# Introduction to MySQL

## SQL

* SQL is designed for working with relational databases
  + Most relational databases were created before SQL was standardized, so each database uses a slight variation on the standard for backwards compatibility
* SQL is whitespace insensitive—line breaks and indentation are often used to make complex statements more readable
* SQL parts
  + Statements: units of execution
    - Ends with ;
  + Clauses: part of a statement starting with a keyword???
  + Expression: a piece of SQL that evaluates to a value
    - Commonly used within clauses

## MySQL

h

# Create

## Create schemas/databases

H

h

## Create Tables

H

h

## Add Data to Tables

### Manually Add Records

INSERT INTO table (column 1, column 2) VALUES (value 1, value 2);

* The columns don’t need to be listed if they’re all having values added to them and the values are in the same order as the columns

# Read—Select Statements

## Select statements—rename when more structure found

SELECT \* FROM table;

* h

# Update

## Update Records

UPDATE table

SET changing column = new value

WHERE condition column = condition value;

* h

# Delete

## Deleting Info

h

## Deleting Tables

h

## Deleting Records

DELETE FROM table WHERE condition column = condition value;

* Delete statements are usually very similar to select statements except they begin with DELETE instead of SELECT and a list of columns
  + This allows delete statements to be previewed by crafting the statement with SELECT \* instead of DELETE
  + h

# From Database Clinic

CREATE TABLE table name (

Column name variable type NOT NULL AUTO\_INCREMENT,

Column name variable type,

PRIMARY KEY (column name));

LOAD DATA LOCAL INFILE ‘absolute path to CSV file, escaping all backslashes’

INTO TABLE table name

FIELDS TERMINATED BY ‘field delimeter’

ENCLOSED BY ‘field enclosing marks, usually quotations’

LINES TERMINATED BY ‘record delimeter’

IGNORE 1 LINES # if file contains headers

(@field1, @field2, @field3, @field4) # only needed if not loading all fields or fields not in same order in source data and table

SET column name=@field1, column name=@field3; # Field2 and field4 values won’t be imported

Note: @ signals variable name

Inner join:

SELECT

Table1.field,

AVG(table.field)

FROM table1

JOIN table2 ON table1.field=table2.field # can be repeated for multiple joins in a single query

WHERE table.field LIKE ‘string to match where % is a multi-character wildcard’

GROUP BY table.field #required if an aggregate function like AVG is used

ORDER BY table.field

AS is for aliases

UPDATE table name SET field = REPLACE(field, ‘old value’, ‘new value);

RLIKE allows for matching with regexes

SELECT

subquery.field1 AS ‘display name’,

table2.field2 AS ‘display name’,

SUM(an aggregated column)

FROM (SELECT field1, field2, SUM(field3) AS sum FROM table1 GROUP BY field1, field2) AS subquery

JOIN table2

ON table2.field1 = CASE # works as if-else statement

WHEN subquery.field = x THEN a value in table2.field1

WHEN subquery.field = y THEN a value in table2.field1

ELSE a value in table2.field1 END

GROUP BY non-aggregated columns;