

Data Analysis with SQL

This lesson introduces structured query language, often referred to as SQL. You can use SQL to pull and manipulate data from a relational database.

Structured Query Language

SQL provides a way to query, or talk to, the database. You might use SQL to answer specific questions about the data, like "What was the average number of transactions per store last week?" or "How many districts in Region A have sold more than 1,000 units of our new product?" And with SQL, you can do more than simply access information; there are also other types of queries available. You can use SQL to update, delete, and create tables by modifying rows and columns.

Every programming language has its own syntax, which is a set of rules that define the language. In upcoming lessons, you'll learn these syntax rules. You'll see how to write specific code to instruct the database management system to do what you need it to do.

SQL is used in many relational database management systems, such as PostgreSQL, MySQL, and SQL Server. This course uses PostgreSQL to practice using the SQL language. However, note that different management systems can have slight differences in syntax. For example, the function ISNULL exists in SQL Server but not in PostgreSQL. The equivalent function in PostgreSQL is COALESCE. These differences occur because of the way that each database management system is set up.

Query

The query is an important element of SQL. Query means question, and as you learned in the previous module, a query asks the relational database management system (RDBMS) to retrieve data based on specific criteria. But a query doesn't perform any actions or manipulations on the data itself.

Statement

The statement is one complete action that can be executed. The difference between a statement and a query is that a query retrieves data based on certain criteria, but a statement isn't limited to retrieving data. A statement refers to any action that is performed. A statement may manipulate data, add database connections, or perform diagnostics.

The two images below illustrate the difference between a query and a statement in SQL.

The first example below asks the RDBMS to return data, so it's a query and a statement.

```
SELECT *  
FROM payment;
```

Statement
and
Query

But the following example performs an action on the data (creates a table) but doesn't return any data—so it's just a statement.

```
CREATE TABLE inventory (  
    product_A text,  
    product_B text,  
    product_C text,  
);
```

Statement

Clause

Like sentences in natural language, SQL statements are composed of clauses. The clause is a component within a statement that states the action being performed. A clause is introduced by its keyword. A keyword is a function name, such as DELETE or SELECT. So, a SELECT clause is the clause that follows the SELECT keyword.

The diagram below shows the main syntax components in a simple query that counts the amount from a table called payment.

Keywords

Clause

```
SELECT COUNT (amount)
```

```
FROM payment;
```

Statement

Syntax rules

There are a few SQL syntax rules that you need to follow when writing queries and statements:

- Start SQL statements with keywords. SQL statements always begin with keywords. When you type keywords in the pgAdmin screen, those keywords are automatically bolded so that they stand out. This helps you find them if you need to change them or review your code.
- End SQL statements with a semicolon. Every SQL statement ends in a semicolon ;. PostgreSQL won't give you an error if you forget to include a semicolon, but it is considered good practice to include one when coding in SQL. A semicolon will also help to visually identify where your statement ends when you're reviewing your code. If you have multiple statements in a query, a semicolon marks where one statement ends and another begins.
- Don't worry about capitalization. Some languages are very strict about when to use capital versus lowercase letters. Luckily, SQL doesn't have such rules; it isn't case sensitive, so typing SELECT is the same as typing select. You may choose to use capital letters, lowercase letters, or a mixture of both. But typically, people tend to use capital letters when typing keywords so that they stand out from the rest of the text. This can make it easier to find certain clauses and expressions when reviewing or fixing your code.
- Start comments with --. Comments within the code always start with two hyphens --. Leaving instructive comments about what the code is doing—and how it's doing it—is one of the best habits to pick up when learning how to code. In the sample query below, the comment takes up the first two lines. All lines of comments begin with --, even if the second line is part of the same comment. This comment explains to the user what the query is doing.

Query Editor	Query History
<pre>1 --The following is a query to pull actor_id and actor's last_name 2 --from the table actor 3 SELECT actor_id, last_name 4 FROM actor;</pre>	

Comments are used for many reasons. Sometimes a query can become quite long with multiple statements, so comments help explain to the user what each section of code is doing. Comments can be used to separate sections of code, making it easier to find your place while reviewing or changing code. Comments are also used to instruct the user to do something, such as update a year or date within the code.

SQL data types

Every database, whether it's a tabular or relational database, has its own definitions of data types that it uses. You learned earlier that Excel's data types include strings, numbers, and dates. SQL has its own data types, and some are similar to the ones that you saw in Excel.

Numeric data types

Numeric data is data that has a numerical value. There are three main numeric data types that you need to know for this course: integer, decimal, and float.

The integer data type is a positive or negative whole number, such as 1.

The decimal data type is a decimal number, such as 1.5.

The float data type is a floating-point approximation. This is similar to a decimal data type but with less accuracy, meaning that it may round values if the length of the decimal is long or has repeating digits, such as 1.3333333333.

Here's an example of each:

Integer	Decimal	Float
1	1.5	1.52348735873495734952783 469234920374023423472234 82364893243842398423874

String data types

String data is data that contains text. String data types include character, varchar, and text.

- The character data type has a fixed length. If the character is shorter than the defined length, SQL pads it with blank spaces to reach the defined length. Say a column titled Gender contains the value options Male and Female, and the defined character length is 6. The character data type adds two extra spaces to Male so that both values achieve a length of 6. So, character data types are used for columns where values are expected to be the same length.
- The varchar data type can have varying lengths. If the Gender column contains Male and Female, then Male has a length of 4 and Female has a length of 6. Varchar data types cannot exceed the defined length.
- The text data type is just like the varchar data type, except that it has no length limit.

Character	Varchar	Text
Defined length Values are expected to be the same length	Defined length Values are varying lengths, but values cannot exceed the defined length	No defined length Values are varying lengths
Male__ (length = 6 with added spaces) Female (length = 6)	Male (length = 4) Female (length = 6)	Male (length = 4) Female (length = 6) Prefer not to answer (length = 20)

Time and date data types

There are three main time and date data types: date, time, and timestamp.

- The date data type is a date in calendar format, without any time associated with it.
- The time data type is a time, without any date associated with it.
- The timestamp data type is a combination of the two, showing date and time. But the timestamp data type doesn't have time zone information, so it won't update automatically if you change your database server to a new time zone.

Date	Time	Timestamp
2017-07-23	12:10:11	2017-07-23, 12:10:11

Boolean data type

Boolean data has one of two possible values, or it can be unknown (NULL). Specifically, a boolean data type can have the values TRUE, FALSE, or NULL. Or it may have the values 0, 1, or NULL. For example, if a yes-no question is asked on a survey, answers are recorded in the data as 0 for no, 1 for yes, and NULL for choose not to answer. Another example is a data table with patient information that has a column called smoker. The values for this column could be 0 or FALSE if they aren't a smoker, 1 or TRUE if they are a smoker, or NULL if it's unknown.

pgAdmin browser components

This lesson shows where key features of the pgAdmin browser are located. Once you're familiar with these, you'll be able to navigate pgAdmin easily when you begin writing code.

Now that you have PostgreSQL and pgAdmin installed on your machine, open pgAdmin and follow any prompts that pop up. Now you can see the pgAdmin browser window and can begin to explore the browser components.

Tree control pane

The first thing that you may notice is the tree control pane (also called the Browser pane) on the left of the window. This pane allows you to see an overview of the servers that you're connected to, as well as further details in their branches. If you click the drop-down arrow next to Servers, you'll see all the databases that you have access to on the server.

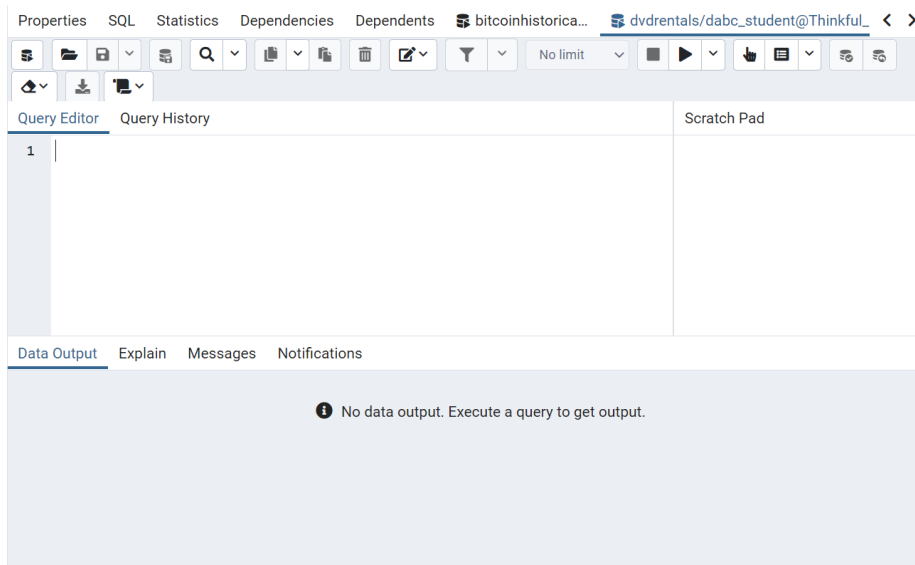
Click one of the databases—`dvdrentals`, for example—and you'll see a lot of options to learn more about the database. You can also learn more about the tables that make up this database. Select Schemas under `dvdrentals`. Go to `Public`, and then `Tables`. You'll see 24 tables listed there.

You can use pgAdmin to learn more about the columns for these tables. Right-click the `film` table and select `Properties`. Then, head to the `Columns` tab. Every column has a name and a specific data type. The `Columns` tab lists every column in the table, each column's data type, and any length or precision constraints. This is a quick way to see what a table contains.

Query tool

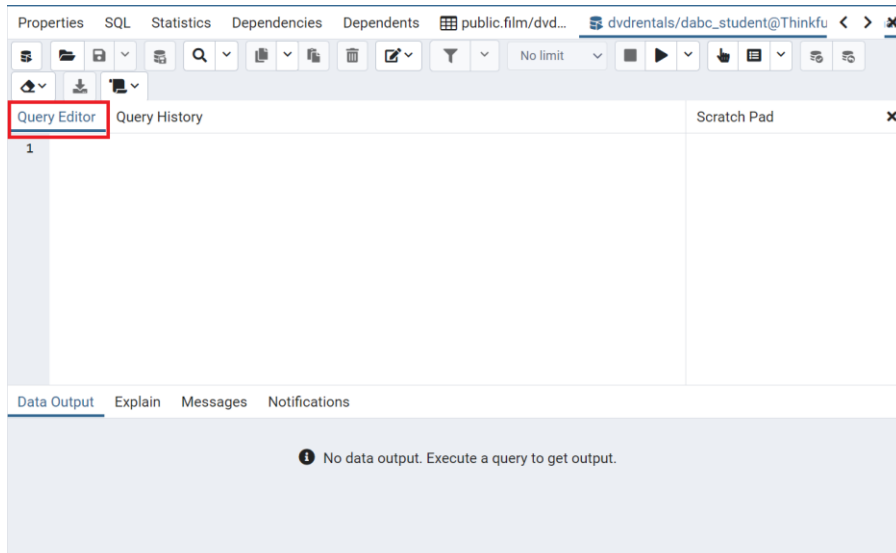
Now, right-click the `film` table again and select `Query Tool` from the menu. This reveals a window at the top of your screen, called the Query Tool. The Query Tool is where you can write and execute SQL queries, cancel queries, and look through past queries that you've created, among other things.

Note that some buttons might be in different locations depending on what version of pgAdmin you're using.



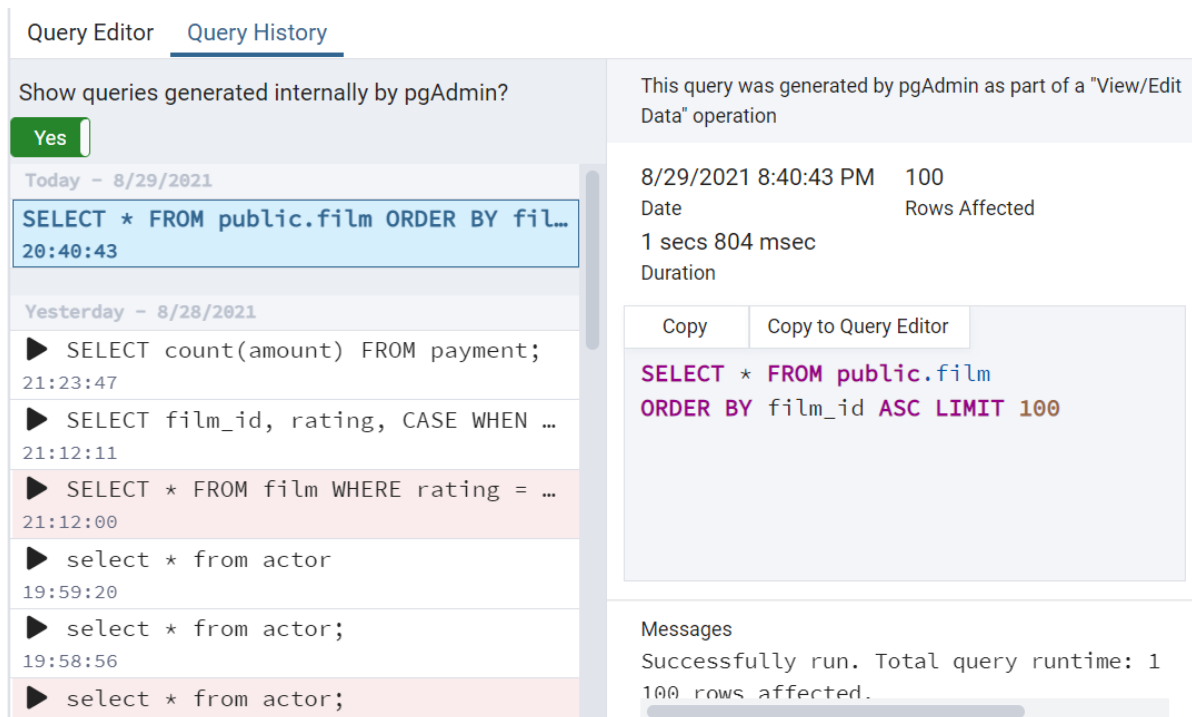
Query Editor

The Query Editor tab is located within the Query Tool. It's the place where SQL code is written and executed, and it's where you'll spend most of your time for the remainder of this course!



Query History

The Query History tab within the Query Tool shows your recent queries. It also indicates if they were run successfully or not. This makes it easy to execute a query again without having to rewrite or remember it. By default, your last 20 queries are shown in this tab.



Executing a query

Once you've written a query and want to run it, or execute it, click the Execute/Refresh ► button in the top toolbar.

The screenshot shows the top toolbar of the Query Editor. The 'Execute/Refresh' button, represented by a play icon, is highlighted with a red rectangle. Below the toolbar, the 'Query Editor' tab is active, showing a SQL query:

```
1 select *
2 from actor
```

The 'Data Output' tab is selected at the bottom, displaying a message: "No data output. Execute a query to get output."


Data Output

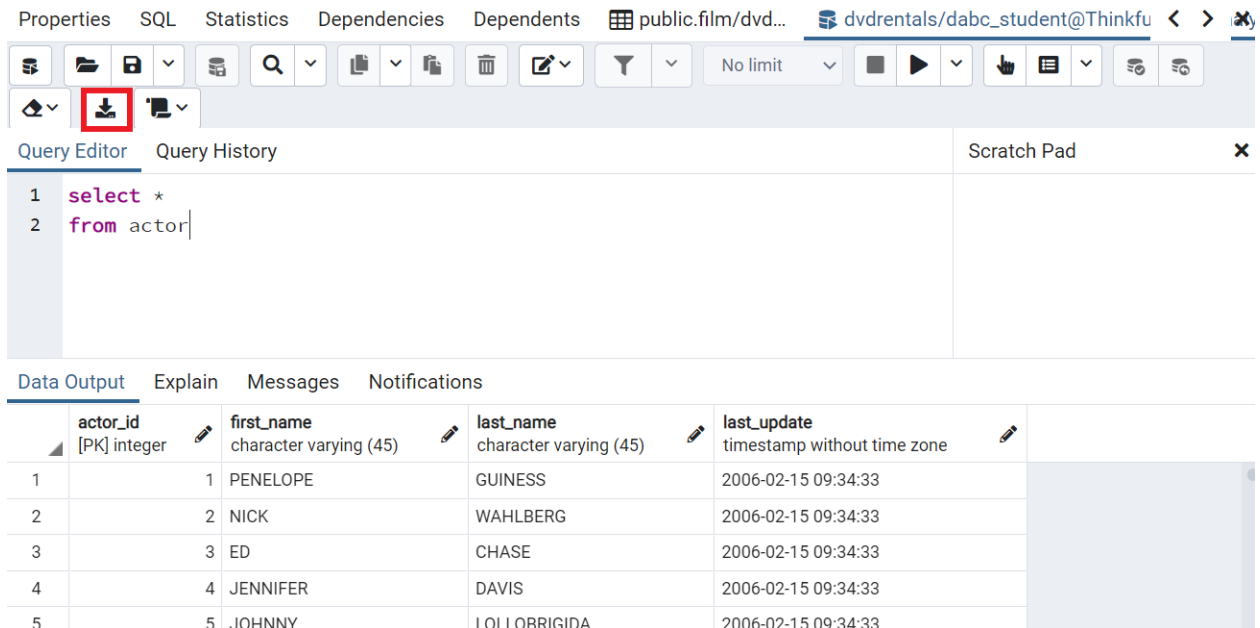
The Data Output window is at the bottom of the Query Editor. When you execute a query, Data Output shows the retrieved data immediately. This gives you an immediate view of what your query pulled, making it easy to fix or change your code if necessary.

The screenshot shows the 'Data Output' window at the bottom of the Query Editor. The query results are displayed in a table with the following columns: actor_id, first_name, last_name, and last_update. The results are highlighted with a red rectangle.

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2006-02-15 09:34:33
2	NICK	WAHLBERG	2006-02-15 09:34:33
3	ED	CHASE	2006-02-15 09:34:33
4	JENNIFER	DAVIS	2006-02-15 09:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 09:34:33
6	BETTE	NICHOLSON	2006-02-15 09:34:33
7	GRACE	MOSTEL	2006-02-15 09:34:33
8	MATTHEW	JOHANSSON	2006-02-15 09:34:33
9	JOE	SWANK	2006-02-15 09:34:33

Downloading output

If you ever want to download the output of your query, click Save results to file . Then you'll be able to name your file and download it to a location of your choice.



The screenshot shows the pgAdmin interface with the 'Save results to file' button (floppy disk icon) highlighted in the toolbar. The Query Editor contains the following SQL query:

```
1 select *
2 from actor
```

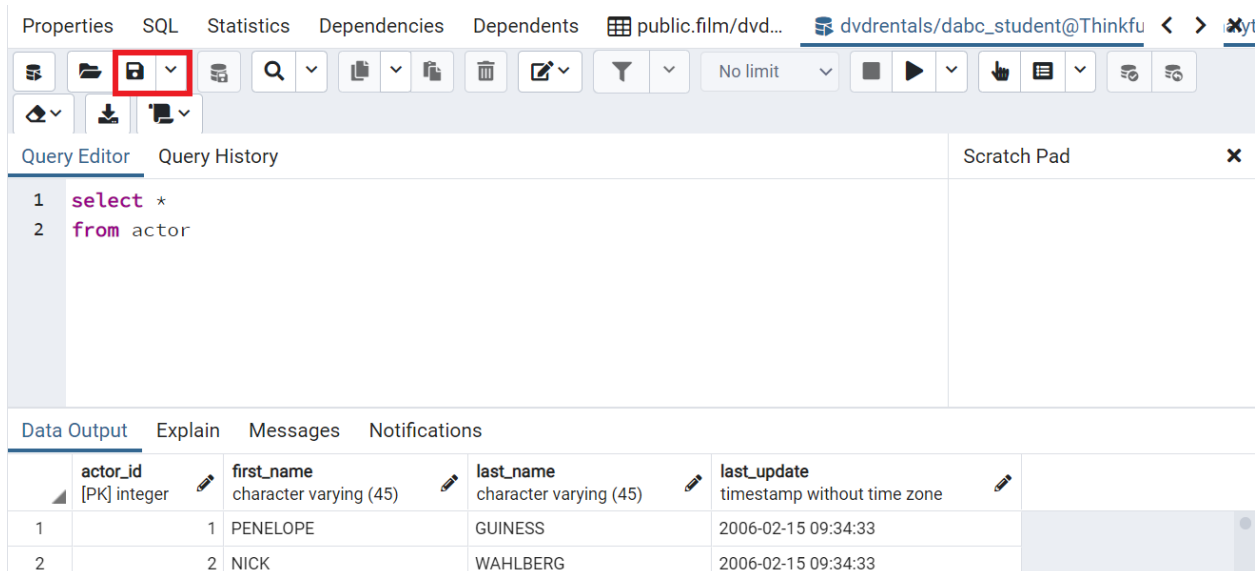
The Data Output tab is active, displaying the following table:

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	PENELOPE	GUINNESS	2006-02-15 09:34:33
2	2	NICK	WAHLBERG	2006-02-15 09:34:33
3	3	ED	CHASE	2006-02-15 09:34:33
4	4	JENNIFER	DAVIS	2006-02-15 09:34:33
5	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 09:34:33

Depending on the version of pgAdmin you're using, the Save results to File button might be right above your Data Output.

Opening and saving files

To save your work, click Save File . Then you'll be able to name and save your file.




The screenshot shows the pgAdmin interface with the 'Save File' button (floppy disk icon) highlighted in the toolbar. The Query Editor contains the following SQL query:

```
1 select *
2 from actor
```

The Data Output tab is active, displaying the following table:

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	PENELOPE	GUINNESS	2006-02-15 09:34:33
2	2	NICK	WAHLBERG	2006-02-15 09:34:33

To open an existing file in pgAdmin, such as an old query that you were working on, click Open File  and locate the file.

The screenshot shows a database management tool with a top toolbar containing icons for Properties, SQL, Statistics, Dependencies, and Dependents. The main window is divided into three panes: Query Editor, Query History, and Scratch Pad. The Query Editor contains the following SQL query:

```
1 select *
2 from actor
```

Below the Query Editor is a tabbed interface with 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with the following data:

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	PENELOPE	GUINNESS	2006-02-15 09:34:33
2	2	NICK	WAHLBERG	2006-02-15 09:34:33

Exporting SQL data

This lesson shows you how to export data and save query results.

In this course, you've learned numerous SQL commands to execute different types of queries. As you've seen, SQL is a very powerful tool that lets you quickly pull data or manipulate data from very large datasets. Once a query is run successfully, you may want to save your query results to your computer so that you can do further analysis or refer to the results again later. You may also want to save the original data table so that you can work with it in a familiar format like Excel.

This lesson shows you how to export data and save your query results as comma-separated value (CSV) files. Though you can also save data in other formats, CSV files are used widely because they are simple and can be read by almost all types of software.

Exporting a data table

To export a data table, first find the data table in the tree control pane and click it. For practice, use the data table called owners in the dog_saloon database. Right-click this data table and then click Import/Export.

In the Options tab of this window, you have the option to import or export data. For this exercise, leave it on the Export setting. Next, create a filename for this data table, such as dog_owners. The Format is csv, and the Header setting should be Yes so that you keep column names. Finally, select the comma in the Delimiter option; this ensures that a comma is used to separate values in the exported data.

Import/Export data - table 'owners'

Options
Columns

Import/Export
Export

File Info

Filename
dog_owners

Format
csv

Encoding
Select an item...

Miscellaneous

OID
No

Header
Yes

Delimiter
,

Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This must be a single one-byte character. This option is not allowed when using binary format.

Quote
"

Cancel
OK

The Columns tab in this window lets you choose which columns you want to export. It also lets you choose if you want to treat NULL values in any special way.

Import/Export data - table 'owners'

Options
Columns

Columns to export
owner_id pet_id size

An optional list of columns to be copied. If no column list is specified, all columns of the table will be copied.

NULL Strings

Specifies the string that represents a null value. The default is \N (backslash-N) in text format, and an unquoted empty string in CSV format. You might prefer an empty string even in text format for cases where you don't want to distinguish nulls from empty strings. This option is not allowed when using binary format.

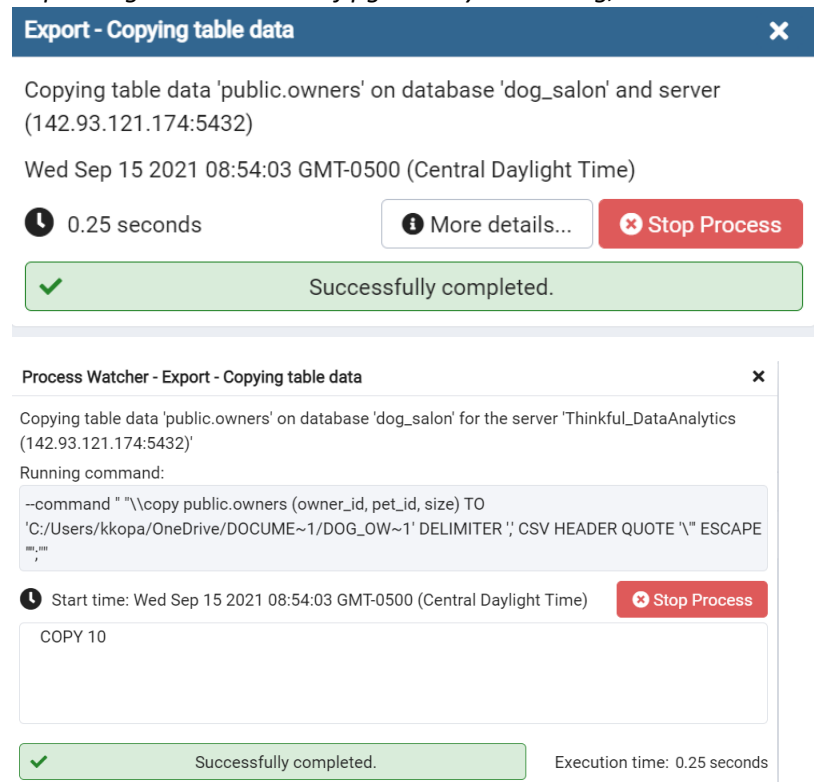
Not null columns

Do not match the specified column values against the null string. In the default case where the null string is empty, this means that empty values will be read as zero-length strings rather than nulls, even when they are not quoted. This option is allowed only in import, and only when using CSV format.

Cancel
OK

There's no need to change these default settings, so click OK to export this data. You'll be notified when the export is successfully accomplished. It will also be specified where this data is located.

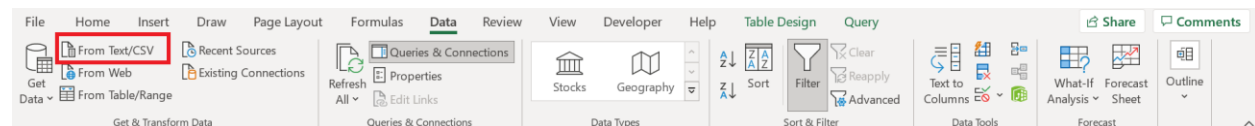
Depending on the version of pgAdmin you're using, these screens might be organized differently.



Opening exported data in Excel

Now that you've successfully exported a data table to a CSV file, try opening it in Excel. To do this, open a new workbook in Excel and head over to the ribbon's Data tab. In the Get & Transform Data section, click From Text/CSV.

The layout of your ribbon may vary depending on your computer's operating system and the version of Excel that you're using. If you aren't seeing this option, try going to File > Import > CSV file.



Find and select your exported data file and click Import. A preview window will pop up to show you how your data looks. Select Load, and you will see the owners table from the dog_salon database in your new Excel workbook!

	A	B	C	
1	owner_id ▾	pet_id ▾	size ▾	
2	11	67	small	
3	13	86	medium	
4	52	43	small	
5	65	23	large	
6	87	79	medium	
7	99	32	large	
8	34	13	large	
9	13	66	small	
10	52	34	small	
11	13	87	large	

Exporting query results

Finally, you may want to save your query results rather than the original data table. To do this, first execute your query. Then, click the Save results to file button in the toolbar.

The screenshot shows the pgAdmin interface. At the top is a toolbar with various icons. The 'Save results to file' icon (a floppy disk) is highlighted with a red box. Below the toolbar, the 'Query Editor' tab is active, displaying a SQL query:

```
1 SELECT *
2 FROM owners
3 ORDER BY size DESC;
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query in a table format:

	owner_id integer	pet_id integer	size text
1	13	66	small
2	52	34	small
3	11	67	small
4	52	43	small
5	13	86	medium
6	87	79	medium
7	13	87	large
8	65	23	large
9	99	32	large
10	34	13	large

Depending on the version of pgAdmin you're using, the Save results to File button might be right above your Data Output.

You'll be able to name your file and select the location where you want it to be saved. By default, it saves as a CSV file. If you want to open the query results in Excel, follow the steps outlined above to open the exported data in Excel.