Official PostgreSQL 8.3 Documentation URL: http://www.postgresql.org/docs/8.3/static/

We cover only a subset of what we feel are the most useful constructs that we could squash in a single cheatsheet page

commonly used

[1] New in this release.

Admin Functions

```
COPY .. FROM ..
COPY .. TO ..
current_setting
pg_cancel_backend
pg_column_size
pg_database_size
pg_relation_size
pg_size_pretty
pg_tablespace_size
pg_total_relation_size
set_config
vacuum analyze verbose
vacuum full
```

Common Functions

```
cast,  ::
coalesce
generate_series
greatest
least
nullif
random
```

Sequence (Serial) Functions

```
currval
lastval
nextval
```

String Functions

```
||
ascii
chr
initcap
length
lower
lpad
ltrim
md5
position
quote_ident
quote_literal
regexp_matches
regexp_replace
regexp_split_to_array
regexp_split_to_table
repeat
replace
rpad
rtrim
split_part
strpos
substr
trim
upper
```

Database Globals

```
current_date
current_time
current_timestamp
current_user
localtime
```

Date Functions

```
age
date_part(text, timestamp)
 century
 day
 decade
 dow
 doy
 epoch
 hour
 month
 quarter
 second
 week
 year
date_trunc
extract
interval
to_char
to_date
to_timestamp
```

Date Predicates

```
overlaps
```

Array Constructs

```
ANY(array)
ARRAY[[4,5,6],..]
ARRAY()
array_append
array_cat
array_dims
array_lower
array_prepend
array_to_string
array_upper
SOME(array)
string_to_array
```

Array Operators

```
=
<>
<
>
<=
||
```

Math Operators

```
% , ^, |/
||/,!, !!
@, &, |
#,~, <<
>>
```

Math Functions

This is a subset

```
abs
cbrt
ceiling
degrees
exp
floor
log
ln
mod
pi
power
radians
random
sqrt
trunc
```

Trig Functions

```
acos
asin
atan
atan2
cos
cot
sin
tan
```

Command Line

```
pg_dump
pg_dumpall
pg_restore
psql
```

JOIN Types

```
CROSS JOIN
EXCEPT (ALL)
FULL JOIN
[INNER] JOIN
INTERSECT (ALL)
LEFT JOIN
NATURAL JOIN
RIGHT JOIN
UNION (ALL)
```

SQL Keywords

```
BETWEEN .. AND
CASE WHEN .. END
DELETE FROM
DISTINCT
DISTINCT ON
EXISTS
FROM
GROUP BY
HAVING
ILIKE
IN(..)
LIKE
LIMIT ..OFFSET
NOT
NOT IN(..)
NULLS FIRST[1]
NULLS LAST[1]
ORDER BY
SELECT
SET
SIMILAR TO
TRUNCATE TABLE
UPDATE
USING
WHERE
```

Aggregates

```
avg
bit_and
bit_or
boolean_and
boolean_or
count
count(DISTINCT)
every
max
min
stddev
stddev_pop (a bunch more)
sum
sum(DISTINCT)
variance
xml_agg[1]
```

DDL

```
ADD CONSTRAINT
CREATE AGGREGATE
CREATE CAST
CREATE (DEFAULT) CONVERSION
CREATE DATABASE
CREATE DOMAIN
CREATE [OR REPLACE] FUNCTION
CREATE (UNIQUE) INDEX
CREATE LANGUAGE
CREATE OPERATOR
CREATE OPERATOR FAMILY [1]
CREATE ROLE
CREATE RULE
CREATE SCHEMA
CREATE SEQUENCE
CREATE TABLE
CREATE TABLESPACE
ALTER TABLE
CREATE TYPE
CREATE [OR REPLACE] VIEW
DROP [object]
```

Enums [1]

```
> < <= >= =
enum_cmp
enum_first
enum_larger
enum_last
enum_range
enum_smaller
```

XML [1]

```
database_to_xml
database_to_xmlschema
query_to_xml
query_to_xml_and_xmlschema
table_to_xml
xmlattributes
xmlcomment
xmlconcat
xmlelement
xmlforest
xpath
xmlpi
xmlroot
```

Languages

```
c
pljava
plpgsql
plperl(u)
plphp
plproxy
plpython
plr
plruby
plsh
pltcl
sql
```

Key information_schema Views

```
columns
sequences
tables
views
```

Key pg_catalog Tables/Views

```
pg_class
pg_rules
pg_settings
pg_stat_activity
pg_stat_database
pg_tablespaces
```

Large Object

Server	Client
lo_creat	lo_close
lo_create	lo_creat
lo_export	lo_create
lo_import	lo_export
lo_unlink	lo_import
	lo_lseek
	lo_open
	lo_read
	lo_tell
	lo_unlink
	lo_write

DATA TYPES

Below are common data types with common alternative names.
Note: There are many more and one can define new types with create type. All table structures create an implicit type struct as well.

datatype[] - e.g. varchar(50)[] (defines an array of a type)

```
bit
boolean
bytea
character varying(length) - varchar(length)
character(length) - char(length)
date
enum [1]
double precision - float4 float8
integer - int4
bigint - int8
numeric(length,precision)
oid
serial - serial4
bigserial - serial8
text
time without timezone - time
time with timezone - timez
timestamp without timezone - timestamp
timestamp with timezone - timestampz
xml [1]
```

ADMIN EXAMPLES

```
select pg_size_pretty(pg_tablespace_size('pg_default')) as tssize,
       pg_size_pretty(pg_database_size('somedb')) as dbsize,
       pg_size_pretty(pg_relation_size('someschema.sometable')) as tblsize;
```

```
--Example importing data to table sometable
--from tab delimited where NULLs appear as NULL
COPY sometable FROM "/path/to/textfile.txt" USING DELIMITERS '\t' WITH NULL As 'NULL';

--Example exporting a query to a comma separated (CSV) called textfile.csv
--setting NULLS to text NULL
COPY (SELECT * FROM sometable WHERE somevalue LIKE '%') TO '/path/to/textfile.csv'
 WITH NULL As 'NULL' CSV HEADER QUOTE AS '"';
```

```
vacuum analyze verbose;
vacuum sometable;
vacuum full;
```

```
--Kills all active queries in selected db and list out process id
--and usename of process and if kill successful
SELECT procpid, usename, pg_cancel_backend(procpid)
FROM pg_stat_activity
WHERE datname = 'somedb';
```

JOIN EXAMPLES

```
SELECT o.order_id, o.order_date, o.approved_date,
 COUNT(i.item_id) As nlineitems,
 SUM(i.unit_price*i.num_units) As total
FROM orders o
 INNER JOIN orderitems i ON o.order_id = i.order_id
GROUP BY o.order_id, o.order_date, o.approved_date
HAVING SUM(i.unit_price*i.num_units) > 200
ORDER BY o.approved_date NULLS FIRST;
```

```
SELECT 'x' As bucket, o.order_id, o.order_date,
 COUNT(i.item_id) As nlineitems,
 SUM(i.unit_price*i.num_units) As total
FROM xorders o
 INNER JOIN xorderitems i ON o.order_id = i.order_id
GROUP BY o.order_id, o.order_date
UNION ALL
SELECT 'y' as bucket, o.order_id, o.order_date,
 COUNT(i.item_id) As nlineitems,
 SUM(i.unit_price*i.num_units) As total
FROM yorders o
 INNER JOIN yorderitems i ON o.order_id = i.order_id
GROUP BY o.order_id, o.order_date
ORDER BY 1,3,2;
```

DDL EXAMPLES

```
CREATE DATABASE somedb
 WITH OWNER = somelogin
 ENCODING = 'WIN1252';
```

```
CREATE TABLE orders(
  order_id serial NOT NULL,
  order_addeddt timestamp without time zone,
  order_rating rating,
  CONSTRAINT pk_orders_order_id PRIMARY KEY (order_id)
)
WITH (OIDS=FALSE);
```

```
CREATE TYPE rating AS
 ENUM('none', 'bronze', 'silver',
       'gold', 'platinum');
```

```
CREATE AGGREGATE sum(text) (
  SFUNC=textcat,
  STYPE=text
);
```

```
CREATE OR REPLACE FUNCTION cp_test(somearg integer)
  RETURNS SETOF sometable AS
$$SELECT * FROM sometable where msg_id = $1;$$
  LANGUAGE 'sql' STABLE;
```

UPDATE/INSERT/DELETE EXAMPLES

```
UPDATE sometable
 SET somevalue = 5
 WHERE sometable.somename = 'stuff';
```

```
UPDATE sometable
 SET calccount = s.thecount
  FROM (SELECT COUNT(someothertable.someid) as thecount,
    someothertable.someid
    FROM someothertable
     GROUP BY someothertable.someid) s
  WHERE sometable.someid = s.someid;
```

```
--This only works on 8.1+ --

INSERT INTO orders(order_addeddt, order_rating)
 VALUES ('2007-10-01 20:40', 'gold'),
 ('2007-09-01 11:00 AM', 'silver'),
   ('2007-09-02 10:00 PM', 'none'),
   ('2007-10-10 PM', 'bronze');
```

```
--Pre 8.1+ only supports single values inserts

INSERT INTO orders(order_addeddt, order_rating)
 VALUES ('2007-10-01 20:40', 'gold');
```

```
DELETE FROM sometable
 WHERE somevalue = 'something';
```

```
--This is a fast delete that deletes everything in a table so be cautious.
--Also only works on tables not referenced in foreign key contraints
TRUNCATE TABLE sometable;
```

MISCELLANEOUS EXAMPLES

```
--Enum range query using enum defined above - returns all orders in (bronze, silver, gold)
--Sorts in order bronze, silver, gold.  Keep in mind if you reverse gold and bronze you get nothing
SELECT *
       FROM orders
       WHERE order_rating
       BETWEEN 'bronze' AND 'gold'
       ORDER BY order_rating;
```

```
SELECT monthperiod.*,
 array_to_string(ARRAY(SELECT (d + 1)::varchar(20)
   FROM generate_series(0,30) d
        WHERE monthperiod.start_date + (d || ' day')::interval
     BETWEEN monthperiod.start_date
         AND
          monthperiod.end_date), ',') as thedays
 FROM (SELECT (n + 1) As mnum,
  trim(to_char(date '2007-01-01' + (n || ' month')::interval, 'Mon')) As short_mname,
  trim(to_char(date '2007-01-01' + (n || ' month')::interval, 'Month')) As long_mname,
  date '2007-01-01' + (n || ' month')::interval As start_date,
  date '2007-01-01' + ((n + 1) || ' month')::interval +  - '1 day'::interval As end_date
 FROM generate_series(0,11) n) As monthperiod;

 EXPLAIN ANALYZE SELECT * FROM sometable;
```

COMMAND LINE EXAMPLES

These are located in bin folder of PostgreSQL
To get more info about each do a --help e.g. psql --help

```
pg_dump -i -h someserver -p 5432 -U someuser -F c -b -v -f "\somepath\somedb.backup" somedb
pg_dumpall -i -h someserver -p 5432 -U someuser -c -o -f "\somepath\alldbs.sql"
pg_restore -i -h someserver -p 5432 -U someuser -d somedb -l "\somepath\somedb.backup"
psql -h someserver -p 5432 -U someuser -d somedb -f "\somepath\somefiletorun.sql"
psql -h someserver -p 5432 -U someuser -d somedb -c "CREATE TABLE sometable(st_id serial, st_name varchar(25))"
```

[1] New XML feature - output query as xml
-P "t" only output rows

```
psql -h someserver -p 5432 -U someuser -d somedb -P "t" -c "SELECT query_to_xml('select * from sometable', false, false, 'sometable')" -o "outputfile.xml";
```