MySQL- Structured Query Language to communicate with a database. Can add, remove, modify, search and join tables in database using code

**ERDs**

Using MySQL Workbench- it’s a great way to organize your tables and visualize your databases

-Open a new model

-Add new diagram

-Add a table for each data table you think you will need. Double click and try to follow this convention:

* Table name should be plural
* First column is ‘id’. It should be the primary key, AI (auto-increment) and be an integer (INT)
* Add any fields that are likely to go in there (name, password, email, address, etc). Select the appropriate datatype based on expected length of each entry
* Good practice to finish with ‘created\_at’ and ‘updated\_at’ columns of datatype DATETIME to be able to make timestamps
* Once each table has been added, connect them with relationship lines. If going one-to-many, use the single to branch icon selecting the many table first.
* If you have a situation where you are going many to many, you need an intermediary table with the manys branching off the intermediary and connecting to the ‘ones’.
* Finally, clean up the auto-generated column id tags that will be plural and singularize them

Remember the 3 forms of normalization:

1. **Each Column in your table can only have 1 value.**

Ex. You should not have an **address** column in your table that lists the address, city, state, and zip, all separated by commas.

2. **Each Column in your table that is not a *key (primary or foreign)* must have unique values.**

Ex. If you have a **movies** table with a **categories** column, you should not have a category repeated more than once.

3. **You cannot have a non-key column that is dependent on another non-key column.**

Ex. If you have a **books** table with columns **publisher\_name**and **publisher\_address,**the publisher\_address and publisher\_name should be separated into a separate table and linked to books with a foreign key. The publisher\_address is dependent on the publisher\_name and neither column is a key column.

**Importing a SQL file:**

Open up the file in Workbench. You should see all of the SQL commands in your file. Execute it with the Lightning Bold and refresh your list of schemas to see it and access it.

**MySQL Queries Syntax:**

**SELECT: used to generally query a database.**

SELECT \* FROM world (will pull up everything in the world database)

SELECT cities FROM world (will pull up the city column from the world table

SELECT \* FROM users WHERE first\_name LIKE "%e" OR “e%” OR ‘%e%’); (selects all users whose first\_name entry ends in e, starts with e, or contains an e)

SELECT # FROM users ORDER BY first\_name DESC or ASC (default is to list ascending (ASC) but you can input descending too)

**INPUT: used to insert data into a table. Can be done manually with code or using the GUI**

INSERT INTO table\_name (column\_name1, column\_name2)

VALUES('column1\_value', 'column2\_value');

**UPDATE: Can edit individual values or columns**

UPDATE table\_name SET column\_name1 = 'some\_value', column\_name2='another\_value' WHERE condition(s)

**IMPORTANT**: if **WHERE** condition is not added to the **UPDATE** statement, the changes will be applied to every record in the table.

**DELETE: Can delete individual or groups of records**

DELETE FROM table\_name WHERE condition(s)

**IMPORTANT**: if **WHERE** condition is not added to the **DELETE** statement, it will delete all the records on the table.

**Functions: There are a long list of functions to make your data behave in the way you want.**

Ex: SELECT CONCAT(‘Mr. ‘, first\_name, ‘ ‘, last\_name) FROM users; (will concatenate all user first and last names with spacing and a Mr in front.

Review for a list of ways to change the DATETIME output into a more readable output (just days, hour and minute, date, etc.)

**JOIN: can join table data by connecting the foreign IDs together**

If we wanted to join our clients first and last name to another billing table with amounts and dates:

*-2 table join*

SELECT clients.first\_name, clients.last\_name, billing.amount, billing.charged\_datetime (select our columns to join)

FROM clients (looks like you could also select FROM billing too)

JOIN billing ON clients.id = billing.clients\_id (joins primary key to foreign key)

*-join 3 tables together*

SELECT clients.first\_name AS client\_first, clients.last\_name, sites.domain\_name, leads.first\_name AS leads\_first(the columns we want, we edit the client first name and lead first name titles so first\_name isn’t listed twice)

FROM clients (our first table)

JOIN sites ON clients.id = sites.clients\_id (joins these two together with primary key and foreign key)

JOIN leads ON sites.id = leads.sites\_id (joins second tables with second table primary to 3rd table foreign)

-*left join:* in the previous examples, it will not pair columns if there is no info. To show all results, we can join our first table (left) to the second (right )

SELECT clients.first\_name, clients.last\_name, sites.domain\_name

FROM clients

LEFT JOIN sites ON clients.id=sites.clients\_id (joins our left to the right tables even if there isn’t something to join on right table)

-*group by: this can apply some function to group data better. Here we want to list a total billing amount for each client*

SELECT clients.first\_name, clients.last\_name, SUM(billing.amount) (applies a function on billing amount to add values, could be MIN(), MAX(), AVG() etc. )

FROM clients

JOIN billing ON clients.id = billing.clients\_id (joins primary key to foreign key)

GROUP BY clients.id (you will need some sort of function to condense the field)