 # Commands
- pwd: tells you your current directory loc
- Ls: gives list of what is contained in current directory
- Cd Desktop/databases: takes me to databases folder
- Cd .. : allows you to move back a folder

 # Display Settings
- .tables will give you the names of all the tables in a DB
- .mode column gives output in column view
- .mode csv gives output in csv view
- .headers on adds headers to output
- .help provides additional information
- .shell cls to clear powershell (in windows!)
- .schema shows column names for all tables
- PRAGMA table_info(tablename); also shows details for columns for all tables

 # Sqlite3 Commands
- ; always needs to be at the end of a command
- Ctrl+C to get back to command prompt (windows)
- Sqlite3 databasename.db : takes you to the database

 # Select Queries
Query is a code you run to get data
Select
- select [columns] from [tables]; *gives you back ALL of the columns*
e.g. select * from facilities;
- Select col1, col2, col6, col10 from table; *to select specific columns only*
- Select col1, col2 from table limit 5; *gives you the first 5 lines of data from the table. Sets a limit to 5 rows*

 # Math
equals =
greater than >
Less than <
Greater than or equal to > =
Less than or equal to <=
Not equal to !=
Divide /
Multiply *
Round()
Can use all the above to perform math on data
e.g. select id, name, member_cost, monthly_maintenance from facilities where member_cost > 0 and member_cost < monthly_maintenance / 50;

 # Misc Operators
- In: within a list of values
- like: like a word etc within the values
e.g. select *
from facilities
where name like '%Tennis';
- Between: falls between two values
e.g. select id, last_name, first_name, join_date
from members
where join_date between '2012-09-01' and '2012-09-02';
- Is null/is not null: includes/does not include values that are null. *Can be used for both strings and integers. Not null includes 0!*
A note on wildcard: %Tennis allows SQL to output values with anything before the word ' Tennis' e.g. table tennis.. Tennis% outputs values with any word after Tennis, e.g. tennis court. %Tennis% will allow sql to output anything with the word tennis in it, so both table tennis and tennis court.

 # Distinct
e.g. select (count(distinct address)) from members;

 # Aggregate Functions
- Max: max(columnname)
e.g. select max(monthly_maintenance)
from facilities;
Min: min(columnname)
Sum: sum(columnname)
Avg: avg(Columnname)
Count: count(columnname)
Count Distinct: count(distinct colname)

 # Group by, Order by
- Group by doesn't make sense on its own unless its aggregated at the top with something
- You can label columns as 1 and 2 instead of taking col names if you say group by 1 and order by 2
- Order by can be used with descending and ascending
e.g.
Select last_name, count(*)
from members
group by 1
order by 2 desc;

 # Queries with Conditions
Allows you to set conditions on your output by using where and having
e.g. select round(avg(monthly_maintenance)) from facilities where monthly_maintenance < 3000;
select last_name, count(*) as frequency from members
Group by last_name
having frequency > 1;
Having specifies a search condition for a group or an aggregate function used in SELECT statement. A Group by clause is required before having

 # Naming Columns
By writing 'as column name' after selecting a column allows us to name a column in the output (e.g. above in having)

 # Date/Time
In Sqlite
select start_time,
  strftime('%Y-%m', start_time) as month,
  strftime('%Y', start_time) as year
from bookings; *Where* '%Y-%m' gives month and '%Y' gives year in Postgres there are two ways of doing this
extract
extract (month from start_time) or (year from start_time)
date truncate datetrunc()

 # Case
Case is basically an if stmt, except you don't need an elseif
select first_name, last_name, recommended_by,
  case
    when recommended_by is null then 'not recommended'
    else 'recommended'
  end as rec_status
from members;
In the above, I have named the case output col as rec_status

 # Join!!!
Sql allows us to join multiple tables together using unique id's and then we can use this data to manipulate
e.g.
select first_name || ' ' || last_name as member,
  facilities.name as facility, bookings.start_time
from bookings
join members
  on members.id = bookings.member_id
join facilities
  on facilities.id = bookings.facility_id;
For join statements, order of join doesn't matter

 # Sub Queries
e.g. select name, revenue
From companies
 where revenue > (select avg(revenue)
From companies);

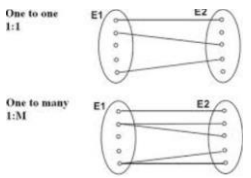
 # Creating an AWS DB
Very easy, just follow instructions on AWS. Make sure to make the DB access publicly if you want to allow others to access it. Ensure you update security settings
Creating roles to give read access for example:
Create role student;
Alter role student with password 'password';
Alter role student with login;
Grant select on attendance to student;

 # Dropping/Creating Tables
Drop table if exists table_name;
Make sure to not drop a table if you need it!! Easiest way to update table is to drop it and make updates and recreate it

create table climate (
ID text primary key,
STATION text,
ELEVATION real,
DATE date,
);
Can add more info to columns by adding not null etc to the above. Primary key is not necessary and HAS to be unique

 # Manipulating Existing Tables
renaming the tables and adding columns is the only updates one can make after the tables are created
Ofcourse, we can insert more data into an already created table by using INSERT, for example:
insert into movies (title, year)
values ('Black Panther', 2018);

 # Linking Tables
One to many and many to many (using primary keys)



 # modeanalytics
Great tool to help with data visualization (account already exists)

 # Saving a CSV
.mode csv (make sure this is the mode before saving)
.once filename
.once facilities.csv THEN write
select * from table_name;

SQL Cheat Sheet
Created By: Naveen Qureshi