PostgreSQL

- PostgreSQL commands

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

```
dwd
```

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 in the PostgreSQL database server

\l
List all schemas:

\dn

List all stored procedures and functions:

\df
List all views:

\dv

Lists all 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 or function code:

\df+ function_name
Show query output in the pretty-format:

\x
List all users:

\du
Create a new role:

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:

CREATE DATABASE [IF NOT EXISTS] db_name;

Delete a database permanently:

DROP DATABASE [IF EXISTS] db_name;

Managing tables
Create a new table or a 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 to a table:

ALTER TABLE table_name ADD COLUMN new_column_name TYPE;

Drop a column in a table:

ALTER TABLE table_name DROP COLUMN column_name;

Rename a 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 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.

ALTER TABLE table_name RENAME TO new_table_name;

Drop a table and its dependent objects:

DROPTABLE [IF EXISTS] table_name CASCADE;

Managing views
Create a view:

CREATE OR REPLACE view_name AS
query;

Create a recursive view:

CREATE RECURSIVE VIEW view_name(column_list) AS
SELECT column_list;

Create a materialized view:

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 to a column in the result set:

SELECT column_1 AS new_column_1, ...
FROM table;

Query data using the LIKE operator:

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

Query data using the BETWEEN operator:

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

Query data using the IN operator:

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

Constrain the returned rows with the LIMIT clause:

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

Query data from multiple using the inner join, left join, full outer join, cross join and 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 clause.

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

Filter groups using the HAVING clause.

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

Set operations
Combine the result set of two or more queries with UNION operator:

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

Minus a result set using 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:

```
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 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 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;
```