SQL (Structured Query Language) Notes

***THINGS TO FIGURE OUT!***

* **Why is default Andrew and why is the password wrong?**

**BASIC NOTES**

pgAdmin 4 is the gui

SQL Shell (psql) is the terminal

Psql -h localpost -p 5432 -U postgres test 🡪 host port user database

\l 🡪 gives you a list of all the databases

\c database 🡪 connects you to that database

CREATE DATABASE test 🡪 creates database test

DROP DATABASE test 🡪 gets rid of database test

**SET UP**

can use sql shell

can use pythonanywhere's bash shell

can use DBeaver

ALSO TERMINAL

but if you want to terminal, you have to set the directory to where postgres.exe is

C:\Program Files\PostgreSQL\13\bin

Follow link for steps

https://linuxhint.com/connect-to-postgresql-database-command-line-windows/

wget https://www.pg4e.com/tools/sql/library.csv

to use wget, download wget.exe and place it in system32

to RUN wget properly need to open terminal as an admin

**WEEK 1**

CRUD operations (create, read, update, delete)

$ psql –U postgres

\l

* shows all the databses in a server

\dt

* shows all relations in a databse

\d+

* show schema of a relation

\q

* you're out of psql and back into linux
  + but your console is left open so you don't need to log in again

control d or exit

* take you out of psql and closes console which means you would need to start over and log in again

-------------------------------------------------------------------

**Connecting to database**

psql -h pg.pg4e.com -p 5432 -U pg4e\_4e9f6b3537 pg4e\_4e9f6b3537

pg4e\_p\_fb8ba898f06f8ec **ERASE THIS BEFORE POSTING**

**How to create database as super user**

postgres=# CREATE USER WITH PASSWORD 'secret';

postgres=# CREATE DATABASE people WITH OWNER '';

postgres=# \q

**If you want to delete any exisitng tables/relations**

DROP TABLE users; # if you want to try it out make sure you do this first

**How to make a table**

CREATE TABLE users(

name VARCHAR(128)

email VARCHAR(128)

);

apparently it doesnt need to be upper case

professor just did it for clarity

**WEEK 2**

INSERT INTO <table name> (<column name1>, <column name2>) VALUES ('blah','blah') ;

this inserts those values to column 1 and 2 in that order from that table

DELETE FROM <table name> WHERE <column name>='blah' ;

deletes a specific entry from a column in a specific table

DELETE FROM <table name> ;

deletes all the rows from that table

**so BE CAREFUL!**

UPDATE <table> SET <column1>='blah1' WHERE <column2>='blah2' ;

*setting all values in column 1 to blah where values of column 2 are blah 2*

SELECT \* FROM <table> WHERE <column>='blah' ;

*only retrieving columns that = blah*

SELECT \* FROM <table> **; getting everything**

SELECT \* FROM <table> ORDER BY <column> ASC **or** DESC ;

*by default it's ASC so only have to specify if you want DESC*

SELECT \* FROM <table> WHERE <column> LIKE '%e%' ;

*%e% is a wild card...looking for entries with e anywher*

*but it'll do a full table scan which is sort of inefficient*

SELECT \* FROM <table> ORDER BY <column> DESC OFFSET 1 LIMIT 2;

*offset skips the first entry then limit only shows the next two entries*

SELECT COUNT(\*) FROM <table> WHERE <column>='blah'

*this counts how many rows satisfies this criteria*

* the asterisk after select is saying that you want all the columns, but you COULD instead specify only specific columns you want to see

-------------------------------------------------------------------

Things you can specify for your schema

CHAR(n)

VARCHAR(n)

* CHAR is efficient, but only when you know exactly how many characters its going to be all the time, and if the characters aren't too long
* othewise use VARCHAR

TEXT

* they arent simple 8 bit characters, but rather long paragraphs or HTML pages

BYTEA(n)

* blobs of info small images

SMALLINT (-32768, 32768)

INTEGER (2 billion)

BIGINT (2\*10^18)

* mostly just use integer

REAL(32 bits number) *seven digits of accuracy AKA FLOAT*

DOUBLE PRECISION (64 bit) *10\*\*304 with 14 digits of accuracy*

NUMERIC (accuracy, decimal) *used for money*

TIMESTAMP - y/m/d h/m/s

DATE - y/m/d

TIME - h/m/s

NOW() - gives you current date

-------------------------------------------------------------------

**Keys and Indexes**

DROP TABLE users; *if you want to try it out make sure you do this first*

CREATE TABLE users(

id SERIAL,

name VARCHAR(128),

email VARCHAR(128) UNIQUE,

PRIMARY KEY(id)

);

* we're creating a column called id which is sequenced by some function SERIAL
* and we're saying that we're doing to use the column id as our primary key or index
* UNIQUE - is a logical key that says you can't have the same email for any rows

NOT SURE WHAT THIS MEANS

When use SERIAL or HASH?

name, sorting --> SERIAL

GUI --> HASH

---------------------------------------------------------

**Using NOT NULL**

CREATE TABLE automagic(

id SERIAL,

name VARCHAR(32),

height REAL NOT NULL,

PRIMARY KEY (id)

);

* NOT NULL means...you must provide a value for that column when using INSERT or UPDATE
* REAL is the same as float

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

After downloading a csv. file with wget

you can copy the exact materials to existing table

* type...wget https://www.pg4e.com/tools/sql/library.csv
* to use wget, download wget.exe and place it in system32
* to RUN wget properly need to open terminal as an admin

\copy track\_raw(title,artist,album,count,rating,len) FROM 'library.csv' WITH DELIMITER ',' CSV;

* *here the table is track\_raw*

**WEEK 3**

**CREATE** **TABLE** artist (

id SERIAL,

name **VARCHAR**(128) **UNIQUE**,

**primary** **KEY**(id)

);

**CREATE** **TABLE** album (

id SERIAL,

title **VARCHAR**(128) **UNIQUE**,

artist\_id **INTEGER** **REFERENCES** artist(id) **ON** **DELETE** **CASCADE**,

**PRIMARY** **KEY**(id)

);

/\*ON DELETE OPTION\*/

/\*Default/RESTRICT-dont allow changes that break the contraint\*/

/\*CASCADE-adust child rows by removing or updating to maintain consistency\*/

/\*SET NULL-set the foreign key columns in the child rows to null\*/

/\*If you want to use set null you have to specify INTEGER/VARCHAR/etc then NULL\*/

/\*Im guessing the parent row is the part that comes after REFERENCES\*/

**CREATE** **TABLE** genre (

id SERIAL,

name **VARCHAR**(128) **UNIQUE**,

**primary** **KEY**(id)

);

**CREATE** **TABLE** track (

id SERIAL,

title **VARCHAR**(128),

len **INTEGER**,

rating **INTEGER**,

**count** **INTEGER**,

album\_id **INTEGER** **REFERENCES** genre(id) **ON** **DELETE** **CASCADE**,

genre\_id **INTEGER** **REFERENCES** album(id) **ON** **DELETE** **CASCADE**,

**UNIQUE**(title, album\_id),

**PRIMARY** **KEY**(id)

);

**INSERT** **INTO** artist (name) **VALUES** ('Led Zeppelin');

**INSERT** **INTO** artist (name) **VALUES** ('AC/DC');

**SELECT** \* **FROM** artist;

**INSERT** **INTO** album (title, artist\_id) **VALUES** ('Who Made Who', 2);

**INSERT** **INTO** album (title, artist\_id) **VALUES** ('IV', 1);

**INSERT** **INTO** genre (name) **VALUES** ('Rock');

**INSERT** **INTO** genre (name) **VALUES** ('Metal');

**INSERT** **INTO** track (title, rating, len, **count**, album\_id, genre\_id)

**VALUES** ('Black Dog', 5, 297, 0, 2, 1) ;

**INSERT** **INTO** track (title, rating, len, **count**, album\_id, genre\_id)

**VALUES** ('Stairway', 5, 482, 0, 2, 1) ;

**INSERT** **INTO** track (title, rating, len, **count**, album\_id, genre\_id)

**VALUES** ('About to Rock', 5, 313, 0, 1, 2) ;

**INSERT** **INTO** track (title, rating, len, **count**, album\_id, genre\_id)

**VALUES** ('Who Made Who', 5, 207, 0, 1, 2) ;

/\*use control+shift+/ to comment things out\*/

**SELECT** album.title, artist.name

**FROM** album **JOIN** artist

**ON** album.artist\_id = artist.id

**SELECT** album.title, album.artist\_id, artist.id, artist.name

**FROM** album **INNER** **JOIN** artist

**ON** album.artist\_id = artist.id;

**SELECT** track.title, track.genre\_id, genre.id, genre.name

**FROM** track **CROSS** **JOIN** genre;

/\*You would almost never do this, but just for demonstration\*/

**SELECT** track.title, genre.name

**FROM** track **JOIN** genre

**ON** track.genre\_id = genre.id;

**SELECT** track.title, artist.name, album.title, genre.name

**FROM** track

**JOIN** genre **ON** track.genre\_id = genre.id

**JOIN** album **ON** track.album\_id = album.id

**JOIN** artist **ON** album.artist\_id = artist.id;

**select** \* **from** track;

**select** \* **from** genre;

/\*Delete from genre where name='Metal'\*/

/\*If we were to execute the line above, it would get rid of all entries in genre with metal along with all track's rows with genre\_id=2\*/