Quick PostgreSQL Reference

Tools

psql	Command-line client packaged with PostgreSQL. Good for automating SQL jobs, copying data, outputing simple html reports.
createdb, dropdb	Commands for creating and dropping a database from the OS shell.
pg_restore, pg_dump	Command-line tool for creating and restoring compressed or tar backups.
pgsql2shp, shp2pgsql	Command-line tools for importing/exporting ESRI Shapefiles and DBFs packaged with PostGIS (free spatial extender for PostgreSQL).
pgAdmin III	Popular graphical user interface packaged with PostgreSQL.
phpPgAdmin	Similar to phpMyAdmin, it allows administration of PostgreSQL via web interface.

Common Tasks with psql, createdb, pg_dump, pg_restore, shp2pgsql

Create user:
```
createuser -h localhost -p 5432 -U postgres -W -P someuser
```

Create database:
```
createdb -h localhost -p 5432 -U postgres -W -O someuser -T template1 somedb
createdb -h localhost -p 5432 -U postgres -W -O someuser -T template_postgis somedb
```

Using psql client:
```
psql [OPTION]... [DBNAME [USERNAME]]
```
Some options:

-d	or	--**dbname**=DBNAME	database name to connect to (defaults to the system user name)
-h	or	--**host**=HOSTNAME	database server host or socket directory (defaults to localhost)
-p	or	--**port**=PORT	database server port (defaults to 5432)
-U	or	--**username**=USERNAME	database user name (defaults to system user name

Execute an SQL script from file:
```
psql –h localhost –U postgres –p 5432 –f myscript.sql
```

Load data into an existing table from a CSV or some text file:
```
psql –h localhost –U postgres –d somedb –c "\copy sometable FROM 'data.csv' WITH CSV"
psql –U postgres –d dsomedb –c "\copy sometable FROM 'notes.txt' WITH DELIMITER AS','"
```

Execute a single statement against a database:
```
psql –U postgres –p 5432 –d somedb –c "CREATE TABLE test(id serial PRIMARY KEY, notes text);"
```

Output data in html format:
```
psql –h localhost –p 5432 –U postgres –d somedb –H –c "SELECT * FROM pg_tables"
```

Create a compressed backup:
```
pg_dump –h someserver –p 5432 –U someuser –F c –b –v –f "somedb.backup" somedb
```

Restore a compressed backup:
```
pg_restore –h someserver –d targetdb –U someuser somedb.backup
```

Load an ESRI Shapefile (requires PostGIS install):
```
shp2pgsql –s 27700 somefile.shp sometable | psql –d somedb –U someuser –p 5432 –h localhost
```

PSQL Interactive Mode

Initiate PostgreSQl interactive terminal:

```
psql –U username –p 5432 –h localhost –d somedb
```

Common psql commands:

\q	Quit
:q	Quit more screen
\?	Help on psql commands
\h some command	Help on SQL commands
\connect somedb	Switch database
\l	List all databases
\dtv a*	List tables and views that start with a.
\du	List user/group roles and their group memberships and server level permissions
\d sometable	List columns, data types, and constraints for a table
\d+ sometable	List additional detail about some table
\i somefile	Execute SQL script stored in a file.
\o somefile	Output contents to file
↑ & ↓ keyboard arrows	Retrieve prior commands
\x	toggle expanded output (show columns as rows and vice versa)
\timing	Toggle query timing on and off
\copy	Copy from client computer to server and from server to client computer. Example: The following command string copies data to local client computer in CSV format with header. `\copy (SELECT * FROM sometable) TO 'sometable.csv' WITH HEADER CSV FORCE QUOTE`

Basic SQL commands

Create user	`CREATE USER someuser`
	`  WITH CREATEDB`
	`  LOGIN ENCRYPTED PASSWORD 'secret'`
	`  VALID UNTIL 'infinity';`

Create database	`CREATE DATABASE somedatabase WITH OWNER = someuser;`

Delete object	`DROP DATABASE somedatabase;`
	`DROP TABLE sometable;`
	`ALTER TABLE sometable DROP COLUMN somecolumn;`
	`DROP FUNCTION somefunction;`

Delete object and dependents	`DROP TABLE sometable CASCADE;`

Create an empty table	`CREATE TABLE sometable (`
	`  id serial PRIMARY KEY,`
	`  city varchar(50),`
	`  capital boolean NOT NULL DEFAULT false);`

Create table from another table	`CREATE TABLE sometable AS`
	`  SELECT id, value FROM othertable`
	`  WHERE value >100;`

Create a view (virtual table)	`CREATE OR REPLACE VIEW someview AS`
	`  SELECT * FROM sometable`
	`  WHERE modif_date >'2012-01-01'::date;`

Add column	`ALTER TABLE sometable ADD COLUMN somecolumn timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP;`
Rename column	`ALTER TABLE sometable RENAME somecolumn TO othername;`
Add auto-incremented column	`ALTER TABLE sometable ADD COLUMN id SERIAL;`
Add primary key	`ALTER TABLE sometable ADD PRIMARY KEY(id);`

Add index	`CREATE INDEX someindex`
	`  ON sometable`
	`  USING btree (lower(somecolumn));`

Create a check constraint	`ALTER TABLE sometable`
	`  ADD CONSTRAINT somecheckcontraint`
	`  CHECK (value BETWEEN 1 AND 100);`

View data	`SELECT somecolumn, someothercolumn FROM sometable;`

View all columns for 100 rows	`SELECT *`
	`  FROM sometable`
	`  LIMIT 100;`

View unique data	`SELECT DISTINCT name, surname FROM sometable;`

Query table	`SELECT * FROM sometable`
	`  WHERE somecolumn = 'XXX'`
	`  OR someothercolumn ILIKE '%dev8d%'`

Sort query results	`SELECT *`
	`  FROM sometable`
	`  ORDER BY id ASC;`

Aggregate rows	`SELECT username, COUNT(*) AS visits`
	`FROM somestatstable`
	`GROUP BY username`
	`HAVING COUNT(*)>100;`

Inserting row	`INSERT INTO sometable(id, notes) VALUES (1,'first')`

Inserting multiple rows	`INSERT INTO sometable(id, notes) VALUES`
	`  (1, 'first'),`
	`  (1, 'second'),`

Copy data from one table to another	`INSERT INTO sometable(id, notes)` `  SELECT gid, name FROM someothertable;`
Insert from a tab delimited file	`COPY sometable FROM "/tmp/mydata.txt"` `  WITH DELIMITER '\t' NULL AS 'NULL';`

Update some value	`UPDATE sometable`
	`  SET somecolumn = 'newvalue'`
	`  WHERE anycolumn = 'somevalue';`

Update some column	`UPDATE sometable` `  SET modif_date = now();`

Cross update – update some column with values from another table	`UPDATE onetable a`
	`  SET price = b.price`
	`  FROM othertable AS b`
	`  WHERE a.id = b.id;`

Cast data type	`SELECT CAST(1 AS text);` `SELECT 1::text`

Join tables on row match	`SELECT a.id AS aid, b.id AS baid, b.name`
	`  FROM sometable a, someothertable b`
	`  WHERE a.id = b.id;`

Join tables also returning unmatched rows from the first table	`SELECT a.id AS aid, b.id AS baid, b.name`
	`  FROM sometable a`
	`  LEFT JOIN someothertable b`
	`  ON (a.id = b.id);`

Join tables also returning unmatched rows from both tables	`SELECT a.id AS aid, b.id AS baid, b.name`
	`  FROM sometable a`
	`  FULL JOIN someothertable b`
	`  ON (a.id = b.id);`

View running queries	`SELECT * FROM pg_stat_activity;`
Set owner	`ALTER TABLE sometable OWNER TO someuser;`
Grant privileges	`GRANT ALL ON TABLE sometable TO someuser;` `GRANT SELECT ON TABLE sometable TO spublic;`
Revoke privileges	`REVOKE ALL PRIVILEGES ON sometable FROM someuser;`

Create SQL function	`CREATE OR REPLACE FUNCTION somefn(p_someparam text,`
	`  OUT id integer,`
	`  OUT somecol text)`
	`RETURNS SETOF record AS`
	`$$`
	`  SELECT id, somecol`
	`  FROM sometable`
	`  WHERE somecol LIKE $1||'%';`
	`$$`
	`LANGUAGE 'sql' STABLE;`
	`--Example call:`
	`SELECT * FROM somefn('A');`

Create PLpgSQL function	`CREATE FUNCTION register(p_name varchar, p_surname varchar)`
	`RETURNS void AS`
	`$$`
	`BEGIN`
	`  IF NOT EXISTS (`
	`  SELECT * FROM sometable`
	`  WHERE name = p_name AND surname = p_surname`
	`  ) THEN`
	`  INSERT INTO users (name, surname)`
	`  VALUES(p_name, p_surname);`
	`  END IF;`
	`RETURN;`
	`END;`
	`$$`
	`LANGUAGE PLpgSQL VOLATILE;`
	`--Example call:`
	`SELECT register ('John','Smith');`

Basic PostGIS functions

PostGIS version	`SELECT PostGIS_Full_Version();`
Add geometry column	`SELECT AddGeometryColumn ('someschema','sometable','geom',4326,'POINT',2);`
Add geography column	`ALTER TABLE sometable ADD COLUMN geog geography(POINT,4326);`
Add spatial index	`CREATE INDEX somespatindex ON sometable USING gist (geom);`
Reorder data on disk to match spatial index	`CLUSTER somespatindex ON sometable;`
Create point	`SELECT ST_MakePoint(x, y, [z], [m]);`
Create line	`SELECT ST_MakeLine( ST_MakePoint(x1, y1), ST_MakePoint(x2, y2) );`
Create rectangle	`SELECT ST_MakeEnvelope( xmin, ymin, xmax, ymax, srid );`
Convert LAT, LON to geometry	`SELECT ST_Point(-71.1, 42.3);`
Convert LAT, LON to geography	`SELECT geography(ST_SetSRID(ST_Point(-71.1, 42.3),4326));`
Convert geometry to geography type	`UPDATE sometable SET geog = CASE WHEN (ST_SRID(geom) != 4326) THEN ST_Transform(geom, 4326)::geography ELSE geog::geography END;`
Find spatial reference system (SRS/SRID)	`SELECT DISTINCT ST_SRID(geom) FROM sometable;`
Find best SRS for geography feature	`SELECT _ST_BestSRID(geog) FROM sometable;`
Set spatial reference system	`UPDATE sometable SET geom = ST_SetSRID(ST_Point(lon_col, lat_col), 4326);`
Transform geometry to new spatial reference system	`UPDATE sometable SET geom = ST_Transform(geom, 27700);`
Find geometry type	`SELECT DISTINCT ST_GeometryType(geom) FROM sometable;`
Derive centroids for polygons and lines	`UPDATE sometable SET centroidcolumn = ST_PointOnSurface(geom);`
Add bounding box to the geometry (speed queries)	`UPDATE sometable SET the_geom = PostGIS_AddBBox(the_geom) WHERE PostGIS_HasBBox(the_geom) = false;`
Get geographic extent of table	`SELECT ST_Extent(geom) FROM sometable;`
View geometry as text	`SELECT ST_AsText(geom) FROM sometable LIMIT 1;`
Calculate area	`SELECT *, ST_Area(geom) FROM sometable;`
Merge/dissolve geometries	`SELECT ST_Union(geom) FROM sometable WHERE municipality = 'London';`
Convert geometry to GeoJSON	`UPDATE sometable SET jsoncol = ST_AsGeoJSON(geom, 2);`
Explode multipolygons to polygons	`SELECT *, (st_dump(geomcol_multi)).geom AS geom_poly FROM sometable;`

Intersecting geometries (get all stations in London)	`SELECT s.*`
	`  FROM station s, cities c`
	`  WHERE c.name = 'London'`
	`  AND ST_Intersects(s.geom, c.name);`

Get data that overlaps map window	`SELECT * FROM sometable` `  WHERE geom && ST_SetDRID(('BOX(1 1,10 10)'::box2d);`

Find my nearest hotel	`SELECT h.*`
	`FROM hotels h,`
	`  (SELECT`
	`  ST_SRID(ST_Point(-71.1, 42.3), 4326) AS geom`
	`  ) location`
	`ORDER BY`
	`  ST_Distance(h.geog, location.geom::geography) DESC`
	`LIMIT 1;`

Find all hotels in 1 km radius around me (using _ST_BestSRID() to choose the best data projection)	`SELECT h.*,`
	`  ST_Distance(h.geog, location.geog) AS distance`
	`FROM hotels h,`
	`  (SELECT ST_SRID(`
	`  ST_Point(-71.1, 42.3)`
	`  , 4326)::geography AS geog`
	`  ) location`
	`WHERE h.geog @ geography(`
	`  ST_Transform(`
	`  ST_Buffer(`
	`  ST_Transform(`
	`  geometry(location.geog),`
	`  _ST_BestSRID(h.geog, location.geog)`
	`  ),`
	`  1000),`
	`  4326))`
	`ORDER BY ST_Distance(h.geog, location.geog) DESC;`

Spatial operators:

&&	Returns TRUE if A's bounding box overlaps B's.
&<	Returns TRUE if A's bounding box overlaps or is to the left of B's.
&<|	Returns TRUE if A's bounding box overlaps or is below B's.
&>	Returns TRUE if A' bounding box overlaps or is to the right of B's.
<<	Returns TRUE if A's bounding box is strictly to the left of B's.
<<|	Returns TRUE if A's bounding box is strictly below B's.
=	Returns TRUE if A's bounding box is the same as B's.
>>	Returns TRUE if A's bounding box is strictly to the right of B's.
@	Returns TRUE if A's bounding box is contained by B's.
|&>	Returns TRUE if A's bounding box overlaps or is above B's.
|>>	Returns TRUE if A's bounding box is strictly above B's.
~	Returns TRUE if A's bounding box contains B's.
~=	Returns TRUE if A's bounding box is the same as B's.