More about the Primary Key Constraint:

Uniqueness → means that the value of the primary key must be unique; no two rows in the table can have the same value for the primary key field(s); duplicate values are not allowed for the primary key field(s)

Minimality → means that the primary key will consist of the minimum number of fields that it takes to uniquely identify each instance of an entity

CREATE TABLE

To create a table, use the keywords CREATE TABLE followed by the name that you want to give to the table. Include an open parenthesis and a close parenthesis after the table name. In between the parentheses, list the names of the columns (i.e. attributes, fields) followed by the data type for the column.

Info about the data types in MySql:

The most widely used data types are: DOUBLE, INT, VARCHAR, CHAR and DATE. Numerical primary keys are typically listed as either INT or DOUBLE.

VARCHAR and CHAR are used for columns that can store alpha-numeric characters (i.e. letters or combination of letters and numbers). Both data types take one parameter. Use the CHAR data type to specify the exact number of characters that a column MUST contain. Use the VARCHAR data type to specify the maximum number of characters that a column may contain. For example, a column of the type CHAR(5) MUST contain exactly 5 characters (no more and no less). A column of the type VARCHAR(10) can contain 1 character, 10 characters or anything in between, but it cannot store more than 10 characters.

After you specify the data type, you can optionally specify the keywords NOT NULL to indicate that the column cannot contain a NULL value. The default is that a column can contain a NULL value.

After listing the data type, use a comma to separate the fields. Do not use a comma after specifying the last item before the closing parenthesis.

After specifying the fields and data types, list the constraints. You should create constraints using the keyword CONSTRAINT. The naming convention for constraints is that the name should begin with the name of the table followed by an underscore and then an appropriate suffix (more about about the suffix in a moment). It is ALWAYS best to give a specific name to the constraints, otherwise the database management system will give the constraint a generic name…a generic name which will not help you to readily identify the table to which the constraint pertains.

You MUST end each SQL statement with a semicolon (;).

Columns are stored by the sequence in which they are listed in the CREATE TABLE command.

CONSTRAINTS…

Constraint names MUST be unique. So, be sure to use the naming conventions discussed below.

PRIMARY KEY CONSTRAINT…

A primary key is used to uniquely identify each record in the table. To create a primary key constraint, list the keyword CONSTRAINT followed by the name of the table, an underscore and the suffix PK (for primary key). Then use the keywords PRIMARY KEY followed by an open parenthesis, the name of the field(s) that make up the primary key and a closed parenthesis. For example,

```
CREATE TABLE Student (
        sid             INT,
        firstName       VARCHAR(15)  NOT NULL,
        lastName        VARCHAR(20)  NOT NULL,
        login           VARCHAR(7),
        dob             DATE,
        CONSTRAINT student_PK PRIMARY KEY (sid));
```

In the example above, sid is the primary key. If the primary key was a composite primary key (i.e. a primary key that consists of more than one column), then indicate all of the columns that make up the primary key, separated by commas, inside of the parentheses after the keywords PRIMARY KEY. For example, CONSTRAINT tablename_PK PRIMARY KEY(column1, column2).

There can only be one primary key per table!

FOREIGN KEY CONSTRAINTS…

To create a foreign key constraint: start with the keyword CONSTRAINT followed by the table name, an underscore, FK (for foreign key) and a sequential number. Then use the keywords FOREIGN KEY followed by a set of parentheses. Inside those parentheses is the name of the column in the table that you are creating that is a foreign key. Then you must specify the keyword REFERENCES followed by the name of the table that contains the field that your foreign key is referencing. Then you must specify a set of parentheses in which you list the specific name of the field in the table that is being referenced. For example,

```
CREATE TABLE Major (
                    majorID             INT,
                    majorName           VARCHAR(15)  NOT NULL,
                    CONSTRAINT major_PK PRIMARY KEY (majorID))
CREATE TABLE Student (
                    sid             INT,
                    firstName       VARCHAR(15)  NOT NULL,
                    lastName        VARCHAR(20)  NOT NULL,
                    login           VARCHAR(7),
                    dob             DATE,
                    majorID         INT,
                    CONSTRAINT student_PK PRIMARY KEY (sid),
                    CONSTRAINT student_FK1 FOREIGN KEY (majorID) REFERENCES
Major(majorID));
```

IMPORTANT NOTES ABOUT FOREIGN KEYS:

1. The foreign key column and the primary key that it references MUST be of the same data type! The column names do not have to be the same, but the data types MUST be the same.

2. The table that contains the primary key that is referenced by another table MUST be created BEFORE the table that references it. For example, the Major table MUST be created BEFORE the Student table. Think of it as follows: You can't reference something that doesn't yet exist.

3. Unlike a primary key, there can be many foreign keys in each table. Thus, you should use a sequential number after the FK suffix (even if there is only one foreign key because additional foreign keys can be added at a later time).

UNIQUE CONSTRAINTS…

Use the unique constraint to indicate that a column (or the combination of a set of columns) must contain a unique value. To create a unique contraint, use the keyword CONSTRAINT followed by the table name, an underscore, the suffix UQ followed by a sequential number. Then use the keyword UNIQUE followed by a set of parentheses in which is listed the name of the column(s) that must contain a unique value. For example,

if the login should contain a unique value, then the DDL statement for the Student table would be as follows:

```
CREATE TABLE Student (
        sid             INT,
        firstName       VARCHAR(15)  NOT NULL,
        lastName        VARCHAR(20)  NOT NULL,
        login           VARCHAR(7),
        dob             DATE,
        majorIDLONG,
        CONSTRAINT student_PK PRIMARY KEY (sid),
        CONSTRAINT student_FK1 FOREIGN KEY (majorID) REFERENCES Major(majorID),
        CONSTRAINT student_UQ1 UNIQUE(login));
```

NOTE: If the combination of two columns should be unique, then list those columns inside of the parentheses separated by commas. For example, CONSTRAINT tablename_UQ1 UNIQUE(column1, column2).

CHECK CONSTRAINTS…

Use the check constraint if the values in a column MUST be a value from a list of enumerated values. The values will be checked when the user tries to insert records into the table. To create a check constraint, use the keyword CONSTRAINT followed by the table name, an underscore, CK followed by a sequential number. Then use the keyword CHECK followed by an open parenthesis, the name of the column whose values you want to check, followed by the keyword IN followed by a set of parentheses that contain a comma separated list of valid values that the field can contain. Then close the parenthesis. For example, if the Student table contained a field named gender, which could only contain M or F (in either upper or lower case), then the DDL statement would be as follows:

```
CREATE TABLE Student (
        sid             LONG,
        firstName       VARCHAR(15)  NOT NULL,
        lastName        VARCHAR(20)  NOT NULL,
        login           VARCHAR(7),
```

```
dob             DATE,
majored         INT,
gender  CHAR(1),
CONSTRAINT student_PK PRIMARY KEY (sid),
CONSTRAINT student_FK1 FOREIGN KEY (majorID) REFERENCES Major(majorID),
CONSTRAINT student_UQ1 UNIQUE(login),
CONSTRAINT student_CK1 CHECK (gender IN ('M', 'm', 'F', 'f')));
```

MODIFYING AN EXISTING TABLE

Once a table has been created, you can use the ALTER TABLE statement to alter the table's structure. You can either use the ALTER TABLE statement to modify the existing columns OR to add a new column or to add a new constraint.

You use the DROP TABLE statement to drop a table and its data from the database.

VERY IMPORTANT NOTE ABOUT THE DROP TABLE STATEMENT:

When you have tables that have foreign keys, you MUST drop the tables that contain the foreign keys BEFORE you drop the tables that they are referencing. If you try to do it the other way (I.e. drop the tables that are being referenced BEFORE the tables that are referencing them), then you will get a foreign key violation AND an error. Given the Major and Student tables below, the following DROP TABLE statement would fail and generate an error:

```
DROP TABLE Major;
```

```
CREATE TABLE Major (
                majorID         INT,
                majorName       VARCHAR(15)  NOT NULL,
                CONSTRAINT major_PK PRIMARY KEY (majorID))
CREATE TABLE Student (
```

```
sid             INT,
firstName       VARCHAR(15)  NOT NULL,
lastName        VARCHAR(20)  NOT NULL,
login           VARCHAR(7),
dob             DATE,
majorID         INT,
CONSTRAINT student_PK PRIMARY KEY (sid),
CONSTRAINT student_FK1 FOREIGN KEY (majorID) REFERENCES Major(majorID),
CONSTRAINT student_UQ1 UNIQUE(login));
```

The statement would fail because the Major table is being referenced by the Student table. To successfully drop the tables, you would have to first drop the Student table, and then drop the Major table as follows:

```
DROP TABLE Student;

DROP TABLE Major;
```

So, when you use the DROP TABLE statement, then you must drop the tables in the opposite order that you created them.

If you do not want to worry about the order in which you have to drop the tables, then you should use the keywords CASCADE CONSTRAINTS after specifying the table name in the DROP TABLE statement. Those keywords force the database engine to successfully drop the tables in the correct order. For example, the following statements would work:

```
DROP TABLE Major CASCADE CONSTRAINTS;

DROP TABLE Student CASCADE CONSTRAINTS;
```

I highly recommend using the CASCADE CONSTRAINTS keywords with the DROP TABLE statement.