- Basic Commands
    - Data Definition Language (DDL)
        - CREATE
            - Crete database, tables columns, indexes
        - ALTER
            - Add or remove columns, indexes, etc.
            - Ex: ALTER TABLE products
                ADD COLUMN coffee_origin VARCHAR(30);
        - DROP
            - Delete tables, indexes, or database
            - EX: DROP TABLE table;
        - RENAME
        - TRUNCATE
            - Clear out table contents
            - Ex: TRUNCATE TABLE table;
    - Data Manipulation Language (DML)
        - SELECT
            - Can use DISTINCT keyword to return rows non-repeatedly
            - Can use LIMIT N to return only N rows
            - Can use LIMIT N OFFSET M to return N rows offset from start by M
                - Or LIMIT M, N
        - INSERT
        - UPDATE
        - DELETE
    - Additional Commands
        - USE database;
        - SHOW TABLES;
        - DESCRIBE table;

- Data Types
    - Numeric
        - INT
        - FLOAT(p) – approximate
            - Automatically makes as precise as possible
        - DECIMAL(M,D) – precise
    - Non-Numeric
        - CHAR(N) – 255 character max
        - VARCHAR(N) – variable length
        - ENUM(list of accepted values)
        - BOOLEAN
    - Dates and Times
        - DATE(YYYY-MM-DD)
        - DATETIME(YYYY-MM-DD HH:MM:SS.ssssss)
        - TIME(HHH:MM:SS.ssssss)
        - YEAR(YYYY)
    - NULL
        - Indicate no value has been stored in column
        - Not the same as 0 or empty string
        - Can’t use =, <, > operators
            - IS NULL, IS NOT NULL, IFNULL() instead

- Changing Columns
    - Change Example
        - *Can rename and modify column in one command*
        - ALTER TABLE table
        - CHANGE cur_name new_name data_type;
    - Rename Example
        - ALTER TABLE pets
        - RENAME COLUMN animal_type TO species;
    - Modify Example
        - ALTER TABLE addresses
        - MODIFY COLUMN city CHAR(25)
- Data Manipulation Language
    - INSERT Example
        - INSERT INTO products (name, price, coffee_origin)
        - VALUES ('Espresso', 2.50, 'Brazil');
    - UPDATE Example
        - UPDATE products
        - SET coffee_origin = ‘Sri Lanka’
        - WHERE id = 7;
        - *sql_safe_updates* = *true* means we can use only key columns in WHERE clause
    - DELETE Example
        - DELETE from people
        - WHERE gender = ‘F’;
- Equality Operators
    - !=
    - <> (not equal to)
    - >
    - >=
    - <
    - <=

- SELECT with a Single Table
  - Multiple Values Inclusive Example
    - SELECT * FROM customers
    - WHERE last_name IN ('Taylor', 'Bluth', 'Armstrong');
  - Exclusive Example
    - SELECT * FROM customers
    - WHERE first_name NOT IN ('Katie, 'George, 'John');
  - BETWEEN Example
    - SELECT * FROM orders
    - WHERE order_time BETWEEN '2023-01-01' AND '2023-01-31 23:59:59';
    - *Default time midnight if not specified*
  - BETWEEN Example 2
    - SELECT * FROM customers
    - WHERE last_name BETWEEN 'A' AND 'L';
    - *Range inclusive, but will not return values after last value*
      - Ex: last names starting with L in this example
  - LIKE Example 1
    - SELECT * FROM customers
    - WHERE last_name LIKE 'W%';
    - *Returns all customers whose last name begins with* 'W'
  - LIKE Example 2
    - SELECT * FROM customers
    - WHERE last_name LIKE '%o%';
    - *Returns all customers whose last name contains* 'o'
  - LIKE Example 3
    - SELECT * FROM customers
    - WHERE first_name LIKE '_o__';
    - *In example, returns* 'John' *and* 'Toby'
    - *Underscore matches only a single character at position*
  - ORDER BY Example
    - SELECT * FROM products
    - ORDER BY price ASC;
    - *Can do* DESC *instead of* ASC (default)
    - NULL *values appear first in* ASC *order*
  - ORDER BY Example 2
    - SELECT * FROM customers
    - ORDER BY last_name, first_name;
    - ORDER BY *is usually last in SQL query (but before LIMIT)*
  - Alias Example
    - SELECT id, name AS coffee, price, coffee_origin AS country FROM products;

- SELECT and JOIN with Multiple Tables
    - INNER JOIN – retrieves data only when there are matching values in both tables
    - LEFT JOIN – retrieves all data from first table and matching rows from second table
    - RIGHT JOIN – retrieves all data from second table and matching rows from first table
    - OUTER JOIN – retrieves all data from both first and second tables
        - *Not in MySQL, rare in practice*
- JOIN Examples
    - INNER JOIN Example
        - SELECT products.name, orders.order_time FROM orders
        - INNER JOIN products ON orders.product_id = products.id;
        - orders *is table 1,* products *is table 2*
    - Alternate INNER JOIN Example with Aliases
        - SELECT p.name, o.order_time FROM orders AS o
        - JOIN products p ON o.product_id = p.id;
    - LEFT JOIN Example
        - SELECT o.*, c.* FROM orders o
        - LEFT JOIN customers c ON o.customer_id = c.id
        - ORDER BY o.order_time;
        - *Returns results where* o.customer_id *is* NULL*, whereas* RIGHT JOIN *will not*
        - *Recommended to only use* LEFT JOIN *as some systems do not support* RIGHT JOIN
    - Joining Three Tables Example
        - SELECT p.name, p.price, c.first_name, c.last_name, o.order_time FROM products p
        - JOIN orders o ON p.id = o.product_id
        - JOIN customers c ON c.id = o.customer_id
        - ORDER BY o.order_time;