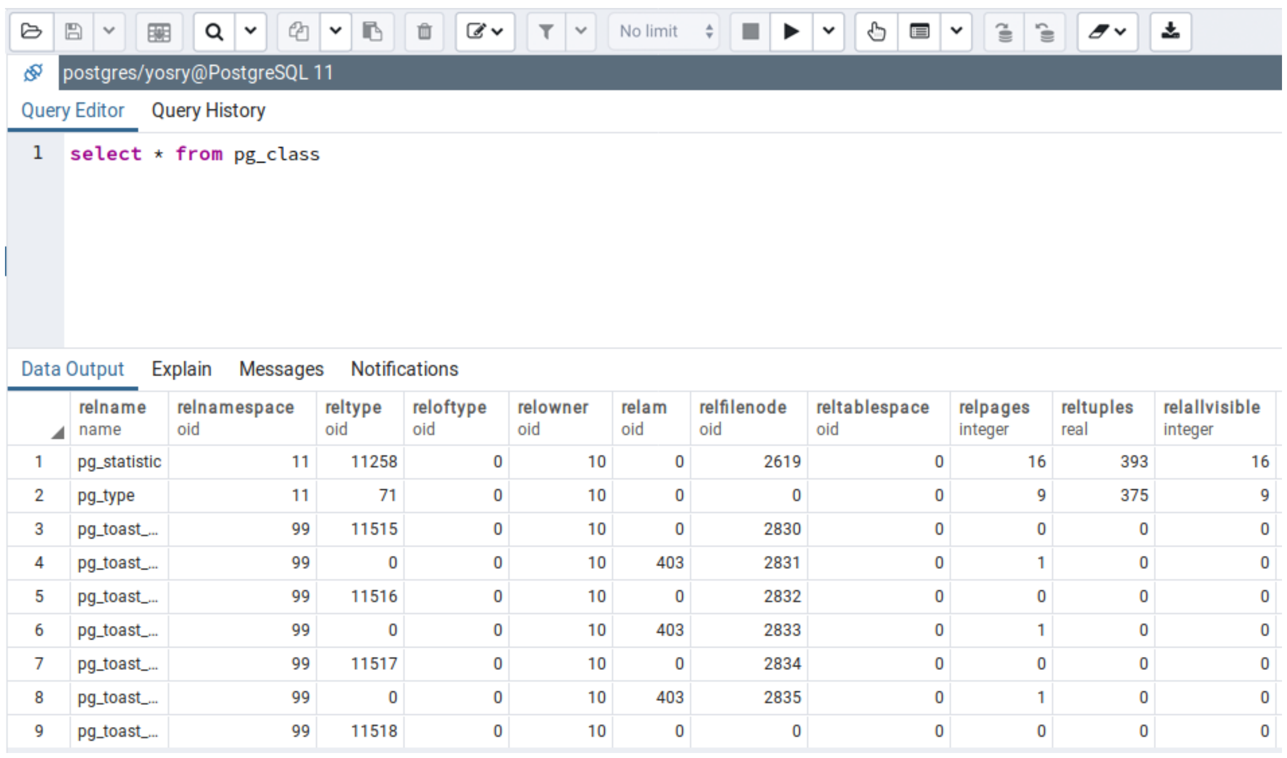
Options on the *Edit* menu offer functionality that helps with code formatting and commenting:

- The auto-indent feature will automatically indent text to the same depth as the previous line when you press the Return key.
- Block indent text by selecting two or more lines and pressing the Tab key.
- Implement or remove SQL style or toggle C style comment notation within your code.

You can also **drag and drop** certain objects from the treeview which can save time in typing long object names. Text containing the object name will be fully qualified with schema. Double quotes will be added if required. For functions and procedures, the function name along with parameter names will be pasted in the Query Tool.

The Data Output Panel ¶

The *Data Output* panel displays data and statistics generated by the most recently executed query.



```
select * from pg_class
```

	relname name	relnamespace oid	reltype oid	reloftype oid	relowner oid	relam oid	relfilenode oid	reltablespace oid	relpages integer	reltuples real	relallvisible integer
1	pg_statistic	11	11258	0	10	0	2619	0	16	393	16
2	pg_type	11	71	0	10	0	0	0	9	375	9
3	pg_toast_...	99	11515	0	10	0	2830	0	0	0	0
4	pg_toast_...	99	0	0	10	403	2831	0	1	0	0
5	pg_toast_...	99	11516	0	10	0	2832	0	0	0	0
6	pg_toast_...	99	0	0	10	403	2833	0	1	0	0
7	pg_toast_...	99	11517	0	10	0	2834	0	0	0	0
8	pg_toast_...	99	0	0	10	403	2835	0	1	0	0
9	pg_toast_...	99	11518	0	10	0	0	0	0	0	0

The *Data Output* tab displays the result set of the query in a table format. You can:

- Select and copy from the displayed result set.

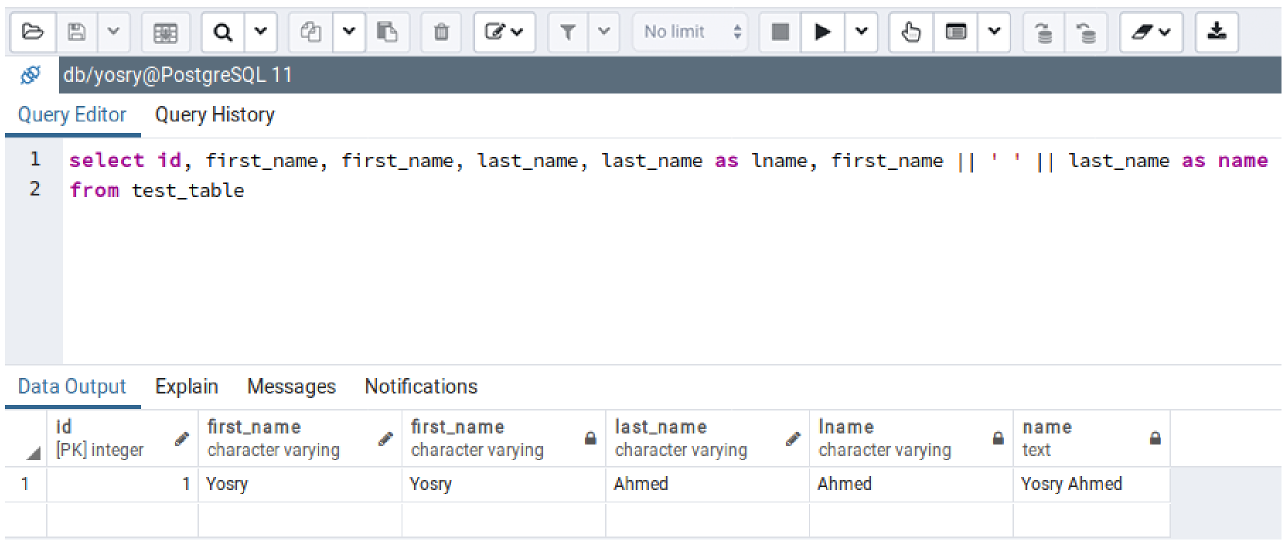
- Use the *Execute/Refresh* options to retrieve query execution information and set query execution options.
- Use the *Download as CSV* icon to download the content of the *Data Output* tab as a comma-delimited file.
- Edit the data in the result set of a SELECT query if it is updatable.

A result set is updatable if:

- All columns are either selected directly from a single table, or are not table columns at all (e.g. concatenation of 2 columns). Only columns that are selected directly from the table are editable, other columns are read-only.
- All the primary key columns or OIDs of the table are selected in the result set.

Any columns that are renamed or selected more than once are also read-only.

Editable and read-only columns are identified using pencil and lock icons (respectively) in the column headers.



```
select id, first_name, first_name, last_name, last_name as lname, first_name || ' ' || last_name as name
from test_table
```

	id [PK] integer ✎	first_name character varying ✎	first_name character varying 🔒	last_name character varying ✎	lname character varying 🔒	name text 🔒
1	1	Yosry	Yosry	Ahmed	Ahmed	Yosry Ahmed

The psycopg2 driver version should be equal to or above 2.8 for updatable query result sets to work.

An updatable result set is identical to the Data Grid in View/Edit Data mode, and can be modified in the same way.

If Auto-commit is off, the data changes are made as part of the ongoing transaction, if no transaction is ongoing a new one is initiated. The data changes are not committed to the database unless the transaction is committed.

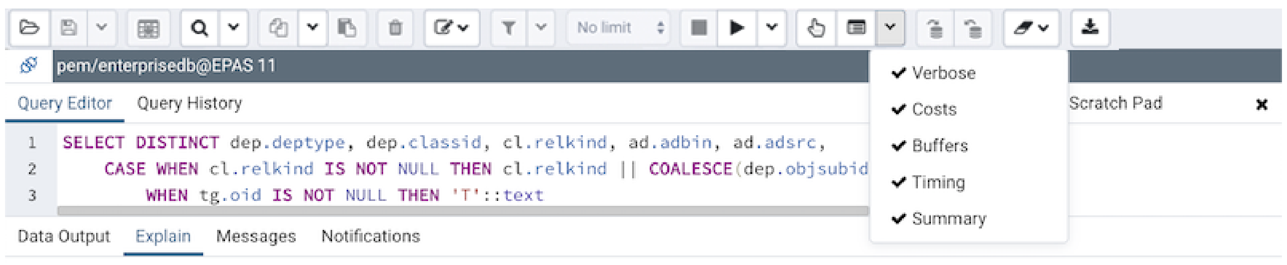
If any errors occur during saving (for example, trying to save NULL into a column with NOT NULL constraint) the data changes are rolled back to an automatically created SAVEPOINT to ensure any previously executed queries in the ongoing transaction are not rolled back.

All rowsets from previous queries or commands that are displayed in the *Data Output* panel will be discarded when you invoke another query; open another Query Tool tab to keep your previous results available.

Explain Panel ¶

To generate the *Explain* or *Explain Analyze* plan of a query, click on *Explain* or *Explain Analyze* button in the toolbar.

More options related to *Explain* and *Explain Analyze* can be selected from the drop down on the right side of *Explain Analyze* button in the toolbar.



```
SELECT DISTINCT dep.deptype, dep.classid, cl.relkind, ad.adbin, ad.adsrc,
    CASE WHEN cl.relkind IS NOT NULL THEN cl.relkind || COALESCE(dep.objsubid
        WHEN tg.oid IS NOT NULL THEN 'T'::text
```

Please note that pgAdmin generates the *Explain [Analyze]* plan in JSON format.

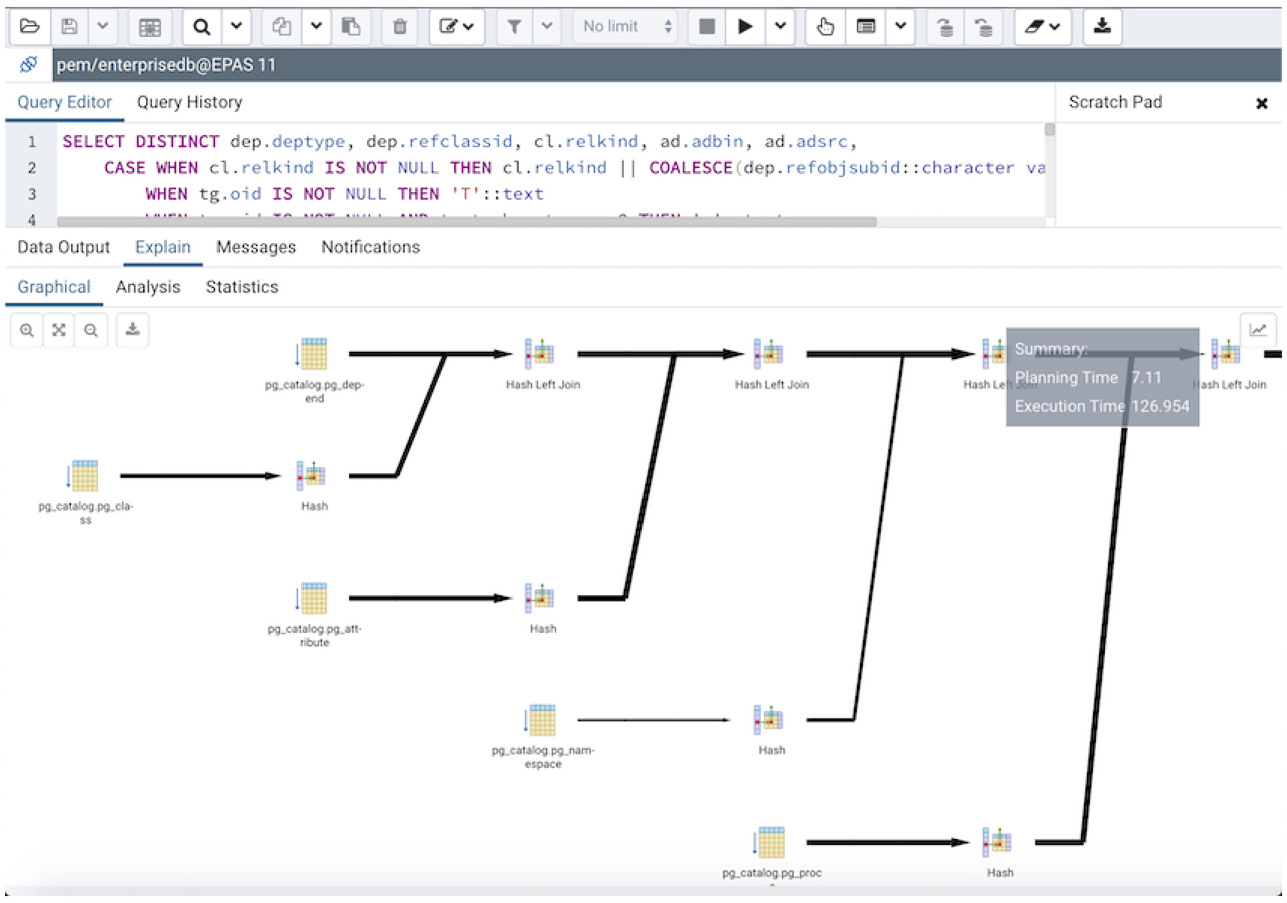
On successful generation of *Explain* plan, it will create three tabs/panels under the Explain panel.

- Graphical

Please note that *EXPLAIN VERBOSE* cannot be displayed graphically. Hover over an icon on the *Graphical* tab to review information about that item; a popup window will display information about the selected object. For information on JIT statistics, triggers and a summary, hover over the icon on top-right corner; a similar popup window will be displayed when appropriate.

Use the download button on top left corner of the *Explain* canvas to download the plan as an SVG file.

Note: Download as SVG is not supported on Internet Explorer.



```
SELECT DISTINCT dep.deptype, dep.refclassid, cl.relkind, ad.adbin, ad.adsrc,
    CASE WHEN cl.relkind IS NOT NULL THEN cl.relkind || COALESCE(dep.refobjsubid::character va
        WHEN tg.oid IS NOT NULL THEN 'T'::text
```

Summary:
Planning Time 7.11
Execution Time 126.954

Note that the query plan that accompanies the *Explain analyze* is available on the *Data Output* tab.

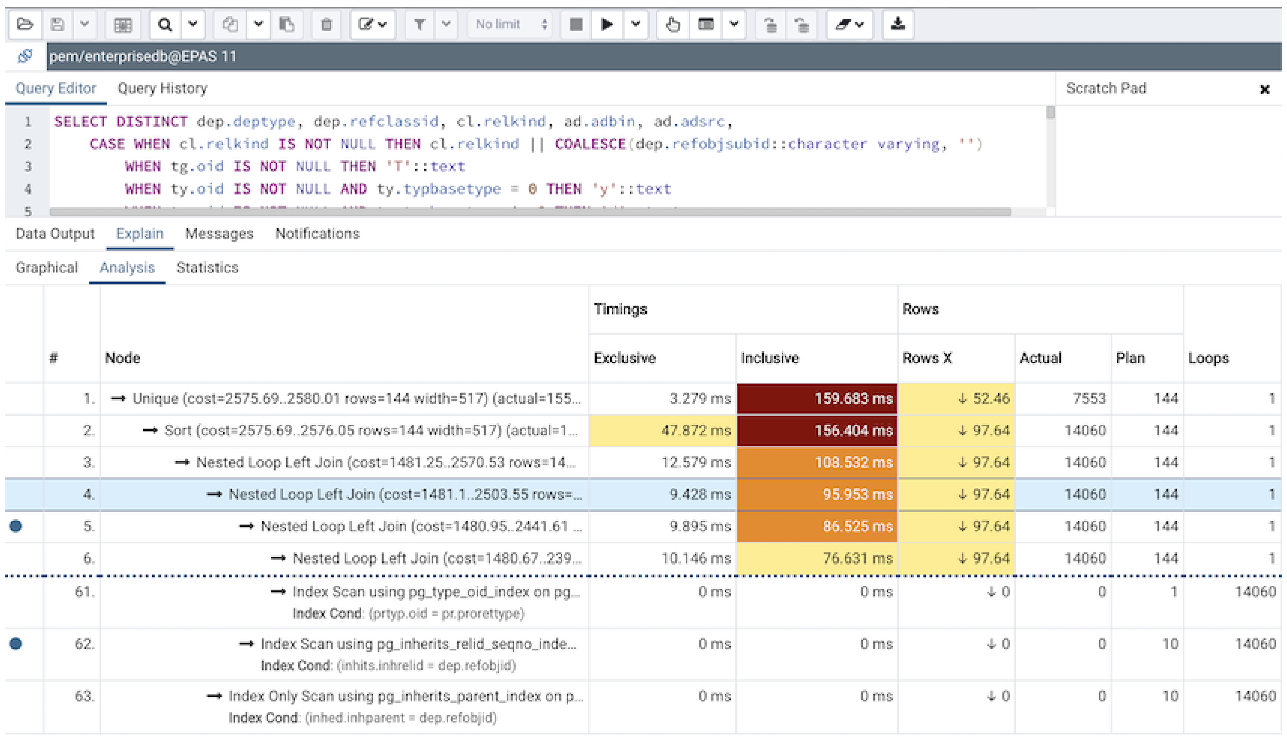
- Table

Table tab shows the plan details in table format, it generates table format similar to *explain.depsez.com*. Each row of the table represent the data for a *Explain Plan Node*. It may contain the node information, exclusive timing, inclusive timing, actual vs planned rows differences, actual rows, planned rows, loops.

background color of the exclusive, inclusive, and Rows X columns may vary based on the difference between actual vs planned.

If percentage of the exclusive/inclusive timings of the total query time is: > 90 - Red color > 50 - Orange (between red and yellow) color > 10 - Yellow color

If planner mis-estimated number of rows (actual vs planned) by 10 times - Yellow color 100 times - Orange (between Red and Yellow) color 1000 times - Red color

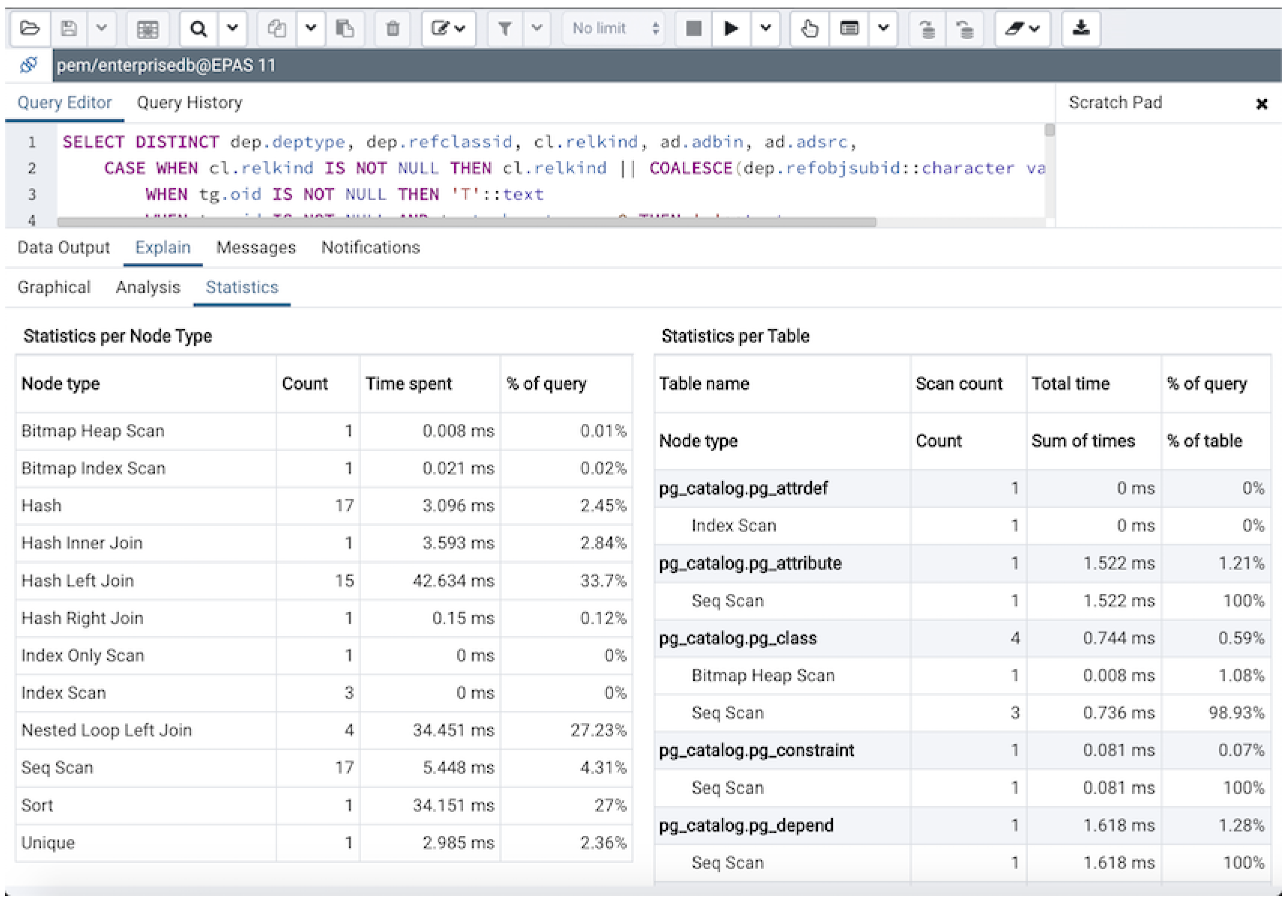


```
SELECT DISTINCT dep.deptype, dep.refclassid, cl.relkind, ad.adbin, ad.adsrc,
    CASE WHEN cl.relkind IS NOT NULL THEN cl.relkind || COALESCE(dep.refobjsubid::character varying, '')
        WHEN tg.oid IS NOT NULL THEN 'T'::text
        WHEN ty.oid IS NOT NULL AND ty.typbasetype = 0 THEN 'y'::text
```

			Timings		Rows			
	#	Node	Exclusive	Inclusive	Rows X	Actual	Plan	Loops
	1.	→ Unique (cost=2575.69..2580.01 rows=144 width=517) (actual=155...	3.279 ms	159.683 ms	↓ 52.46	7553	144	1
	2.	→ Sort (cost=2575.69..2576.05 rows=144 width=517) (actual=1...	47.872 ms	156.404 ms	↓ 97.64	14060	144	1
	3.	→ Nested Loop Left Join (cost=1481.25..2570.53 rows=14...	12.579 ms	108.532 ms	↓ 97.64	14060	144	1
	4.	→ Nested Loop Left Join (cost=1481.1..2503.55 rows=...	9.428 ms	95.953 ms	↓ 97.64	14060	144	1
●	5.	→ Nested Loop Left Join (cost=1480.95..2441.61 ...	9.895 ms	86.525 ms	↓ 97.64	14060	144	1
	6.	→ Nested Loop Left Join (cost=1480.67..239...	10.146 ms	76.631 ms	↓ 97.64	14060	144	1
	61.	→ Index Scan using pg_type_oid_index on pg... **Index Cond**: (prtyp.oid = pr.prorettype)	0 ms	0 ms	↓ 0	0	1	14060
●	62.	→ Index Scan using pg_inherits_relid_seqno_inde... **Index Cond**: (inhits.inhrelid = dep.refobjid)	0 ms	0 ms	↓ 0	0	10	14060
	63.	→ Index Only Scan using pg_inherits_parent_index on p... **Index Cond**: (inhed.inhparent = dep.refobjid)	0 ms	0 ms	↓ 0	0	10	14060

- Statistics

Statistics tab shows two tables: 1. Statistics per Plan Node Type 2. Statistics per Table



```
SELECT DISTINCT dep.deptype, dep.refclassid, cl.relkind, ad.adbin, ad.adsrc,
    CASE WHEN cl.relkind IS NOT NULL THEN cl.relkind || COALESCE(dep.refobjsubid::character va
        WHEN tg.oid IS NOT NULL THEN 'T'::text
```

Statistics per Node Type

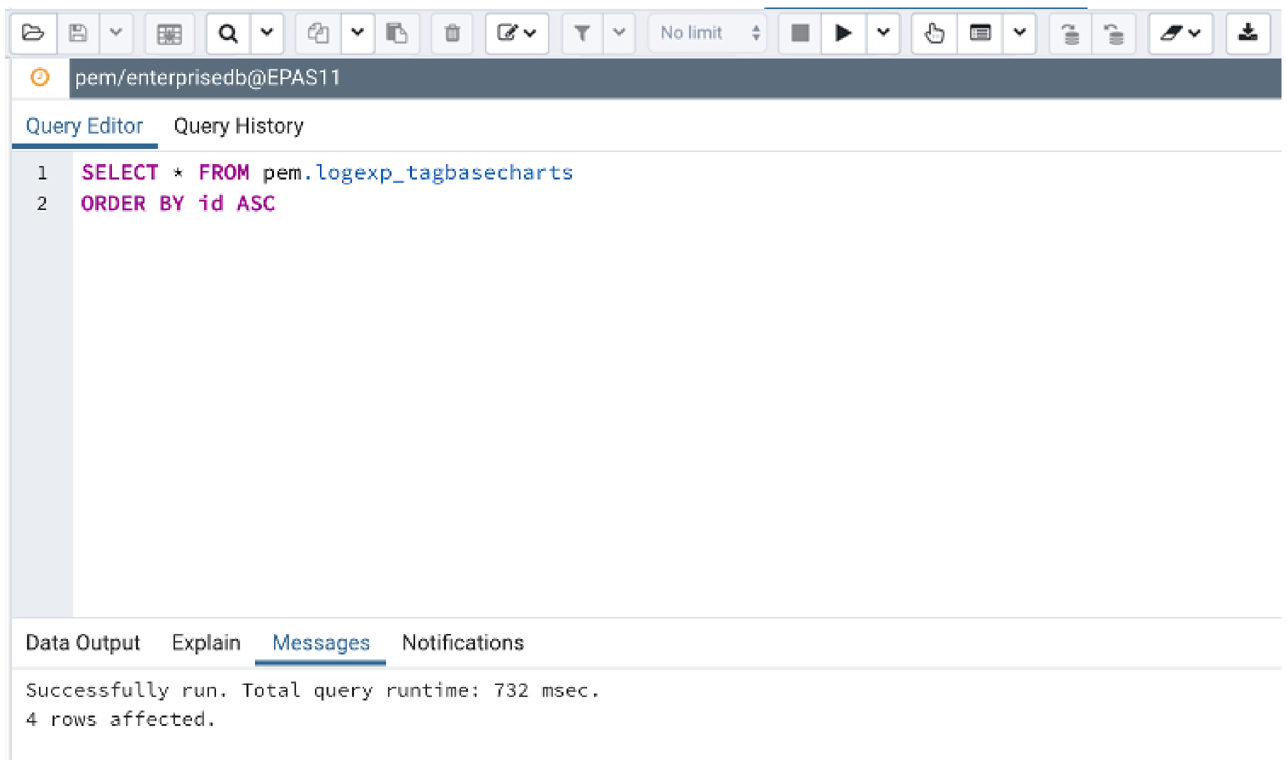
Node type	Count	Time spent	% of query
Bitmap Heap Scan	1	0.008 ms	0.01%
Bitmap Index Scan	1	0.021 ms	0.02%
Hash	17	3.096 ms	2.45%
Hash Inner Join	1	3.593 ms	2.84%
Hash Left Join	15	42.634 ms	33.7%
Hash Right Join	1	0.15 ms	0.12%
Index Only Scan	1	0 ms	0%
Index Scan	3	0 ms	0%
Nested Loop Left Join	4	34.451 ms	27.23%
Seq Scan	17	5.448 ms	4.31%
Sort	1	34.151 ms	27%
Unique	1	2.985 ms	2.36%

Statistics per Table

Table name	Scan count	Total time	% of query
Node type	Count	Sum of times	% of table
pg_catalog.pg_attrdef	1	0 ms	0%
Index Scan	1	0 ms	0%
pg_catalog.pg_attribute	1	1.522 ms	1.21%
Seq Scan	1	1.522 ms	100%
pg_catalog.pg_class	4	0.744 ms	0.59%
Bitmap Heap Scan	1	0.008 ms	1.08%
Seq Scan	3	0.736 ms	98.93%
pg_catalog.pg_constraint	1	0.081 ms	0.07%
Seq Scan	1	0.081 ms	100%
pg_catalog.pg_depend	1	1.618 ms	1.28%
Seq Scan	1	1.618 ms	100%

Messages Panel ¶

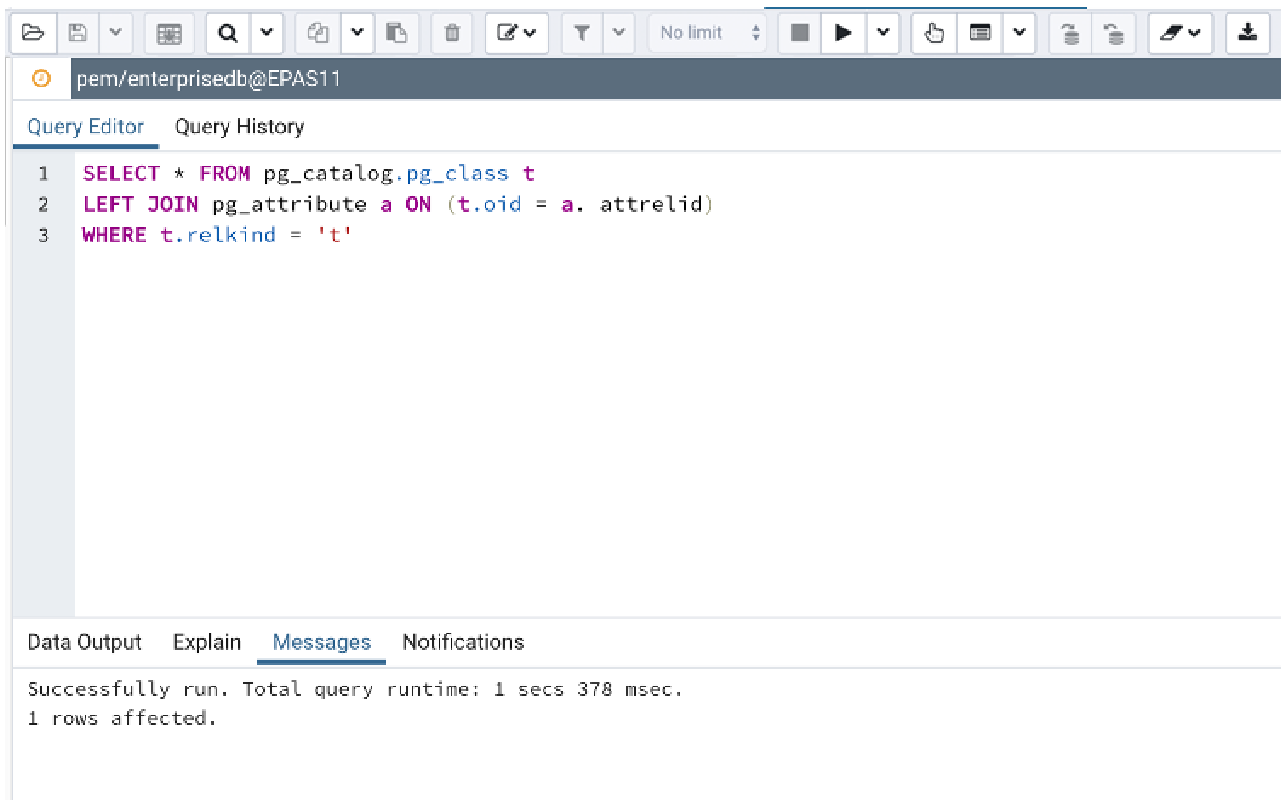
Use the *Messages* tab to view information about the most recently executed query:



```
SELECT * FROM pem.logexp_tagbasecharts
ORDER BY id ASC
```

```
Successfully run. Total query runtime: 732 msec.
4 rows affected.
```

If the server returns an error, the error message will be displayed on the *Messages* tab, and the syntax that caused the error will be underlined in the SQL editor. If a query succeeds, the *Messages* tab displays how long the query took to complete and how many rows were retrieved:

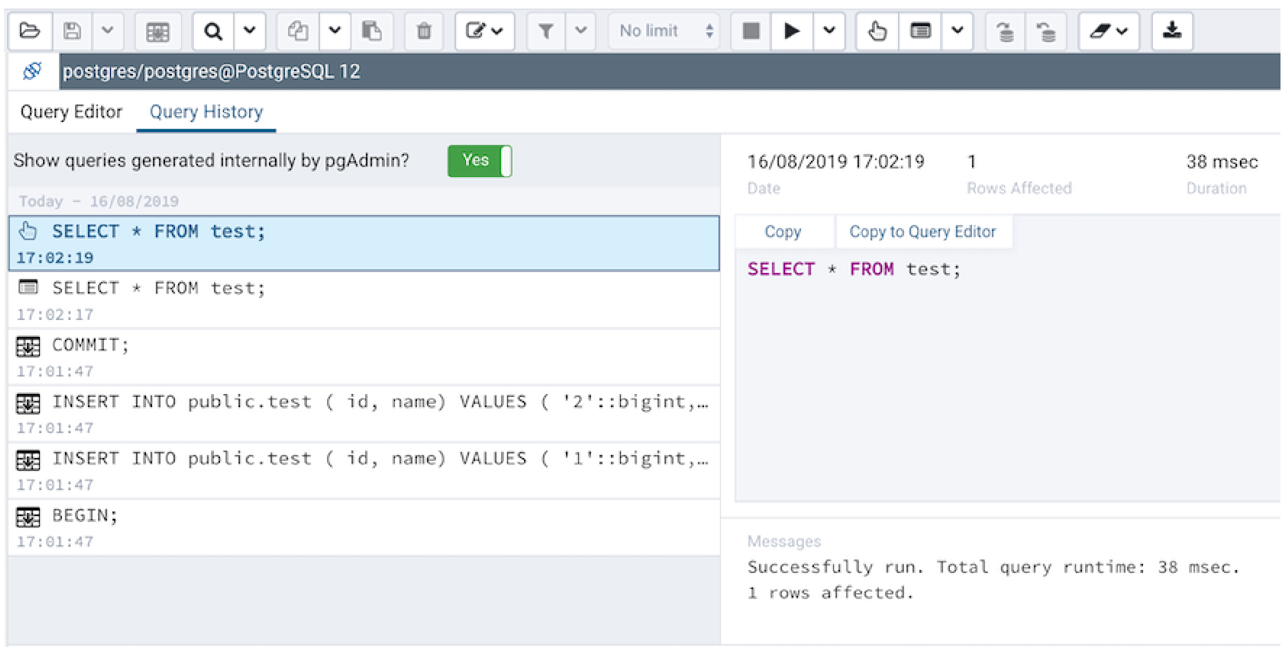


```
SELECT * FROM pg_catalog.pg_class t
LEFT JOIN pg_attribute a ON (t.oid = a. attrelid)
WHERE t.relkind = 't'
```

```
Successfully run. Total query runtime: 1 secs 378 msec.
1 rows affected.
```

Query History Panel ¶

Use the *Query History* tab to review activity for the current session:



```
SELECT * FROM test;
17:02:19
SELECT * FROM test;
17:02:17
COMMIT;
17:01:47
INSERT INTO public.test ( id, name) VALUES ( '2'::bigint,…
17:01:47
INSERT INTO public.test ( id, name) VALUES ( '1'::bigint,…
17:01:47
BEGIN;
17:01:47
```

16/08/2019 17:02:19 1 38 msec

```
SELECT * FROM test;
```

```
Successfully run. Total query runtime: 38 msec.
1 rows affected.
```

The Query History tab displays information about recent commands:

- The date and time that a query was invoked.
- The text of the query.
- The number of rows returned by the query.

- The amount of time it took the server to process the query and return a result set.
- Messages returned by the server (not noted on the *Messages* tab).
- The source of the query (indicated by icons corresponding to the toolbar).

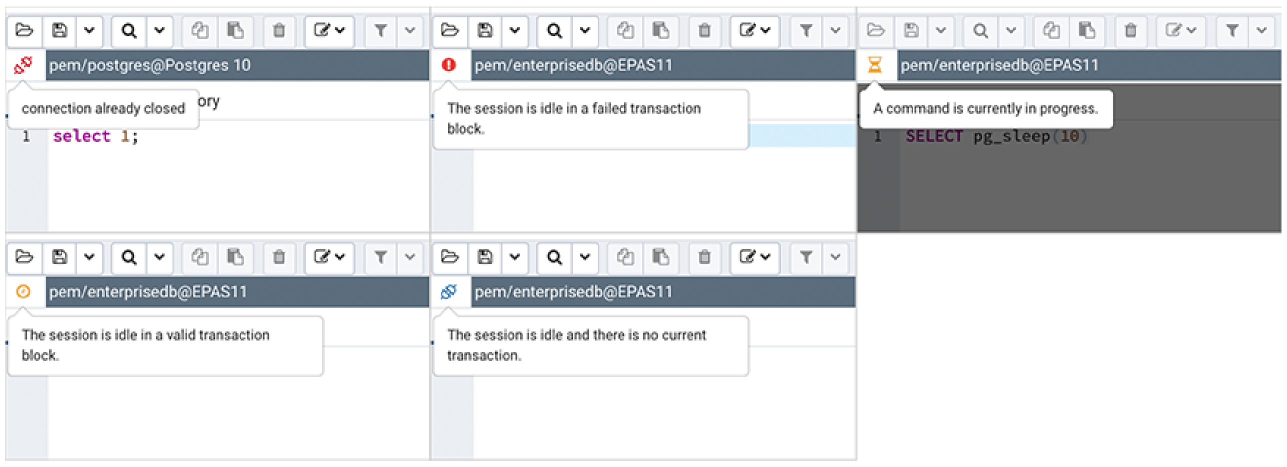
You can show or hide the queries generated internally by pgAdmin (during ‘View/Edit Data’ or ‘Save Data’ operations).

To erase the content of the *Query History* tab, select *Clear history* from the *Clear* drop-down menu.

Query History is maintained across sessions for each database on a per-user basis when running in Query Tool mode. In View/Edit Data mode, history is not retained. By default, the last 20 queries are stored for each database. This can be adjusted in *config_local.py* by overriding the *MAX_QUERY_HIST_STORED* value. See the Deployment section for more information.

Connection Status ¶

Use the *Connection status* feature to view the current connection and transaction status by clicking on the status icon in the Query Tool:



pem/postgres@Postgres 10

connection already closed

```
select 1;
```

pem/enterprisedb@EPAS11

The session is idle in a failed transaction block.

pem/enterprisedb@EPAS11

A command is currently in progress.

```
SELECT pg_sleep(10)
```

pem/enterprisedb@EPAS11

The session is idle in a valid transaction block.

pem/enterprisedb@EPAS11

The session is idle and there is no current transaction.