# UDSQL Project Documentation

Development Team

February 2025

## 1 Introduction

The UDSQL project is a database management system with a command-line interface (CLI). This system allows users to interact with databases and perform operations such as creating, inserting, querying, updating, deleting, and dropping databases and tables.

## 2 Objectives

The goal of this project is to implement a simple system for managing databases using basic SQL commands via a CLI.

## 3 Project Structure

The project consists of the following modules:

- **dbms/database.py** - Manages database operations like creating and dropping databases.
- **dbms/parser.py** - Parses SQL-like commands and translates them into executable structures.
- **dbms/executor.py** - Executes parsed commands.
- **dbms/exceptions.py** - Handles custom exceptions.
- **main.py** - CLI interface to interact with the database.
- **file_manager.py** - Manages creation, update and delete of files csv.
- **table.py** - Manages table operations like updating, inserting and deleting data.

## 4 Running the Project

### 4.1 Prerequisites

Before running the project, ensure that you have the following installed:

- Python 3.x
- Git (optional, for cloning the repository)

### 4.2 Installation and Setup

1. Clone the GitHub repository (if not already downloaded):

```
git clone https://github.com/Daniel-Chavarro/DBMS-Workshop.git
cd DBMS-Workshop
```

2. Install dependencies (if required):

```
pip install -r requirements.txt
```

(If no requirements file is provided, ensure Python is installed.)

### 4.3 Running the UDSQL CLI

To start the database management system, run:

```
python CLI/main.py
```

You should see a prompt like:

```
Welcome to UDSQL
Available databases:

If want to create a new database, type the name of the database.
If you want to exit, type 'exit'
Enter the database name:
```

Notice that inserting the name of the database will automatically create or login into the database, and use it as an environment.

## 5 System Features

The following SQL commands are supported:

- `CREATE TABLE <table_name> col1 type col2 type PRIMARY_KEY col FOREING_KEY(Optional) col1` - Creates a table.
- `INSERT INTO <table_name> VALUES val1 val2, ...` - Inserts data.
- `SELECT * FROM <table_name>` - Retrieves data.
- `UPDATE <table_name> SET col1=val WHERE condition` - Updates data.
- `DELETE FROM <table_name> WHERE condition` - Deletes records.
- `DROP TABLE <table_name>` - Drops a table.
- `DROP DATABASE` - Deletes a database.
- `EXIT` - Exits the CLI.

## 6 Step-by-Step Guide

### 6.1 Creating a Database

To create a database, input the name of the database you wanna create into the startup screen. For example:

```
Welcome to UDSQL
Available databases:

If want to create a new database, type the name of the database.
If you want to exit, type 'exit'
Enter the database name: my_database
```

### 6.2 Supported Data Types

UDSQL supports three fundamental data types:

- **int** - Represents integer values (whole numbers).
- **float** - Represents floating-point numbers (decimals).
- **str** - Represents string values (alphanumeric characters).

## 6.3 Creating a Table

To create a table within the database:

```
CREATE TABLE users id int name str age int PRIMARY_KEY id;
```

## 6.4 Inserting Data

To insert a record into the table:

```
INSERT INTO users VALUES 1 Alice 25 New York
INSERT INTO users VALUES 2 Bob 30 Los Angeles
INSERT INTO users VALUES 3 Charlie 22 Chicago
INSERT INTO users VALUES 4 David 28 New York
INSERT INTO users VALUES 5 Emma 35 San Francisco
```

## 6.5 Managing Conditions

Conditions in UDSQL follow Python-style conditional expressions, additionally the value to be compared must be in single quotes if is str. Below is a breakdown of supported conditions and how they work.

### 6.5.1 Equality and Inequality

```
SELECT * FROM users WHERE age == 25;  // Equal to
SELECT * FROM users WHERE city != 'New York';  // Not equal to
```

### 6.5.2 Comparison Operators

```
SELECT * FROM users WHERE age > 25;  // Greater than
SELECT * FROM users WHERE age >= 30;  // Greater than or equal to
SELECT * FROM users WHERE age < 30';  // Less than
SELECT * FROM users WHERE age <= 22;  // Less than or equal to
```

### 6.5.3 Logical Operators

```
SELECT * FROM users WHERE age > 25 and city == 'New York'  // AND condition
SELECT * FROM users WHERE age < 30 or city == 'Los Angeles'  // OR condition
```

### 6.5.4 Using NOT Operator

```
SELECT * FROM users WHERE not city == 'Chicago'
```

## 6.6 Querying Data

To retrieve all records:

```
SELECT * FROM users;
```

### 6.6.1 Selecting Specific Columns

To retrieve only the names and ages of users:

```
SELECT name, age FROM users
```

#### 6.6.2 Filtering Data with WHERE

To get users who live in New York:

```
SELECT * FROM users WHERE city = 'New York'
```

#### 6.6.3 Using Comparison Operators

To find users older than 25:

```
SELECT * FROM users WHERE age > 25
```

#### 6.6.4 Combining Conditions

To find users older than 25 who live in New York:

```
SELECT * FROM users WHERE age > 25 and city == 'New York'
```

### 6.7 Updating Data

To modify a record:

```
UPDATE users SET age = 26 WHERE id == 1
```

### 6.8 Deleting Data

To remove a record:

```
DELETE FROM users WHERE id == 2
```

### 6.9 Dropping Tables and Databases

To remove a table:

```
DROP TABLE users
```

To delete a database:

```
DROP DATABASE;
```

As you can see, the name of the database is not needed because the CLI works into the database until you restart the app.

## 7 Parser and Command Execution

The parser processes commands and sends them to the executor, which performs the necessary operations.

## 8 Exception Handling

The system includes custom exception handling, such as raising a `DroppedDatabaseError` if a non-existent database is accessed.

## 9 Conclusion

The UDSQL system provides an efficient CLI-based method to manage databases with fundamental SQL commands.