This chapter provides a list of the PostgreSQL SQL commands, followed by the precise syntax rules for each of these commands. This set of commands is taken from the psql command-line tool. Now that you have Postgres installed, open the psql as:

**Program Files > PostgreSQL 9.2 > SQL Shell** `psql`

Using `psql`, you can generate the complete list of commands by using the `\help` command. For the syntax of a specific command, use the following command:

```
postgres-# \help <command_name>
```

### The SQL Statement

An SQL statement is comprised of tokens where each token can represent either a keyword, identifier, quoted identifier, constant, or special character symbol. The table below uses a simple SELECT statement to illustrate a basic, but complete, SQL statement and its components.

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Description</th>
<th>SELECT id, name FROM states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword</td>
<td>Command</td>
<td>Keyword Identifier</td>
</tr>
<tr>
<td>Identifiers</td>
<td>Id and name columns</td>
<td>Clause Table name</td>
</tr>
</tbody>
</table>

### PostgreSQL SQL commands

#### ABORT

Abort the current transaction.

```
ABORT [ WORK | TRANSACTION ]
```

#### ALTER AGGREGATE

Change the definition of an aggregate function.

```
ALTER AGGREGATE name ( type ) RENAME TO new_name
ALTER AGGREGATE name ( type ) OWNER TO new_owner
```

#### ALTER CONVERSION

Change the definition of a conversion.

```
ALTER CONVERSION name RENAME TO new_name
ALTER CONVERSION name OWNER TO new_owner
```

#### ALTER DATABASE

Change a database specific parameter.

```
ALTER DATABASE name SET parameter { TO | = } { value | DEFAULT }
ALTER DATABASE name RESET parameter
ALTER DATABASE name RENAME TO new_name
ALTER DATABASE name OWNER TO new_owner
```
### ALTER DOMAIN

Change the definition of a domain specific parameter.

```
ALTER DOMAIN name { SET DEFAULT expression | DROP DEFAULT }
ALTER DOMAIN name { SET | DROP } NOT NULL
ALTER DOMAIN name ADD domain_constraint
ALTER DOMAIN name DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
ALTER DOMAIN name OWNER TO new_owner
```

### ALTER FUNCTION

Change the definition of a function.

```
ALTER FUNCTION name ( [ type [ , ... ] ] ) RENAME TO new_name
ALTER FUNCTION name ( [ type [ , ... ] ] ) OWNER TO new_owner
```

### ALTER GROUP

Change a user group.

```
ALTER GROUP groupname ADD USER username [, ... ]
ALTER GROUP groupname DROP USER username [, ... ]
ALTER GROUP groupname RENAME TO new_name
```

### ALTER INDEX

Change the definition of an index.

```
ALTER INDEX name OWNER TO new_owner
ALTER INDEX name SET TABLESPACE indexspace_name
ALTER INDEX name RENAME TO new_name
```

### ALTER LANGUAGE

Change the definition of a procedural language.

```
ALTER LANGUAGE name RENAME TO new_name
```

### ALTER OPERATOR

Change the definition of an operator.

```
ALTER OPERATOR name ( { lefttype | NONE }, { righttype | NONE } ) OWNER TO new_owner
```

### ALTER OPERATOR CLASS

Change the definition of an operator class.

```
ALTER OPERATOR CLASS name USING index_method RENAME TO new_name
ALTER OPERATOR CLASS name USING index_method OWNER TO new_owner
```

### ALTER SCHEMA

Change the definition of a schema.

```
ALTER SCHEMA name RENAME TO new_name
ALTER SCHEMA name OWNER TO new_owner
```
### ALTER SEQUENCE

Change the definition of a sequence generator.

```sql
ALTER SEQUENCE name [ INCREMENT [ BY ] increment ]
[ MINVALUE minvalue | NO MINVALUE ]
[ MAXVALUE maxvalue | NO MAXVALUE ]
[ RESTART [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
```

### ALTER TABLE

Change the definition of a table.

```sql
ALTER TABLE [ ONLY ] name [ * ]
action [ , ... ]
ALTER TABLE [ ONLY ] name [ * ]
RENAME [ COLUMN ] column TO new_column
ALTER TABLE name
RENAME TO new_name
```

Where `action` is one of the following lines:

```sql
ADD [ COLUMN ] column_type [ column_constraint [ ... ] ]
DROP [ COLUMN ] column [ RESTRICT ] [ CASCADE ]
ALTER [ COLUMN ] column TYPE type [ USING expression ]
ALTER [ COLUMN ] column SET DEFAULT expression
ALTER [ COLUMN ] column DROP DEFAULT
ALTER [ COLUMN ] column { SET | DROP } NOT NULL
ALTER [ COLUMN ] column SET STATISTICS integer
ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
ADD table_constraint
DROP CONSTRAINT constraint_name [ RESTRICT ] [ CASCADE ]
CLUSTER ON index_name
SET WITHOUT CLUSTER
SET WITHOUT OIDS
OWNER TO new_owner
SET TABLESPACE tablespace_name
```

### ALTER TABLESPACE

Change the definition of a tablespace.

```sql
ALTER TABLESPACE name RENAME TO new_name
ALTER TABLESPACE name OWNER TO new_owner
```

### ALTER TRIGGER

Change the definition of a trigger.

```sql
ALTER TRIGGER name ON table RENAME TO new_name
```

### ALTER TYPE

Change the definition of a type.

```sql
ALTER TYPE name OWNER TO new_owner
```

### ALTER USER

Change a database user account.
ALTER USER name [ [ WITH ] option [ ... ] ]
ALTER USER name RENAME TO new_name
ALTER USER name SET parameter { TO | = } { value | DEFAULT }
ALTER USER name RESET parameter

Where option can be:

[ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| VALID UNTIL 'abstime'

ANALYZE

Collect statistics about a database.

ANALYZE [ VERBOSE ] [ table [ (column [, ... ] ) ] ]

BEGIN

Start a transaction block.

BEGIN [ WORK | TRANSACTION ] [ transaction_mode [, ... ] ]

Where transaction_mode is one of:

ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
| READ UNCOMMITTED }
READ WRITE | READ ONLY

CHECKPOINT

Force a transaction log checkpoint.

CHECKPOINT

CLOSE

Close a cursor.

CLOSE name

CLUSTER

Cluster a table according to an index.

CLUSTER index_name ON table_name
CLUSTER table_name
CLUSTER

COMMENT

Define or change the comment of an object.

COMMENT ON
{ TABLE object_name | COLUMN table_name.column_name |
AGGREGATE agg_name (agg_type) | CAST (source_type AS target_type) |

...
COMMIT

Commit the current transaction.

COMMIT [ WORK | TRANSACTION ]

COPY

Copy data between a file and a table.

COPY table_name [ ( column [, ...] ) ]
FROM { 'filename' | STDIN }
[ [ WITH ]
[ BINARY ]
[ OIDS ]
[ DELIMITER [ AS ] 'delimiter' ]
[ NULL [ AS ] 'null string' ]
[ CSV [ QUOTE [ AS ] 'quote' ]
[ ESCAPE [ AS ] 'escape' ]
[ FORCE NOT NULL column [, ...] ]
COPY table_name [ ( column [, ...] ) ]
TO { 'filename' | STDOUT }
[ [ WITH ]
[ BINARY ]
[ OIDS ]
[ DELIMITER [ AS ] 'delimiter' ]
[ NULL [ AS ] 'null string' ]
[ CSV [ QUOTE [ AS ] 'quote' ]
[ ESCAPE [ AS ] 'escape' ]
[ FORCE QUOTE column [, ...] ]

CREATE AGGREGATE

Define a new aggregate function.

CREATE AGGREGATE name ( 
BASETYPE = input_data_type,
SFUNC = sfunc,
STYPE = state_data_type
[ , FINALFUNC = ffunc ]
[ , INITCOND = initial_condition ]
)

CREATE CAST

Define a new cast.
CREATE CAST (source_type AS target_type)
WITH FUNCTION func_name (arg_types)
[ AS ASSIGNMENT | AS IMPLICIT ]
CREATE CAST (source_type AS target_type)
WITHOUT FUNCTION
[ AS ASSIGNMENT | AS IMPLICIT ]

CREATE CONSTRAINT TRIGGER

Define a new constraint trigger.

CREATE CONSTRAINT TRIGGER name
AFTER events ON
table_name constraint attributes
FOR EACH ROW EXECUTE PROCEDURE func_name (args)

CREATE CONVERSION

Define a new conversion.

CREATE [DEFAULT] CONVERSION name
FOR source_encoding TO dest_encoding FROM func_name

CREATE DATABASE

Create a new database.

CREATE DATABASE name
[ [ WITH ] [ OWNER [=] db_owner ]
[ TEMPLATE [=] template ]
[ ENCODING [=] encoding ]
[ TABLESPACE [=] tablespace ] ]

CREATE DOMAIN

Define a new domain.

CREATE DOMAIN name [AS] data_type
[ DEFAULT expression ]
[ constraint [ ... ] ]

Where constraint is:

[ CONSTRAINT constraint_name ]
{ NOT NULL | NULL | CHECK (expression) }

CREATE FUNCTION

Define a new function.

CREATE [ OR REPLACE ] FUNCTION name ( [ [ arg_name ] arg_type [, ... ] ] )
RETURNS ret_type
{ LANGUAGE lang_name
| IMMUTABLE | STABLE | VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| [ [ EXTERNAL ] SECURITY INVOKER | [ [ EXTERNAL ] SECURITY DEFINER
| AS 'definition'
| AS 'obj_file', 'link_symbol'
}...
[ WITH ( attribute [, ... ] ) ]
CREATE GROUP
Define a new user group.

CREATE GROUP name [ [ WITH ] option [ ... ] ]
Where option can be:
SYSID gid
| USER username [, ...]

CREATE INDEX
Define a new index.

CREATE [ UNIQUE ] INDEX name ON table [ USING method ]
( { column | ( expression ) } [ opclass ] [, ... ] )
[ TABLESPACE tablespace ]
[ WHERE predicate ]

CREATE LANGUAGE
Define a new procedural language.

CREATE [ TRUSTED ] [ PROCEDURAL ] LANGUAGE name
HANDLER call_handler [ VALIDATOR val_function ]

CREATE OPERATOR
Define a new operator.

CREATE OPERATOR name ( PROEDURE = func_name
[ , LEFTARG = left_type ] [ , RIGHTARG = right_type ]
[ , COMMUTATOR = com_op ] [ , NEGATOR = neg_op ]
[ , RESTRICT = res_proc ] [ , JOIN = join_proc ]
[ , HASHES ] [ , MERGES ]
[ , SORT1 = left_sort_op ] [ , SORT2 = right_sort_op ]
[ , LTCMP = less_than_op ] [ , GTCMP = greater_than_op ]
)

CREATE OPERATOR CLASS
Define a new operator class.

CREATE OPERATOR CLASS name [ DEFAULT ] FOR TYPE data_type
USING index_method AS
{ OPERATOR strategy_number operator_name [ ( op_type, op_type ) ] [ RECHECK ]
| FUNCTION support_number func_name ( argument_type [, ... ] )
| STORAGE storage_type
} [, ... ]

CREATE RULE
Define a new rewrite rule.

CREATE [ OR REPLACE ] RULE name AS ON event
to table [ WHERE condition ]
do [ ALSO | INSTEAD ] { NOTHING | command | ( command ; command ... ) }

CREATE SCHEMA
Define a new schema.
CREATE SCHEMA schema_name
[ AUTHORIZATION username ] [ schema_element [ ... ] ]
CREATE SCHEMA AUTHORIZATION username
[ schema_element [ ... ] ]

**CREATE SEQUENCE**

Define a new sequence generator.

CREATE [ TEMPORARY | TEMP ] SEQUENCE name
[ INCREMENT [ BY ] increment ]
[ MINVALUE minvalue | NO MINVALUE ]
[ MAXVALUE maxvalue | NO MAXVALUE ]
[ START [ WITH ] start ] [ CACHE cache ] [ NO CYCLE ]

**CREATE TABLE**

Define a new table.

CREATE [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } TABLE table_name ( column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
table_constraint
LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] [ , ... ]
INHERITS ( parent_table [, ... ] )
WITH OIDS | WITHOUT OIDS
ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP }
TABLESPACE tablespace

Where column_constraint is:

[ CONSTRAINT constraint_name ]
[ NOT NULL ] [ NULL ] [ UNIQUE [ USING INDEX TABLESPACE tablespace ] ]
[ PRIMARY KEY [ USING INDEX TABLESPACE tablespace ] ]
CHECK (expression)
REFERENCES ref_table [ ( ref_column ) ]
MATCH FULL | MATCH PARTIAL | MATCH SIMPLE
[ ON DELETE action ] [ ON UPDATE action ]
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

And table_constraint is:

[ CONSTRAINT constraint_name ]
{ UNIQUE (column_name [, ...]) [ USING INDEX TABLESPACE tablespace ] | PRIMARY KEY (column_name [, ...]) [ USING INDEX TABLESPACE tablespace ] | CHECK (expression)
FOREIGN KEY (column_name [, ...]) REFERENCES ref_table [ ( ref_column [, ... ] ) ]
MATCH FULL | MATCH PARTIAL | MATCH SIMPLE
[ ON DELETE action ] [ ON UPDATE action ]
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]

**CREATE TABLE AS**

Define a new table from the results of a query.

CREATE [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } TABLE table_name ( column_name [ , ... ] )
[ WITH | WITHOUT ] OIDS
AS query

CREATE TABLESPACE
Define a new tablespace.

CREATE TABLESPACE tablespace_name [ OWNER username ] LOCATION 'directory'

**CREATE TRIGGER**

Define a new trigger.

CREATE TRIGGER name { BEFORE | AFTER } { event [ OR ... ] }
ON table [ FOR [ EACH ] { ROW | STATEMENT } ]
EXECUTE PROCEDURE func_name ( arguments )

**CREATE TYPE**

Define a new data type.

CREATE TYPE name AS
( attribute_name data_type [, ... ] )
CREATE TYPE name ( INPUT = input_function,
OUTPUT = output_function
[ , RECEIVE = receive_function ]
[ , SEND = send_function ]
[ , ANALYZE = analyze_function ]
[ , INTERNALLENGTH = { internal_length | VARIABLE } ]
[ , PASSEDBYVALUE ]
[ , ALIGNMENT = alignment ]
[ , STORAGE = storage ]
[ , DEFAULT = default ]
[ , ELEMENT = element ]
[ , DELIMITER = delimiter ]
)

**CREATE USER**

Define a new database user account.

CREATE USER name [ [ WITH ] option [ ... ] ]

Where option can be:

SYSSID uid
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| IN GROUP group_name [, ...]
| VALID UNTIL 'abs_time'

**CREATE VIEW**

Define a new view.

CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ... ] ) ] AS query

**DEALLOCATE**

Deallocate a prepared statement.

DEALLOCATE [ PREPARE ] plan_name

**DECLARE**
Define a cursor.

```
DE宣布 name [ BINARY ] [ INSENSITIVE ] [ [ NO ] SCROLL ]
CURSOR [ { WITH | WITHOUT } HOLD ] FOR query
[ FOR { READ ONLY | UPDATE [ OF column [ , ... ] ] } ]
```

### DELETE

Delete rows of a table.

```
DELETE FROM [ ONLY ] table [ WHERE condition ]
```

### DROP AGGREGATE

Remove an aggregate function.

```
DROP AGGREGATE name ( type ) [ CASCADE | RESTRICT ]
```

### DROP CAST

Remove a cast.

```
DROP CAST (source_type AS target_type) [ CASCADE | RESTRICT ]
```

### DROP CONVERSION

Remove a conversion.

```
DROP CONVERSION name [ CASCADE | RESTRICT ]
```

### DROP DATABASE

Remove a database.

```
DROP DATABASE name
```

### DROP DOMAIN

Remove a domain.

```
DROP DOMAIN name [ , ... ] [ CASCADE | RESTRICT ]
```

### DROP FUNCTION

Remove a function.

```
DROP FUNCTION name ( [ type [ , ... ] ] ) [ CASCADE | RESTRICT ]
```

### DROP GROUP

Remove a user group.

```
DROP GROUP name
```

### DROP INDEX

Remove an index.

```
```
<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROP INDEX name [, ...] [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP LANGUAGE</td>
<td>Remove a procedural language.</td>
</tr>
<tr>
<td>DROP [ PROCEDURAL ] LANGUAGE name [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP OPERATOR name ( { left_type</td>
<td>NONE }, { right_type</td>
</tr>
<tr>
<td>DROP OPERATOR CLASS name USING index_method [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP RULE name ON relation [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP SCHEMA name [, ...] [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP SEQUENCE name [, ...] [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP TABLE name [, ...] [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
<tr>
<td>DROP TABLESPACE tablespace_name</td>
<td>Remove a tablespace.</td>
</tr>
<tr>
<td>DROP TRIGGER name ON table [ CASCADE</td>
<td>RESTRICT ]</td>
</tr>
</tbody>
</table>
**DROP TYPE**
Remove a data type.

```
DROP TYPE name [, ...] [ CASCADE | RESTRICT ]
```  

**DROP USER**
Remove a database user account.

```
DROP USER name
```  

**DROP VIEW**
Remove a view.

```
DROP VIEW name [, ...] [ CASCADE | RESTRICT ]
```  

**END**
Commit the current transaction.

```
END [ WORK | TRANSACTION ]
```  

**EXECUTE**
Execute a prepared statement.

```
EXECUTE plan_name [(parameter [, ...]) ]
```  

**EXPLAIN**
Show the execution plan of a statement.

```
EXPLAIN [ ANALYZE ] [ VERBOSE ] statement
```  

**FETCH**
Retrieve rows from a query using a cursor.

```
FETCH [ direction { FROM | IN } ] cursor_name
```  

Where `direction` can be empty or one of:

```
NEXT
PRIOR
FIRST
LAST
ABSOLUTE count
RELATIVE count
count
ALL
FORWARD
FORWARD count
FORWARD ALL
BACKWARD
BACKWARD count
BACKWARD ALL
```
GRANT

Define access privileges.

GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER } [, ...] | ALL [ PRIVILEGES ] } ON [ TABLE ] table_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] } ON DATABASE db_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] } ON TABLESPACE tablespace_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] } ON FUNCTION func_name ([type, ...]) [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] } ON LANGUAGE lang_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] } ON SCHEMA schema_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER } [, ...] | ALL [ PRIVILEGES ] } ON [ TABLE ] table_name [, ...] TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

INSERT

Create new rows in a table.

INSERT INTO table [ ( column [, ...] ) ] { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) | query }

LISTEN

Listen for a notification.

LISTEN name

LOAD

Load or reload a shared library file.

LOAD 'filename'

LOCK

Lock a table.

LOCK [ TABLE ] name [, ...] [ IN lock_mode MODE ] [ NOWAIT ]

Where lock_mode is one of:

ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE

MOVE

Position a cursor.
MOVE [ direction { FROM | IN } ] cursor_name

NOTIFY
Generate a notification.

NOTIFY name

PREPARE
Prepare a statement for execution.

PREPARE plan_name [ (data_type [, ... ] ) ] AS statement

REINDEX
Rebuild indexes.

REINDEX { DATABASE | TABLE | INDEX } name [ FORCE ]

RELEASE SAVEPOINT
Destroy a previously defined savepoint.

RELEASE [ SAVEPOINT ] savpoint_name

RESET
Restore the value of a runtime parameter to the default value.

RESET name
RESET ALL

REVOKE
Remove access privileges.

REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
[,...] | ALL [ PRIVILEGES ] } ON [ TABLE ] table_name [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] }
ON DATABASE db_name [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ CREATE | ALL [ PRIVILEGES ] }
ON TABLESPACE tablespace_name [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ EXECUTE | ALL [ PRIVILEGES ] }
ON FUNCTION func_name ([type, ...]) [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]
REVOKE [ GRANT OPTION FOR ]
{ USAGE | ALL [ PRIVILEGES ] }
ON LANGUAGE lang_name [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name [, ...]
FROM { username | GROUP group_name | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

ROLLBACK
Abort the current transaction.

ROLLBACK [ WORK | TRANSACTION ]

ROLLBACK TO SAVEPOINT
Roll back to a savepoint.

ROLLBACK [ WORK | TRANSACTION ] TO [ SAVEPOINT ] savepoint_name

SAVEPOINT
Define a new savepoint within the current transaction.

SAVEPOINT savepoint_name

SELECT
Retrieve rows from a table or view.

SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
* | expression [ AS output_name ] [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
[ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
[ FOR UPDATE [ OF table_name [, ...] ] ]

Where from_item can be one of:
[ ONLY ] table_name [*] [ AS ] alias [ ( column_alias [, ...] ) ]
( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
function_name [ ( argument [, ...] ) ]
[ AS ] alias [ ( column_alias [, ...] | column_definition [, ...] ) ]
function_name [ ( argument [, ...] ) AS ( column_definition [, ...] ) ]
from_item [ NATURAL ] join_type from_item
[ ON join_condition | USING ( join_column [, ...] ) ]

SELECT INTO
Define a new table from the results of a query.

SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
* | expression [ AS output_name ] [, ...]
INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table
### FROM

```
FROM from_item [, ...]
```

### WHERE

```
WHERE condition
```

### GROUP BY

```
GROUP BY expression [, ...]
```

### HAVING

```
HAVING condition [, ...]
```

### UNION | INTERSECT | EXCEPT

```
{ UNION | INTERSECT | EXCEPT } [ ALL ] select
```

### ORDER BY

```
ORDER BY expression [ ASC | DESC | USING operator ] [, ...]
```

### LIMIT

```
LIMIT { count | ALL }
```

### OFFSET

```
OFFSET start
```

### FOR UPDATE

```
FOR UPDATE [ OF table_name [, ...] ]
```

### SET

Change a runtime parameter.

```
SET [ SESSION | LOCAL ] name { TO | = } { value | 'value' | DEFAULT }
```

```
SET [ SESSION | LOCAL ] TIME ZONE { time_zone | LOCAL | DEFAULT }
```

### SET CONSTRAINTS

Set constraint checking modes for the current transaction.

```
SET CONSTRAINTS { ALL | name [, ...] } { DEFERRED | IMMEDIATE }
```

### SET SESSION AUTHORIZATION

Set the session user identifier and the current user identifier of the current session.

```
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION username
```

```
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION DEFAULT
```

```
RESET SESSION AUTHORIZATION
```

### SET TRANSACTION

Set the characteristics of the current transaction.

```
SET TRANSACTION transaction_mode [, ...]
```

```
SET SESSION CHARACTERISTICS AS TRANSACTION transaction_mode [, ...]
```

Where `transaction_mode` is one of:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
| READ UNCOMMITTED }
```

```
READ WRITE | READ ONLY
```

### SHOW

Show the value of a runtime parameter.

```
SHOW name
SHOW ALL
```

### START TRANSACTION

Start a transaction block.

```
START TRANSACTION [ transaction_mode [, ...] ]
```

Where `transaction_mode` is one of:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
```
**TRUNCATE**

Empty a table.

```
TRUNCATE [ TABLE ] name
```

**UNLISTEN**

Stop listening for a notification.

```
UNLISTEN { name | * }
```

**UPDATE**

Update rows of a table.

```
UPDATE [ ONLY ] table SET column = { expression | DEFAULT } [, ...]
[ FROM from_list ]
[ WHERE condition ]
```

**VACUUM**

Garbage-collect and optionally analyze a database.

```
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table [ (column [, ...] ) ] ]
```