Extract PostgreSql table data into CSV file

Introduction

It can be done in multiple ways

- **COPY** SQL clause
- **copy** metacommand of psql
- DBeaver **Task** for data export
- pgAdmin **Import/Export Data** Dialog
- Exporting data from an RDS for PostgreSQL DB instance to Amazon S3

This document discusses how the **\copy** metacommand of psql can be used to export data into CSV file.

NOTE	**Prerequisite:** "psql" compatible with postgresql server must be available on client machine (installation on *Unix platforms is easier when compared with Windows platform)

General syntax of the **copy** meta command is

```
\copy { table [ ( column_list ) ] | ( query ) } { from | to } { 'filename' | program 'command' | stdin | stdout | pstdin | pstdout } [ [ with ] ( option [, ⋯] ) ]
```

where ***option*** can be one of:

- **FORMAT** *format_name*
- **OIDS** [*boolean*]
- **FREEZE** [*boolean*]
- **DELIMITER** *'delimiter_character'*
- **NULL** *'null_string'*
- **HEADER** [*boolean*]
- **QUOTE** *'quote_character'*
- **ESCAPE** *'escape_character'*
- **FORCE_QUOTE** { (*column_name* [, ...]) | * }
- **FORCE_NOT_NULL** (*column_name* [, ...])
- **FORCE_NULL** (*column_name* [, ...])

- **ENCODING** *'encoding_name'*

The command performs a frontend (client) copy. This is an operation that runs an SQL COPY command, but instead of the server reading or writing the specified file, psql reads or writes the file and routes the data between the server and the local file system. This means that file accessibility and privileges are those of the local user, not the server, and no SQL superuser privileges are required.

For `\copy ⋯ from stdin`, data rows are read from the same source that issued the command, continuing until `\.` is read or the stream reaches `EOF`. This option is useful for populating tables in-line within a SQL script file. For `\copy ⋯ to stdout`, output is sent to the same place as psql command output, and the COPY count command status is not printed (since it might be confused with a data row). To read/write psql's standard input or output regardless of the current command source or \o option, write from pstdin or to pstdout.

How to extract data from one table

- **Step 1:** Connect to database using **psql** command

```
psql -h <database_server_host_name_or_ip> -p <database_server_port_number> -d <database_name> -U <user_name>
```

For example:

[rkumar1@centos-minimal~]$ psql -U ravi.kumar -p 5432 -h myokardia-dev-app-clinical.ceefoxvmoyon.us-east-1.rds.amazonaws.com -d myokardia-dev-app-clinical

```
psql -h example-db.server.com -p 9999 -d example_database -U example_user_name
```

- **Step 2:** Prepare \copy command \copy (select column1, column2, ..., columnN from schema.table) to 'output_filename' csv header; \copy (select * from schema.table) to 'output_filename' csv header;

- **Step 3:** Run the command on psql prompt.

- **Step 4:** Collect output csv files

How to extract data from multiple tables

- **Step 1:** Connect to database using psql command

- **Step 2:** Prepare **\copy** commands
  - Multiple ways
    - manually prepare \copy command for each table and save all commands in a file
    - use a sql query similar to the one mentioned below and save the result in a file

```
WITH db AS (
    SELECT UNNEST(ARRAY[
        'myokardia-dev-app-clinical'
        -- more db names can be mentioned here
        ])::TEXT AS "name"
)
, aggregated_table_columns AS (
    SELECT table_schema, table_name, string_agg(quote_ident(column_name),',
') AS columns_  FROM information_schema."columns" c
        JOIN db ON c.table_catalog = db."name"
        AND (table_schema LIKE 'myk%' OR table_schema LIKE 'ppd%')
        AND (column_name NOT IN
('__dq_flag','__generation_time','__hash','comprehend_update_time'))
        GROUP BY 1,2
)
, select_queries as(
    SELECT 'SELECT ' || columns_ || ' FROM ' || quote_ident(table_schema) ||
'.' || quote_ident(table_name) AS query, table_schema, table_name FROM
aggregated_table_columns
)
, psql_copy_meta_command AS (
    SELECT  '\copy (' || query || ') to ''' || table_schema || '/' ||
table_name || '.csv' || ''' csv header ;' AS command
        FROM select_queries
)
SELECT command FROM psql_copy_meta_command
```

- **Step 3:** Run all commands present in the file by instructing psql to take commands from the file generated in the previous step.

```
\i 'path_to_commands_file'
```

- **Step 4:** Collect output csv files from current directory. It should be done after closing the db connection.

- **Step 5:** use **mkdir** and **mv** linux commands to organize the extracted csv files as requried. The commands are out of scope for this document. Please refer to their respective help for details.