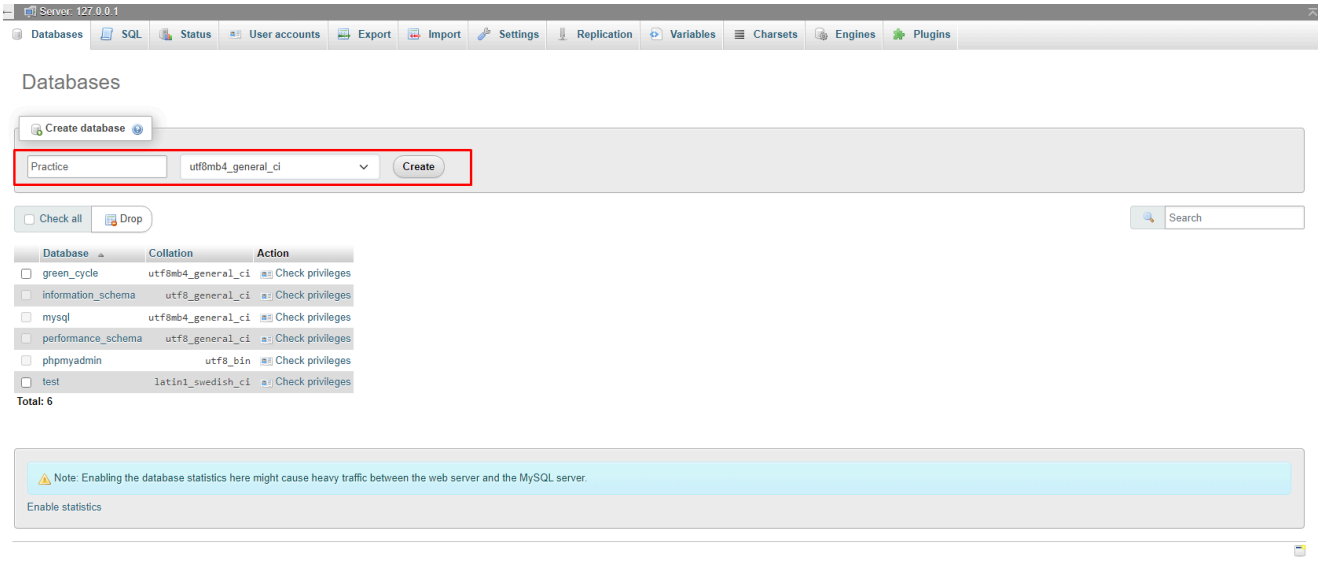
MySQL Fundamentals with XAMPP

Structured Query Language (SQL) is a programming language that allows users to interact with relational databases to store, retrieve, modify, and analyze data.
MySQL is a relational database management system (RDBMS) that stores data for software applications.

Creating Database:

Manually:

Go to New -> Provide name -> Create



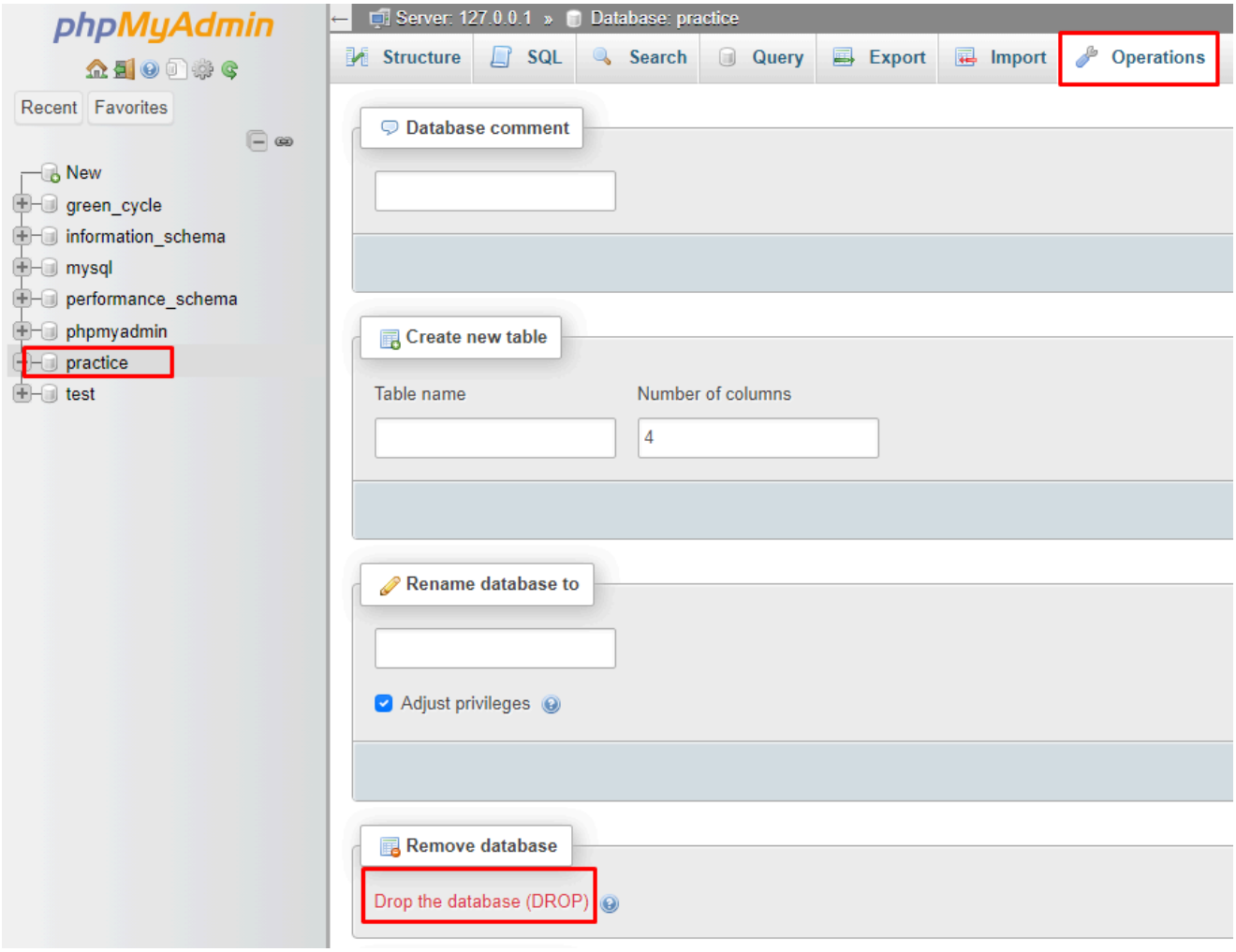
With SQL Syntax:

```
CREATE DATABASE practice;
```

Dropping/Deleting Database:

Manually:

Go to the database -> Operations -> Drop the database (DROP)



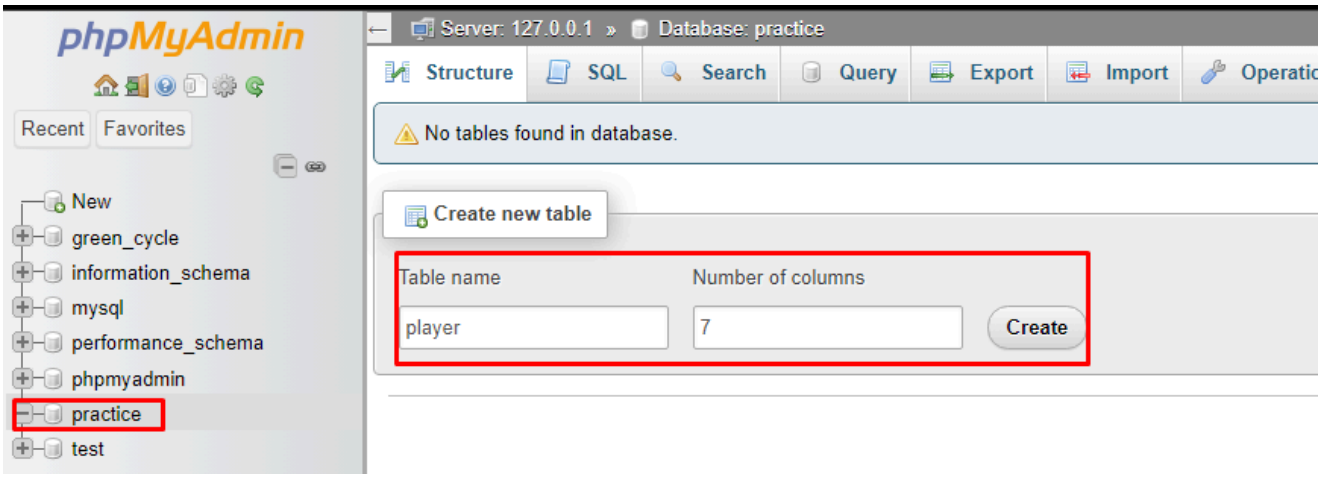
With SQL Syntax:

```
DROP DATABASE practice;
```

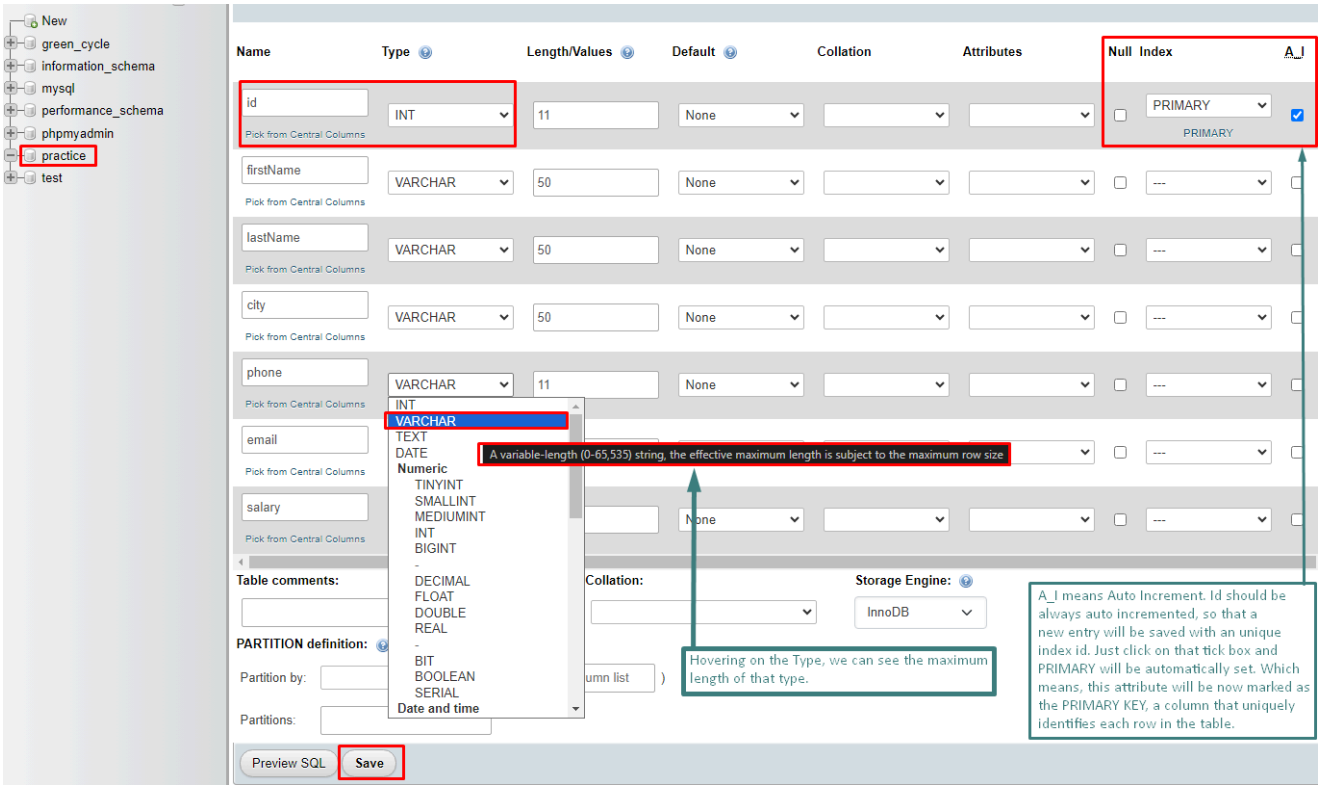
Table Creation:

Manually:

Go to the database -> Provide Table name -> Number of columns needed -> Create



Set the column names, types, A_I etc.



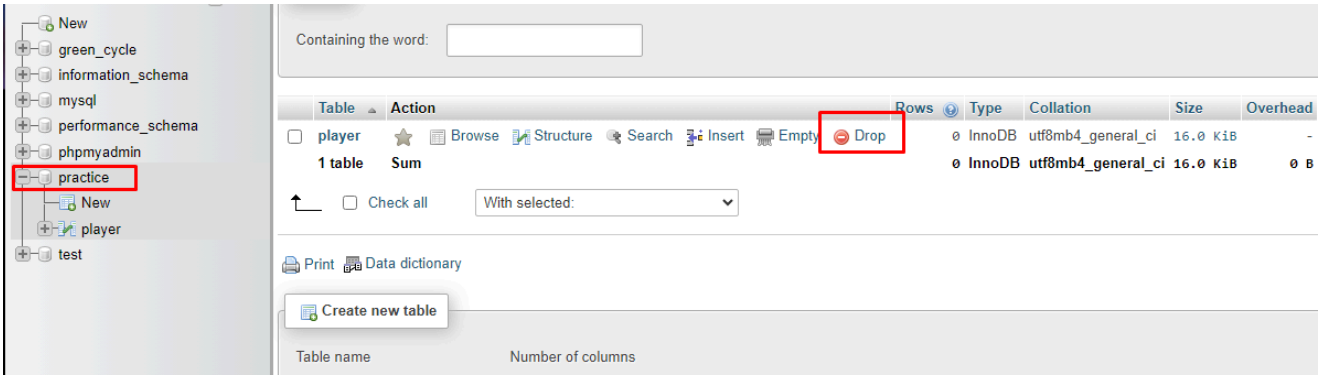
With SQL syntax:

```
CREATE TABLE player (
    id INT(11) NOT NULL AUTO_INCREMENT,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
    phone VARCHAR(11) NOT NULL,
    email VARCHAR(50) NOT NULL,
    salary FLOAT NOT NULL,
    PRIMARY KEY (id)
);
```

Dropping/Deleting Table:

Manually:

Click on the Database -> Click Drop

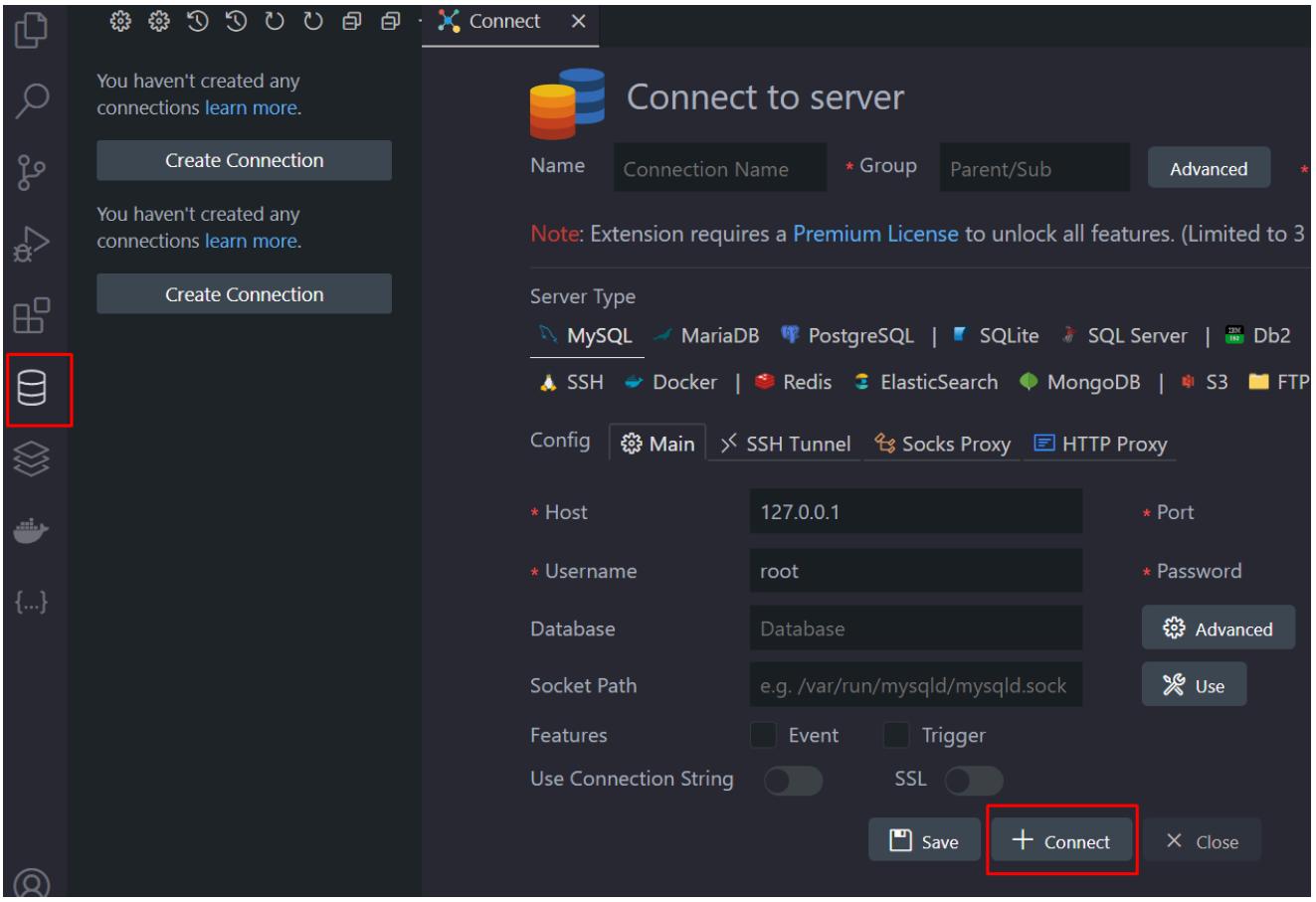


With SQL Syntax:

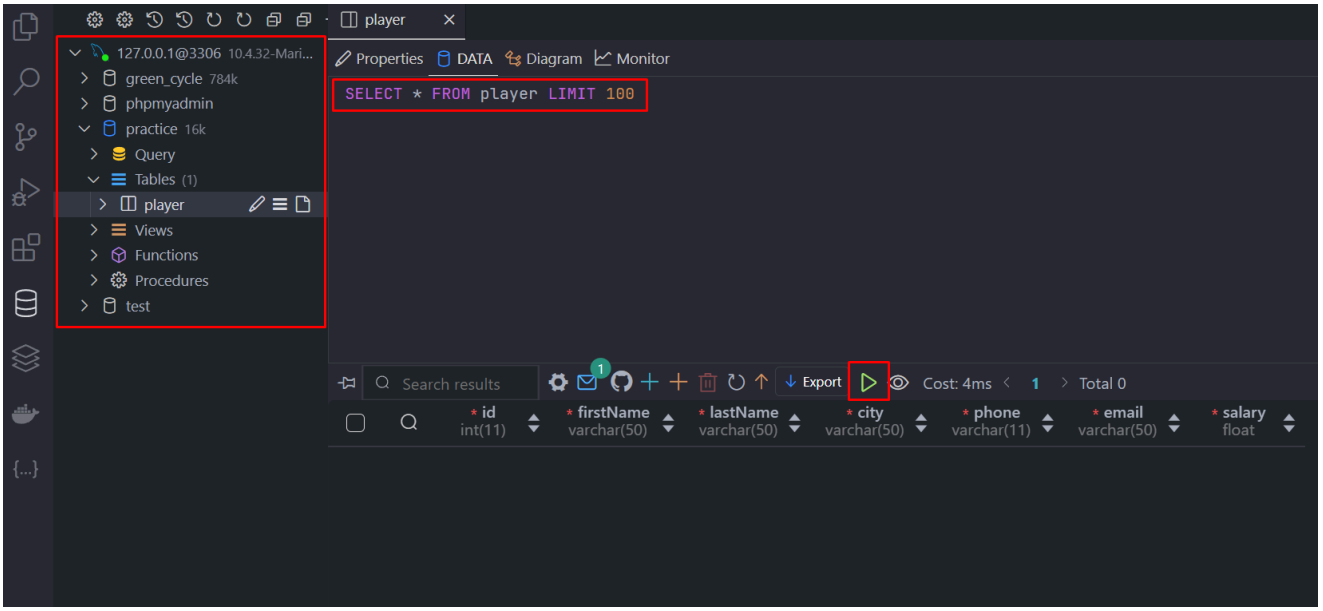
```
DROP TABLE player;
```

Running Database Operations with VS Code:

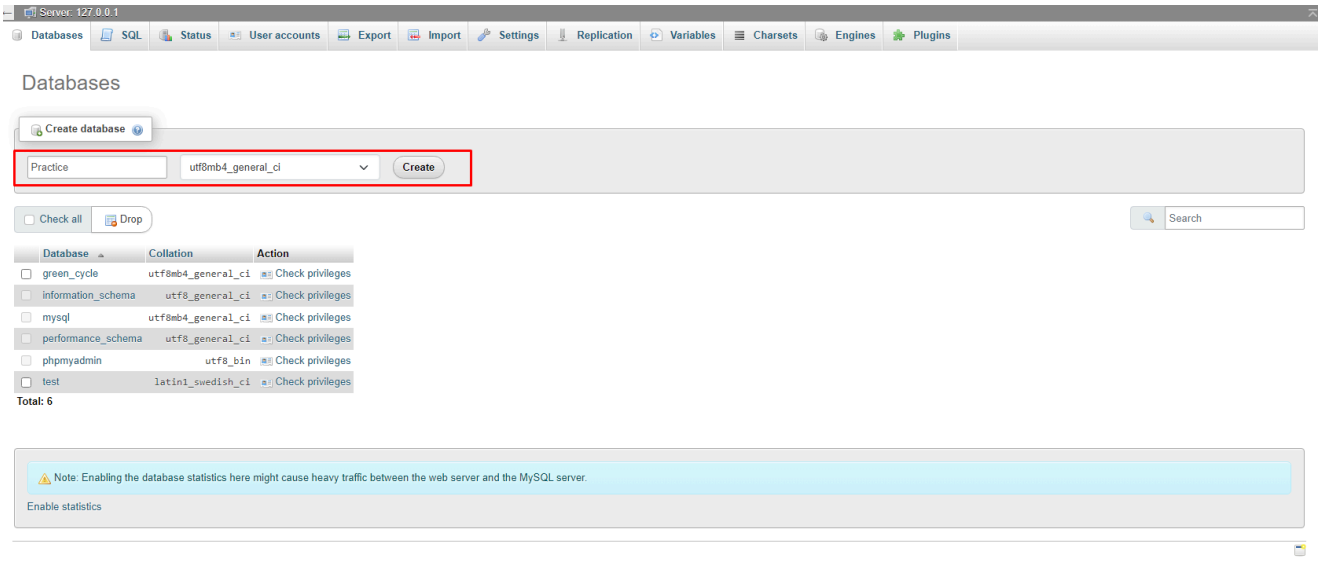
Just run the XAMPP (Apache and MySQL), install the extension in VS Code 'Database Client' -> Go to the extension -> click Create Connection -> Connect



Connection will be established and we can see all the databases, we will also get the writing pad to write our SQL queries and run those.



We can also create a random new file with an extension of .sql and then we can run our queries there.



MySQL Operations:

INSERT (Inserting data in the tables):

```
INSERT INTO
    player (
        firstName,
        lastName,
        city,
        phone,
        email,
        salary
    )
VALUES (
        'John',
        'Doe',
        'New York',
        '1234567890',
        'jdoe@me.com',
        5000.00
    );

INSERT INTO
    player (
        firstName,
        lastName,
        city,
        phone,
```

```
        email,
        salary
    )
VALUES (
        'Jane',
        'Doe',
        'New York',
        '34234424243',
        'janedoe@me.com',
        4000.00
    );
```

DELETE (Deleting a data):

```
DELETE FROM player WHERE id = 2;
-- Jane Doe will be deleted
```

UPDATE SET (Update a data):

```
UPDATE player SET salary = 6000.00, city = 'London' WHERE id = 1;
-- update salary and city for Jane Doe to 6000.00 and London respectively
```

SELECT ALL and SELECT with WHERE clause (Selecting specific data):

```
SELECT * FROM player;
-- SELECT all (*) data

SELECT * FROM player WHERE city = 'London';
-- SELECT all (*) data where city is London
```

SELECT for sorting to ASCENDING/DESCENDING order (ORDER BY) :

```
SELECT * FROM player ORDER BY salary ASC;
-- SELECT all (*) data where salary is in ascending order

SELECT * FROM player ORDER BY salary DESC;
-- SELECT all (*) data where salary is in descending order
```

SELECT with limit (LIMIT):

```
SELECT * FROM player LIMIT 1;
-- SELECT all (*) data where limit is 1
```

SELECT for grouping (GROUP BY):

```
SELECT city FROM player GROUP BY city;
-- SELECT the cities. GROUP BY clause is used to group the data. Duplicates will
be removed.
```

SELECT for finding unique data (DISTINCT):

```
SELECT DISTINCT city FROM player;
-- SELECT the cities. DISTINCT clause is used to remove duplicates.
```

SQL Aggregate function (MAX):

```
SELECT MAX(salary) FROM player;
-- SELECT the maximum salary

SELECT * FROM player WHERE salary = ( SELECT MAX(salary) FROM player );
-- SELECT the column where salary is maximum
```

SQL Aggregate function (MIN):

```
SELECT MIN(salary) FROM player;
-- SELECT the minimum salary

SELECT * FROM player WHERE salary = ( SELECT MIN(salary) FROM player );
-- SELECT the columns where salary is minimum
```

SQL Aggregate function (AVG):

```
SELECT AVG(salary) FROM player;
-- Provide the average salary
```

SQL Aggregate function (SUM):

```
SELECT SUM(salary) FROM player;
-- Provide the sum of salary
```

SQL Aggregate function (COUNT):

```
SELECT COUNT(*) FROM player;
-- Provide the number of rows
```

AS (alias):

```
SELECT MAX(salary) AS Highest FROM player;
-- SELECT the maximum salary with the header 'Highest'

SELECT MIN(salary) AS Lowest FROM player;
-- SELECT the minimum salary with the header 'Lowest'

SELECT AVG(salary) AS Average FROM player;
-- Provide the average salary with the header 'Average'

SELECT SUM(salary) AS Total_Salary FROM player;
-- Provide the sum of salary with the header 'Total_Salary'

SELECT COUNT(*) AS Total_Rows FROM player;
-- Provide the number of rows with the header 'Total_Rows'
```