PostgreSQL Cheat Sheet

The **PostgreSQL cheat sheet** provides you with the common PostgreSQL commands and statements that enable you to work with PostgreSQL quickly and effectively.

Download PostgreSQL cheat sheet

We provide you with a 3-page PostgreSQL cheat sheet in PDF format. You can download and print it out for a quick reference to the most commonly used statements in PostgreSQL:

Download PostgreSQL Cheat Sheet (https://sp.postgresqltutorial.com/wp-content/uploads/2018/03/PostgreSQL-Cheat-Sheet.pdf)

PostgreSQL commands

Access the PostgreSQL server from **psql** with a specific user:

```
psql -U [username];
```

For example, the following command uses the `postgres` user to access the PostgreSQL database server:

```
psql -U postgres
```

Connect to a specific database:

```
\c database_name;
```

For example, the following command connects to the `dvdrental` database:

```
\c dvdrental;
You are now connected to database "dvdrental" as user "postgres".
```

To quit the psql:

```
\q
```

List all databases (https://www.postgresqltutorial.com/postgresql-show-databases/) in the PostgreSQL database server

```
\l
```

List all schemas:

```
\dn
```

List all stored procedures (https://www.postgresqltutorial.com/postgresql-stored-procedures/) and functions:

```
\df
```

List all views (https://www.postgresqltutorial.com/postgresql-views/):

```
\dv
```

Lists all tables (https://www.postgresqltutorial.com/postgresql-show-tables/) in a current database.

```
\dt
```

Or to get more information on tables in the current database:

```
\dt+
```

Get detailed information on a table.

```
\d+ table_name
```

Show a stored procedure (https://www.postgresqltutorial.com/postgresql-stored-procedures/) or function code:

```
\df+ function_name
```

Show query output in the pretty-format:

```
\x
```

List all users:

```
\du
```

Create a new role (https://www.postgresqltutorial.com/postgresql-roles/) :

```
CREATE ROLE role_name;
```

Create a new role with a `username` and `password` :

```
CREATE ROLE username NOINHERIT LOGIN PASSWORD password;
```

Change role for the current session to the `new_role` :

```
SET ROLE new_role;
```

Allow `role_1` to set its role as `role_2`:

```
GRANT role_2 TO role_1;
```

Managing databases

Create a new database (https://www.postgresqltutorial.com/postgresql-create-database/):

```
CREATE DATABASE [IF NOT EXISTS] db_name;
```

Delete a database permanently (https://www.postgresqltutorial.com/postgresql-drop-database/):

```
DROP DATABASE [IF EXISTS] db_name;
```

Managing tables

Create a new table (https://www.postgresqltutorial.com/postgresql-create-table/) or a temporary table (https://www.postgresqltutorial.com/postgresql-temporary-table/)

```
CREATE [TEMP] TABLE [IF NOT EXISTS] table_name(
   pk SERIAL PRIMARY KEY,
   c1 type(size) NOT NULL,
   c2 type(size) NULL,
   ...
);
```

Add a new column (https://www.postgresqltutorial.com/postgresql-add-column/) to a table:

```
ALTER TABLE table_name ADD COLUMN new_column_name TYPE;
```

Drop a column (https://www.postgresqltutorial.com/postgresql-drop-column/) in a table:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Rename a column (https://www.postgresqltutorial.com/postgresql-rename-column/):

```
ALTER TABLE table_name RENAME column_name TO new_column_name;
```

Set or remove a default value for a column:

```
ALTER TABLE table_name ALTER COLUMN [SET DEFAULT value | DROP DEFAULT]
```

Add a primary key (https://www.postgresqltutorial.com/postgresql-primary-key/) to a table.

```
ALTER TABLE table_name ADD PRIMARY KEY (column,...);
```

Remove the primary key from a table.

```
ALTER TABLE table_name
DROP CONSTRAINT primary_key_constraint_name;
```

Rename a table (https://www.postgresqltutorial.com/postgresql-rename-table/).

```
ALTER TABLE table_name RENAME TO new_table_name;
```

Drop a table (https://www.postgresqltutorial.com/postgresql-drop-table/) and its dependent objects:

```
DROP TABLE [IF EXISTS] table_name CASCADE;
```

Managing views

Create a view (https://www.postgresqltutorial.com/managing-postgresql-views/):

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

Create a recursive view (https://www.postgresqltutorial.com/postgresql-recursive-view/) :

```
CREATE RECURSIVE VIEW view_name(column_list) AS
SELECT column_list;
```

Create a materialized view (https://www.postgresqltutorial.com/postgresql-materialized-views/) :

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

Refresh a materialized view:

```
REFRESH MATERIALIZED VIEW CONCURRENTLY view_name;
```

Drop a view:

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

Drop a materialized view:

```
DROP MATERIALIZED VIEW view_name;
```

Rename a view:

```
ALTER VIEW view_name RENAME TO new_name;
```

Managing indexes

Creating an index with the specified name on a table

```
CREATE [UNIQUE] INDEX index_name
ON table (column,...)
```

Removing a specified index from a table

```
DROP INDEX index_name;
```

Querying data from tables

Query all data from a table:

```
SELECT * FROM table_name;
```

Query data from specified columns of all rows in a table:

```
SELECT column_list
FROM table;
```

Query data and select only unique rows:

```
SELECT DISTINCT (column)
FROM table;
```

Query data from a table with a filter:

```
SELECT *
FROM table
WHERE condition;
```

Assign an alias (https://www.postgresqltutorial.com/postgresql-alias/) to a column in the result set:

```
SELECT column_1 AS new_column_1, ...
FROM table;
```

Query data using the LIKE (https://www.postgresqltutorial.com/postgresql-like/) operator:

```
SELECT * FROM table_name
WHERE column LIKE '%value%'
```

Query data using the BETWEEN (https://www.postgresqltutorial.com/postgresql-between/) operator:

```
SELECT * FROM table_name
WHERE column BETWEEN low AND high;
```

Query data using the IN (https://www.postgresqltutorial.com/postgresql-in/) operator:

```
SELECT * FROM table_name
WHERE column IN (value1, value2,...);
```

Constrain the returned rows with the LIMIT (https://www.postgresqltutorial.com/postgresql-limit/) clause:

```
SELECT * FROM table_name
LIMIT limit OFFSET offset
ORDER BY column_name;
```

Query data from multiple using the inner join (https://www.postgresqltutorial.com/postgresql-inner-join/), left join (https://www.postgresqltutorial.com/postgresql-left-join/), full outer join (https://www.postgresqltutorial.com/postgresql-full-outer-join/), cross join (https://www.postgresqltutorial.com/postgresql-cross-join/) and natural join (https://www.postgresqltutorial.com/postgresql-natural-join/):

```
SELECT *
FROM table1
INNER JOIN table2 ON conditions
```

```
SELECT *
FROM table1
LEFT JOIN table2 ON conditions
```

```
SELECT *
FROM table1
FULL OUTER JOIN table2 ON conditions
```

```
SELECT *
FROM table1
CROSS JOIN table2;
```

```
SELECT *
FROM table1
NATURAL JOIN table2;
```

Return the number of rows of a table.

```
SELECT COUNT (*)
FROM table_name;
```

Sort rows in ascending or descending order:

```
SELECT select_list
FROM table
ORDER BY column ASC [DESC], column2 ASC [DESC],...;
```

Group rows using GROUP BY (https://www.postgresqltutorial.com/postgresql-group-by/) clause.

```
SELECT *
FROM table
GROUP BY column_1, column_2, ...;
```

Filter groups using the HAVING (https://www.postgresqltutorial.com/postgresql-having/) clause.

```
SELECT *
FROM table
GROUP BY column_1
HAVING condition;
```

Set operations

Combine the result set of two or more queries with UNION (https://www.postgresqltutorial.com/postgresql-union/) operator:

```
SELECT * FROM table1
UNION
SELECT * FROM table2;
```

Minus a result set using EXCEPT (https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-except/) operator:

```
SELECT * FROM table1
EXCEPT
SELECT * FROM table2;
```

Get intersection of the result sets of two queries:

```
SELECT * FROM table1
INTERSECT
SELECT * FROM table2;
```

Modifying data

Insert a new row into a table (https://www.postgresqltutorial.com/postgresql-insert/):

```
INSERT INTO table(column1,column2,...)
VALUES(value_1,value_2,...);
```

Insert multiple rows into a table:

```
INSERT INTO table_name(column1,column2,...)
VALUES(value_1,value_2,...),
      (value_1,value_2,...),
      (value_1,value_2,...)...
```

Update (https://www.postgresqltutorial.com/postgresql-update/) data for all rows:

```
UPDATE table_name
SET column_1 = value_1,
    ...;
```

Update data for a set of rows specified by a condition in the `WHERE` clause.

```
UPDATE table
SET column_1 = value_1,
    ...
WHERE condition;
```

Delete all rows (https://www.postgresqltutorial.com/postgresql-delete/) of a table:

```
DELETE FROM table_name;
```

Delete specific rows based on a condition:

```
DELETE FROM table_name
WHERE condition;
```

Performance

Show the query plan for a query:

```
EXPLAIN query;
```

Show and execute the query plan for a query:

```
EXPLAIN ANALYZE query;
```

Collect statistics:

```
ANALYZE table_name;
```