# Altering Table Definitions

After we have created a table we can alter it with the ALTER TABLE command.

The grammar for ALTER TABLE is 121 lines long. Below are the parts of the grammar that we'll find useful for our purposes. You can view the complete documentation for ALTER TABLE in section 13.1.9 ALTER TABLE Statement of the MySQL documentation.

```
ALTER TABLE tbl_name
    [alter_option [, alter_option] ...]

alter_option: {
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD [CONSTRAINT] PRIMARY KEY
        (key_part,...)
  | ADD [CONSTRAINT] UNIQUE
        (key_part,...)
  | ADD [CONSTRAINT] FOREIGN KEY
        (col_name,...)
        reference_definition

  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP FOREIGN KEY fk_symbol

  | ALTER [COLUMN] col_name {
        SET DEFAULT {literal | (expr)}
      | DROP DEFAULT
    }

  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST | AFTER col_name]

  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]

  | RENAME COLUMN old_col_name TO new_col_name

  | RENAME [TO | AS] new_tbl_name
}
```

As you notice, the syntax for many of the permissible altercations is similar to clauses of the CREATE TABLE statement. In particular the grammar for key_part, column_definition and reference_definition are the same as in CREATE TABLE.

Multiple ADD, ALTER, CHANGE, AMD DROP clauses are permitted in a single ALTER TABLE statement.

The word COLUMN is optional except when renaming a column.

In the examples below I represent single ALTER statements on multiple lines so that they are easier to read. We often write them on a single line when executing them.

Let's now suppose we've created a table using the following CREATE TABLE statement.

```
USING MyGame_DB;

CREATE TABLE Users_T(
        userId INT AUTO_INCREMENT PRIMARY KEY,
        userName VARCHAR(32),
        countryCode CHAR(2)
);
```

---

## Adding Columns

We can add a new column to the table with an ADD COLUMN clause.

```
ALTER TABLE User_T
        ADD COLUMN highScore INT DEFAULT 0;
```

By default, when adding a column, the column is added as the last column. We can instruct the engine to add the column at the beginning of the table using FIRST, or after an existing column using AFTER col_name.

```
ALTER TABLE User_T
        ADD COLUMN highScore INT DEFAULT 0 FIRST;

ALTER TABLE User_T
        ADD COLUMN highScore INT DEFAULT 0 AFTER lastName;
```

---

## Dropping Columns

We can remove a column using a DROP COLUMN clause. We cannot however drop a column if the table contains only one column.

```
ALTER TABLE User_T
        DROP COLUMN countryCode;
```

---

## Renaming, Redefining, and Reordering Columns

The CHANGE, MODIFY, RENAME, and ALTER clauses have different abilities.

**CHANGE** (column name and definition)

- Can rename a column name and change its definition or both.
- Must provide the column definition even if only changing the name of the column.

**MODIFY** (column definition)

- Can change a column definition but not the column name.

**RENAME COLUMN** (column name)

- Can change the column name but not the column definition.

**ALTER** (column default value)

- Can only change the column default value.

---

## Renaming Tables

We can rename the table using the RENAME TO new_table_name clause or by using a RENAME TABLE statement. Documentation on the latter statement can be found in section 13.1.36 RENAME TABLE Statement.