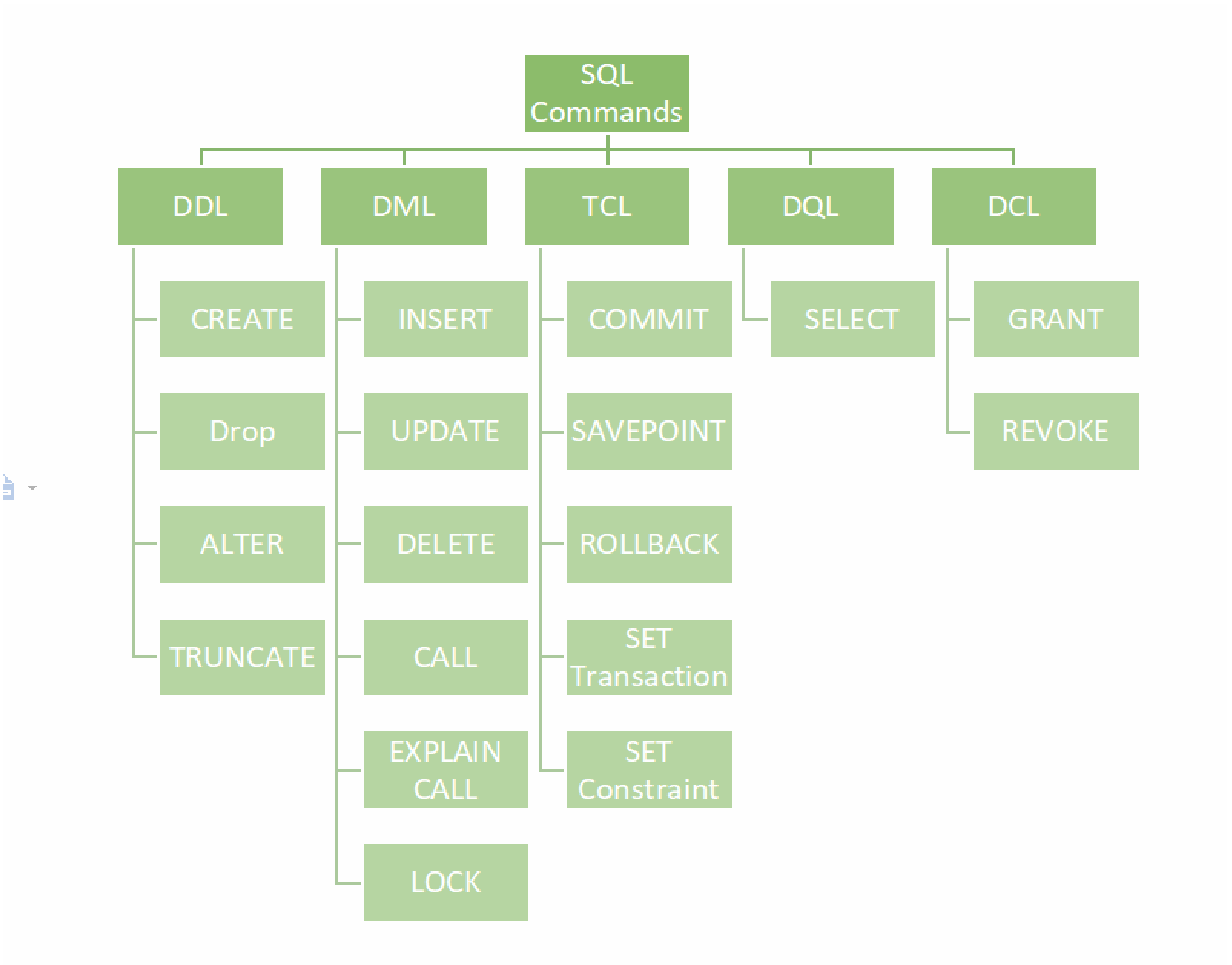
Practical 3: - DDL, DML, DCL queries



Steps to Install MySQL

1. **Install MySQL**: Get the installer from the official MySQL website: https://dev.mysql.com/downloads/
2. **Start MySQL Server**: After the installation is complete, you need to start the MySQL server. On Windows, this is usually done automatically during installation, and the MySQL service will be running in the background.
3. **Launch MySQL Command-Line Client**: To interact with MySQL, you can use the MySQL command-line client (also known as MySQL shell or mysql client). It allows you to execute SQL commands directly.

- On Windows: Open the Command Prompt or PowerShell on your PC. Type **mysql -u root -p** and press Enter. It will prompt you to enter the root password for MySQL (if you set one during installation).
- On macOS and Linux: Open the Terminal application. Type **mysql -u root -p** and press Enter. It will prompt you to enter the root password for MySQL (if you set one during installation).

4. **MySQL Command-Line Client is Open**: Once you enter the correct password, you should see the MySQL command-line prompt, which looks like **mysql>**.

➢ DML (Data Manipulation Language) queries

1. SELECT Query:

Syntax:

```
SELECT column1, column2, ... FROM table_name WHERE condition;
```

Example:

```
SELECT name, salary FROM employees WHERE salary > 50000;
```

Output:

```
+-----------+--------+
| name      | salary |
+-----------+--------+
| John      | 60000  |
| Emily     | 55000  |
| Michael   | 52000  |
+-----------+--------+
```

2. INSERT Query:

Syntax:

```
INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
```

Example:

```
INSERT INTO employees (name, salary) VALUES ('Sarah', 58000);
```

3. UPDATE Query:

Syntax:

```
UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;
```

Example:

```
UPDATE employees SET salary = 54000 WHERE name = 'Michael';
```

4. DELETE Query:

Syntax:

```
DELETE FROM table_name WHERE condition;
```

Example:

```
DELETE FROM employees WHERE name = 'Emily';
```

- **DDL (Data Definition Language) queries**
-

1. **CREATE TABLE Query:**

Syntax:

```
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
);
```

Example:

Let's create a simple "users" table with columns "id," "name," "email," and "age":

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    age INT
);
```

Output:

If the query is successful, no output will be displayed. You can check the table structure using the `DESCRIBE` command:

```
DESCRIBE users;
```

Sample Output:

```
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(50)  | NO   |     | NULL    |                |
| email | varchar(100) | YES  | UNI | NULL    |                |
| age   | int          | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
```

2. **ALTER TABLE Query:**

Syntax:

```
ALTER TABLE table_name
ADD COLUMN new_column datatype constraints;
```

Example:

Let's add a new column "phone" to the "users" table:

```
```

ALTER TABLE users

ADD COLUMN phone VARCHAR(15);

```

Output:

If the query is successful, no output will be displayed. You can check the updated table structure using the `DESCRIBE` command.

**3. \*\*DROP TABLE Query\*\*:**

Syntax:

```

DROP TABLE table_name;

Example:

Let's drop the "users" table:

```

*DROP TABLE users;*

***CASE STUDY***

***Database Name: `bookstore_db`***

Tables:

1. `books`: To store information about books.

- Columns: `book_id` (Primary Key), `title`, `author`, `price`, `quantity`

2. `customers`: To store information about customers.

- Columns: `customer_id` (Primary Key), `name`, `email`, `phone`

3. `orders`: To store information about book orders.
```

- Columns: `order_id` (Primary Key), `customer_id` (Foreign Key referencing the `customers` table), `book_id` (Foreign Key referencing the `books` table), `order_date`, `quantity`

1. **Create the Database**:

First, create the `bookstore_db` database:

```sql
CREATE DATABASE bookstore_db;
```

2. **Use the Database**:

Use the newly created database for further operations:

```sql
USE bookstore_db;
```

3. **Create Tables**:

Now, create the tables as per the mentioned requirements:

```sql
CREATE TABLE books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(50) NOT NULL,
    price DECIMAL(8, 2) NOT NULL,
    quantity INT NOT NULL
);
```

~~~
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL
);

CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    book_id INT,
    order_date DATE NOT NULL,
    quantity INT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES
customers(customer_id),
    FOREIGN KEY (book_id) REFERENCES books(book_id)
);
```
~~~

4. **Insert Data:**

Insert some sample data into the tables:

~~~
```sql
INSERT INTO books (title, author, price, quantity)
VALUES
    ('The Great Gatsby', 'F. Scott Fitzgerald', 12.99, 50),
~~~

```
    ('To Kill a Mockingbird', 'Harper Lee', 10.75, 40),
    ('1984', 'George Orwell', 9.99, 30);

INSERT INTO customers (name, email, phone)
VALUES
    ('John Doe', 'john@example.com', '123-456-7890'),
    ('Jane Smith', 'jane@example.com', '987-654-3210');

INSERT INTO orders (customer_id, book_id, order_date, quantity)
VALUES
    (1, 1, '2023-08-02', 2),
    (1, 2, '2023-08-01', 1),
    (2, 3, '2023-08-02', 3);
```

5. **Retrieve Data**:

Now, you can retrieve data using SELECT queries:

```sql
-- Retrieve all books
SELECT * FROM books;

-- Retrieve all customers
SELECT * FROM customers;

-- Retrieve all orders
SELECT * FROM orders;
```

6. **Update Data**:

You can update existing data using UPDATE queries:

```sql
-- Update book price
UPDATE books
SET price = 14.99
WHERE book_id = 1;

-- Update customer phone number
UPDATE customers
SET phone = '555-555-5555'
WHERE customer_id = 1;
```

7. **Delete Data**:

To remove data, use DELETE queries:

```sql
-- Delete a book from the books table
DELETE FROM books
WHERE book_id = 3;

-- Delete a customer from the customers table
DELETE FROM customers
WHERE customer_id = 2;
```

SYNTAX FOR FOREIGN KEY

```
CREATE TABLE table_name (
column1 datatype PRIMARY KEY,
column2 datatype,
column3 datatype,
...
FOREIGN KEY (foreign_key_column) REFERENCES
        parent_table_name(parent_key_column)
);
```

```
CREATE TABLE Customers (
   customer_id INT PRIMARY KEY,
   customer_name VARCHAR(50),
   customer_email VARCHAR(100)
);

CREATE TABLE Orders (
   order_id INT PRIMARY KEY,
   order_date DATE,
   order_amount DECIMAL(10, 2),
   customer_id INT, -- This column will hold the foreign key
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```