

Inserting a single row, specifying both column names and values:

This method explicitly lists the columns you are providing values for, making it clear and less prone to errors if the table structure changes or if you are not inserting into all columns.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO Customers (CustomerID, CustomerName, ContactName, City)
VALUES (92, 'Example Company', 'John Doe', 'New York');
```

Inserting a single row, providing values for all columns in order:

This method is shorter but requires you to know the exact order of columns in the table and provide a value for every column.

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

-- Assuming 'Customers' table columns are CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country in that order

```
INSERT INTO Customers VALUES (93, 'Another Company', 'Jane Smith', '123 Main St', 'Los
Angeles', '90210', 'USA');
```

Inserting multiple rows in a single statement:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES
  (value1_row1, value2_row1, value3_row1, ...),
  (value1_row2, value2_row2, value3_row2, ...),
  (value1_row3, value2_row3, value3_row3, ...);
```

```
INSERT INTO Products (ProductID, ProductName, Price)
```

VALUES

```
  (101, 'Laptop', 1200.00),
  (102, 'Mouse', 25.00),
  (103, 'Keyboard', 75.00);
```

Important considerations:

Data Types: Ensure the values you provide match the data types of the respective columns (e.g., strings in quotes, numbers without quotes).

NULL Values: If a column allows NULL values and you are not providing a value for it, you can either omit the column from the INSERT INTO statement (when specifying columns) or explicitly use NULL as the value.

Primary Keys: If a primary key column is auto-incrementing, you typically do not need to provide a value for it; MySQL will automatically assign one.

Renaming a Table:

Using RENAME TABLE:

```
RENAME TABLE old_table_name TO new_table_name;
```

To rename multiple tables in a single statement:

```
RENAME TABLE table1 TO new_table1, table2 TO new_table2;
```

Using ALTER TABLE:

```
ALTER TABLE old_table_name RENAME TO new_table_name;
```

Renaming a Column:

```
ALTER TABLE table_name RENAME COLUMN old_column_name TO new_column_name;
```

Modify Data type:

```
ALTER TABLE table_name CHANGE COLUMN old_column_name new_column_name  
new_data_type;
```

ALTER TABLE - ADD Column:

```
ALTER TABLE table_name ADD column_name datatype;
```

ALTER TABLE - DROP COLUMN:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Changing Constraints:

Adding Constraints:

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
```

Removing Constraints:

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Update Query:

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

```
UPDATE Customers SET City = 'New York' WHERE CustomerID = 101;
```

Delete Query:

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

```
DELETE FROM employees WHERE employee_id = 101;
```