版本为 MySQL 5.7 Windows X64

cmd命令：

先配置MySQL的path路径环境，高级系统设置中path中新建添加，添加内容为MySQL的启动文件的路径。

快捷键：CTRL+C（如果命令输错可使用此命令重新输入MySQL中也同样适用）

```
net start mysql
```

命令解析：启动mysql数据库服务。**注意：命令中mysql是服务器名按个人设定为准！**

```
net stop mysql
```

命令解析：关闭mysql数据库服务。**注意：命令中mysql是服务器名按个人设定为准！**

```
mysqld --initialize-insecure --user=root
```

命令解析：mysqld --initialize-insecure（MySQL初始化新建一个data）--user=root(以管理员身份)**注：MySQL其他版本可能自带data文件夹。**

```
MySQL -u root -p
```

命令解析：以管理员的身份链接数据库。-u root(管理员身份也可以是其他用户名),-p(即将输入的密码，也可以在-p后直接输入**但这样会看到密码不安全**)

MySQL字符集编码

因为每台计算机的环境配置可能不同导致字符集编码也有所不同，这使得有些同学在查看有些数据时发生乱码的情况，为防止此情况出现将其放在最前面。

注意：cmd的特殊性我们使用默认编码为GBK，但在实际开发中使用UTF8！

```
show variables like 'character_set_%';
```

命令解析：查看全部字符编码。

例：

```
mysql> show variables like 'character_set_%';
+--------------------------+----------------------------------------------------------+
| Variable_name            | Value                                                    |
+--------------------------+----------------------------------------------------------+
| character_set_client     | gbk                                                      |
| character_set_connection | gbk                                                      |
| character_set_database   | latin1                                                   |
| character_set_filesystem | binary                                                   |
| character_set_results    | gbk                                                      |
| character_set_server     | latin1                                                   |
| character_set_system     | utf8                                                     |
| character_sets_dir       | C:\Program Files\MySQL\MySQL Server 5.7\share\charsets\ |
+--------------------------+----------------------------------------------------------+
8 rows in set, 1 warning (0.02 sec)
```

名词解释

character_set_client 客户端使用的字符集。

character_set_connection 连接数据库时的字符集。（将客户端从客户端发送的语句从character_set_client转换为character_set_connection）

character_set_databases 创建数据库的编码格式。

character_set_filesystem 文件系统的编码格式。（把操作系统上的文件名转化成此字符集，即把character_set_client转换character_set_filesystem， 默认binary是不做任何转换的。）

character_set_results 数据库给客户端返回时使用的编码格式。（如果没有指明，使用服务器默认的编码格式。）

character_set_server 服务器安装时指定的默认编码格式。（这个变量建议由系统自己管理，不要人为定义。）

character_set_system 数据库系统使用的编码格式。（这个值一直是utf8，不需要设置，它是为存储系统元数据的编码格式。）

character_sets_dir 符集安装的目录。

更改字符集编码

```
set character_set_要改的变量=要该成的字符编码;
```

命令解析：更改想要更改的MySQL字符集编码。

MySQL有关名词解释：

在展示Database中源生库名解析以及作用

information_schema(信息图示；存储服务器管理全部数据库的信息)

mysql(数据库中用户账户信息)

performance_schema(收集数据库服务器性能参数)

sys(系统文件)

SQL语句的区分

DDL　data definition language　　数据库定义语言　create(增)　drop(删) alter(改)　show(查)

解析：**可以实现对数据库结构、操作方法等的定义。**

DML　data manipulation language 数据库操纵语言 insert(曾)　delete(删)　update(改) select(查)

解析：**实现对数据的基本操作，"增删改查"。**

MySQL命令：

有关库的操作及命令

展示库命令（查show）

```
show databases;
```

命令解析：展示所有数据库。

```
show create database `库名`;
```

命令解析：查看一个数据库的创建信息。

新建库命令（曾create）

```
create database `库名`;
```

命令解析：新建一个数据库。
注意：关键字最好不用数据库名，如必须使用，须在库名两边加上反引号来实现。例：

```
create database `dabatase`;
```

常用写法

```
create database if not exists `库名`;
```

命令解析：在新建的一个数据库的基础上进行判断（**判断语句：if not exists**）该库名在所有库中是否存在，如不存在就新建。
注意：在创建时如有需要改变字符编码的要求，需在库名后加上charset=你所需要的编码名称。（常用字符编码：UTF8国际通用编码，GBK中文简体编码。）
例：新建一个名为students的库字符编码是utf8

```
create database if not exists `students` charset=utf8;
```

删除库命令（删drop）

```
drop databse 库名;
```

命令解析：删除一个数据库。

常用写法

```
drop database if exists `库名`;
```

命令解析：在删除的一个数据库的基础上进行判断（**判断语句：if exists**）该库名在所有库中是否存在，如存在就删除。

更改库的命令（改alter）

更改数据库字符编码命令

```
alter database `库名` charset=字符编码名;
```

命令解析：更改数据库字符编码。

使用库命令

```
use `数据库名`;
```

命令解析：使用库名的那个库。

有关表的操作及命令

展示表命令（查show）

```
show tables;
```

命令解析：展示库中所有的表。

```
show create table `表名`;
```

命令解析：展示库中表的创建结构以sql语句的方式展现出来。

例:

```
mysql> show create table `student`;
+---------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table   | Create Table                                                                                                                                                                                                                       |
+---------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| student | CREATE TABLE `student` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键id',
  `name` varchar(30) NOT NULL COMMENT '学生名字',
  `gender` varchar(1) NOT NULL COMMENT '性别',
  `student_phone` varchar(20) DEFAULT '暂时未知' COMMENT '学生电话',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk |
+---------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
desc '表名';
```

命令解析：展示库中表的创建结构以表的形式展示出来。

例:

```
mysql> desc `student`;
+---------------+-------------+------+-----+----------+----------------+
| Field         | Type        | Null | Key | Default  | Extra          |
+---------------+-------------+------+-----+----------+----------------+
| id            | int(11)     | NO   | PRI | NULL     | auto_increment |
| name          | varchar(30) | NO   |     | NULL     |                |
| gender        | varchar(1)  | NO   |     | NULL     |                |
| student_phone | varchar(20) | YES  |     | 暂时未知 |                |
+---------------+-------------+------+-----+----------+----------------+
4 rows in set (0.02 sec)
```

新建表命令（曾create）

```
create table `表名`(
`表字段名`数据类型（字符宽度）表字段属性,
`表字段名`数据类型（字符宽度）表字段属性
);
```

命令解析：新建一张表的命令构成。

注意：MySQL的代码终止符号是分号;中英文的分号在输入法中不同需注意，在实际MySQL操纵中会有->符号，该符号为MySQL特有的换行符号与代码无关。

常用写法例:

```
create table if not exists `student`(
id int auto_increment primary key comment'主键id',
name varchar(30) not null comment '学生名字',
student_phone varchar(20) default '暂时未知' comment '学生电话'
)engine=innodb charset=gbk;
```

注意：写该代码时最好不要写错，否则需要全部重写。

命令解析:

create table if not exists `student`

新建一个名为*student*（学生）的表，库中没有则创建。

id int auto_increment primary key comment'主键id',

id 为字段名此例中为学生学号,由此选择数据类型为 *int*

auto_increment 结合学生学号的性质定义列为自增的属性，一般用于主键，数值会自动加*1*

prmary key 定义列为主键，作用：该字段为唯一的。结合例子，学生的学号是唯一的，这张表靠此来区分学生。

comment 为注释。 '主键*id*' 为注释的内容注释的内容使用英文单引号。 **此注释内容并不标准，但流程是需要的，实际注释内容标准需参考详细企业标准。**

name varchar(30) not null comment '学生名字',

根据例子 *name* 字段名使用的字符类型为 *varchar* 考虑到英文名字字符宽度设为*30*。

not null 不想字段是空的可以设置字段的属性为 ***NOT NULL***， 在操作数据库时如果输入该字段的数据为空 ，就会报错。 结合例子学生必须要有名字。

题外话：不知读者知不知道那个名字特长的非洲小哥的梗，他说他没上过学更不会拼自己的名字，可能就是因为他名字太长没法录入才没法上学吧哈哈哈哈。他名字音译是（Uvuvwevwevwe Onyetenyevwe Ugwemuhwem Osas）

student_phone varchar(20) default '暂时未知' comment '学生电话'

根据例子 *phone* 字段名适用类型为 *varchar* 考虑到区号问题字符宽度设为*20.*

default '暂时未知' *default*默认值，用于该字段可填可不填。根据例子，大部分学生都拥有个人手机但是难免会有特殊情况。

下面没有关于字段内容所以代码后面不需要写逗号了。

engine=innodb

设置储存引擎

修改表的命令（改alter）

```
alter table `要改的表名` rename to `要改成的表名`
```

命令解析：改表名。**注意：切记表名不能为复数这不符合规范。**

删除表命令（删drop）

```
drop table if exists `表名`,`表名`,`表名`;
```

命令解析：删除表，可以一下删除多个。

修改字段的命令(改alter)

添加字段

```
alter table `表名` add 字段名 数据类型（字符宽度）字段属性;
```

命令解析：在表中添加字段。

```
alter table `表名` add 字段名 数据类型（字符宽度）字段属性 after 字段名;
```

命令解析：在表中某字段之后添加字段。

```
alter table `表名` add 字段名 数据类型（字符宽度）字段属性 first;
```

命令解析：在表中添加字段在所有字段最前面。

删除字段

```
alter table `表名` drop 字段名;
```

命令解析：在表中删除字段。

更改字段

```
alter table `表名` change 要更改的字段名 要改成的字段名 要改成的数据类型（字符宽度）要改成的字段属性;
```

命令解析：在表中更改字段。

```
alter table `表名` modify 要更改的字段名 要改成的类型(字符宽度);
```

命令解析：在表中只更改字段的类型。

表中的数据有关的命令

插入数据 insert(曾)

默认的插入方式

```
insert into `要插入的表名` values (要插入的数据，要插入的数据，要插入的数据，要插入的数据);
```

命令解析：往表中插入数据。

注意：在使用该命令插入数据时，要插入的数据顺序要严格遵守表中字段内容所对应的顺序,不得丢项少项。

该命令也可以插入多条数据

```
insert into `要插入的表名` values (要插入的数据，要插入的数据，要插入的数据，要插入的数),(要插入的数据，要插入的数据，要插入的数据，要插入的数);
```

可自定义插入顺序方式

```
insert into `要插入的表名` (表中字段名，表中字段名，表中字段名) values (插入的数据，插入的数据，插入的数据);
```

命令解析：表中插入数据，但可选择想要插入数据的字段。

注意：在使用该命令插入数据时

1.要插入的数据顺序，要严格遵守命令前面中的字段所对应的顺序,不得丢项少项!

2.在选择要插入数据的字段时，切记表中设置必填数据的字段不能少，否则抱错!

3.当真不知道必填字段要填什么数据时，可以填 NULL 如果设置了 default 可以填default。（default用法在新建表命令的案例中有简单解析）

上面这两条插入数据命令在企业实战中并非主要使用方式但是这是基础要会的。

查找数据 select(查)

查表全部

```
select * from `表名`;
```

命令解析：查询表中所有的数据。（这种方式吃性能，不建议使用。）

例：

```
mysql> select * from student;
+----+------+--------+---------------+
| id | name | gender | student_phone |
+----+------+--------+---------------+
|  1 | Jake | 1      | 暂时未知      |
+----+------+--------+---------------+
1 row in set (0.00 sec)
```

查表中想要字段

```
select `要查找数据的字段`, `要查找数据的字段` from `表名`
```

命令解析：查询表中你想要查询的字段的数据。

例：

```
mysql> select `name`, `gender` from student;
+------+--------+
| name | gender |
+------+--------+
| Jake | 1      |
+------+--------+
1 row in set (0.00 sec)
```

删除数据 delete(删)

选择删除

```
delete from `要删除数据的表名` where `字段`=对应字段的数据;
```

命令解析：一张表中，删除指定的一条数据。(数据是字符串要用要用单引号)

命令中的where（哪里）`字段`=对应字段的数据; ，用字段对应数据的方式来定位要删除的数据。

例：

```
mysql> select * from `student`;
+----+----------+--------+----------------+
| id | name     | gender | student_phone  |
+----+----------+--------+----------------+
|  1 | Jake     | 1      | 暂时未知       |
|  2 | Prophet  | 1      | 123456789      |
|  3 | TOM      | 1      | 123456789      |
|  4 | Disapain | 2      | 122222222      |
|  5 | SANDEAO  | 2      | 12222222233    |
|  6 | Sitelin  | 2      | 12222223344    |
+----+----------+--------+----------------+
6 rows in set (0.00 sec)

mysql> delete from `student` where `name`='tom';
Query OK, 1 row affected (0.01 sec)

mysql> select * from `student`;
+----+----------+--------+----------------+
| id | name     | gender | student_phone  |
+----+----------+--------+----------------+
|  1 | Jake     | 1      | 暂时未知       |
|  2 | Prophet  | 1      | 123456789      |
|  4 | Disapain | 2      | 122222222      |
|  5 | SANDEAO  | 2      | 12222222233    |
|  6 | Sitelin  | 2      | 12222223344    |
+----+----------+--------+----------------+
5 rows in set (0.00 sec)
```

注意： 删除数据的时候先择字段 切记　要选择主键或者共有数据少的或没有的字段。如果出现重复的数据将会一起删除！那将成为灾难！！！！！！！

比如说按上图例子，我们选择gander字段并数据写2那么id为4 5 6 的数据将一起删除。

变形

```
delete from `要删除数据的表名` where `age`>30;
```

命令解析：这条命令是上面命令的一种变形，我们也可以用筛选的方式选择删除数据。看命令，表中age是年龄大于30的数据将会被删掉。

注意：会留下痕迹，痕迹就是如果该表设置主键自增字段（这很常见）那么当你再去将数据插入这张表时，将不会从1开始，而是未删表之前的数据接着向下排列。解决办法在 高级部分->事务 的最后。

全部删除

方法一

```
delete from `表名`;
```

命令解析：将表中的数据以遍历的方式删除。（就像for循环一样，一条一条删除。）

注意：不推荐使用这种方式，删除时间慢不说还会留下痕迹，痕迹就是如果该表设置主键自增字段（这很常见）那么当你再去将数据插入这张表时，将不会从1开始，而是未删表之前的数据接着向下排列。这也是个不小的灾难！！！如果你想删库跑路这种方法，绝对不是个很好的选择。

方法二

```
truncate table `表名`;
```

命令解析：将表中的数据直接删除，只留下表的结构（字段）。

更新数据 update(改)

```
update `表名` set `要改的字段`=改成的数据 where `字段`=对应字段的数据;
```

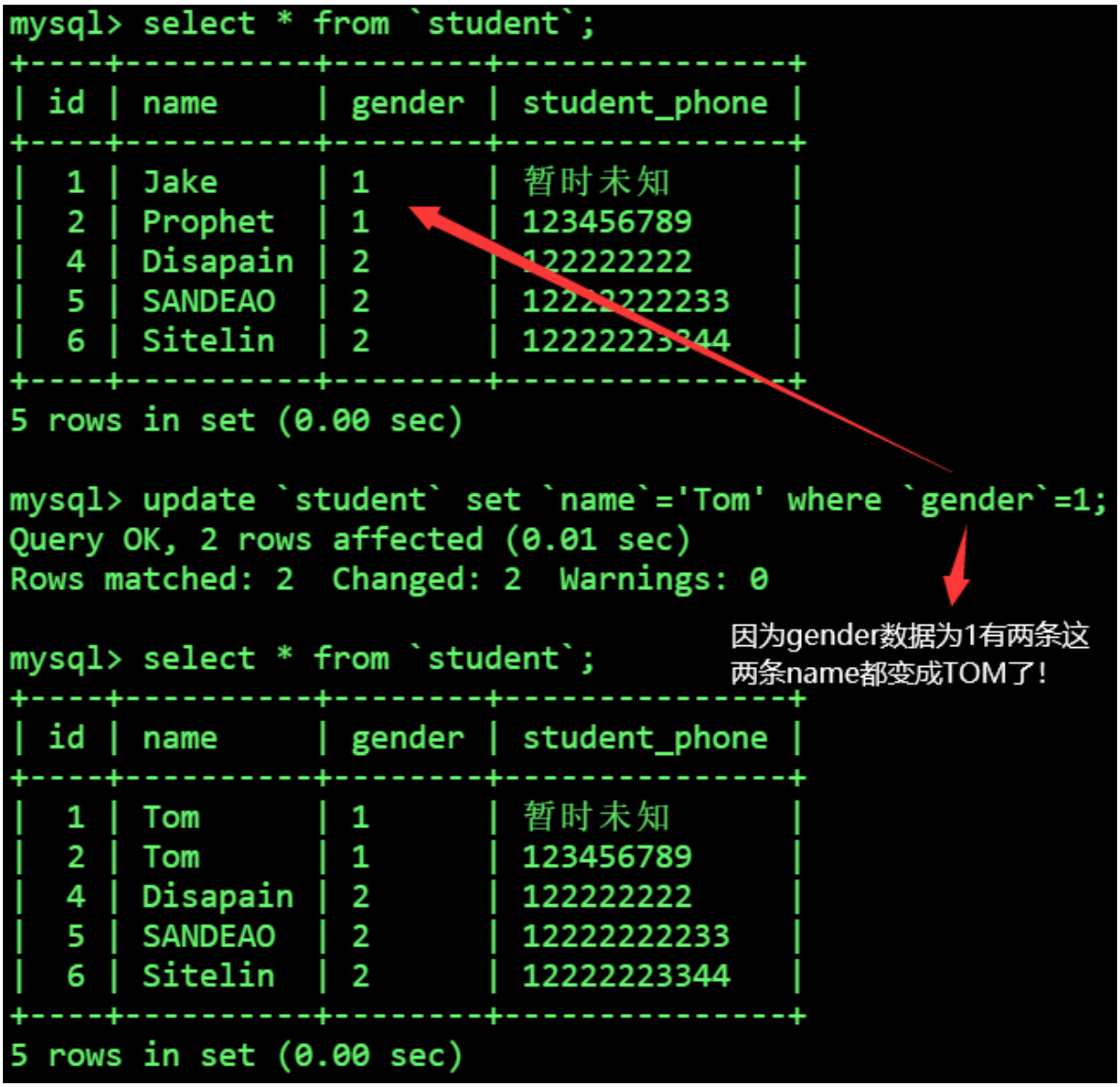
命令解析：

set(放置)，update `表名` set `要改的字段`=改成的数据 ，这半句用来设置要把字段改成什么数据。

where（哪里） `字段`=对应字段的数据;，这半句用来确定要改数据在那条数据里 。

注意： where后面选的字段 切记 要选择主键或者共有数据少的或没有的字段。如果出现重复的数据将会一起更改。无意出现，那将成为灾难！！！！！！！

看不懂看返例:



如果不写where以及后面的语句，表里的内容会被全部更改

变形一

```
update `表名` set `要改的字段`=改成的数据, `要改的字段`=改成的数据 where `字段`=对应字段的数据;
```

命令解析：一条数据中，改多个字段的数据，用逗号隔开。

变形二

```
update `表名` set `要改的字段`=改成的数据 where `字段`=对应字段的数据  or `字段`=对应字段的数据 ;
```

命令解析：or在这条命令中的作用：or前后的数据有哪个改哪个，两个都有就都该，可改多条数据。

关于数据库改名的问题

比如数据库名称old_db想改名为new_db

MySQL修改数据库名称比较麻烦，不支持直接修改，需要通过其它方式间接达到修改数据库名称的目的。
在 MySQL 5.1.23 之前的旧版本中，我们可以使用 RENAME DATABASE 来重命名数据库，但此后版本，因为安全考虑，删掉了这一条命令。
方法一：先导出数据，再导入数据
当数据库体积比较小时，最快的方法是使用mysqldump命令来创建整个数据库的转存副本，然后新建数据库，再把副本导入到新数据库中。

1.1 先创建新库：

```
create database new_db;
```

1.2 使用mysqldump导出数据：

```
mysqldump -uroot -p123456 --set-gtid-purged=OFF old_db > /tmp/old_db.sql
```

仅是做普通的本机备份恢复时,可以添加

```
--set-gtid-purged=OFF
```

作用是在备份时候不出现GTID信息

1.3 导入数据到新库：

```
mysql -uroot -p123456 new_db < /tmp/old_db.sql
```

方法二：通过修改表名称，间接实现修改数据库名称
使用此方法实际上将所有表从一个数据库移动到另一个数据库，这实际上重命名了该数据库（MySQL没有单个语句的操作），移动后原始数据库继续存在，但是里面没有表。

2.1 先创建新库：

```
create database new_db;
```

1
2.2 使用RENAME TABLE命令修改表名，将表移动到新的库里：

```
rename table old_db.tb to new_db.tb;
```

1
2.3 完成后删除旧库：

```
drop database old_db;
```

1
2.4 如何使用shell脚本来批量修改表名：
当库下表比较多的时候，用上面方法纯手动也不现实，好在linux下可以用shell脚本来批处理。
附上一个shell脚本批量修改表名称。

```
#!/bin/bash
mysql -uroot -p123456 -e 'create database if not exists new_db;'
list_table=$(mysql -uroot -p123456 -Nse "select table_name from
information_schema.TABLES where TABLE_SCHEMA='old_db'")

for table in $list_table
do
    mysql -uroot -p123456 -e "rename table old_db.$table to new_db.$table"
done
```

mysql登陆命令行参数
-e, --execute=name # 执行mysql的sql语句
-N, --skip-column-names # 不显示列信息
-s, --silent # 一行一行输出，中间有tab分隔

关于在库中复制表结构与查出表的数据插入新表的方法

复制表结构

```
create table `表名` like `要复制结构的表名`;
```

解析：复制表的结构，包括每个字段的属性。

查出表的数据插入新表

插入全部数据

```
insert into `要插入数据的表名` select * from `数据来源的表名`;
```

解析：查出表的全部数据并插入新表，**谨慎使用select * from 如果新表的字段以及字段属性与原表不同将无法插入。**

关于在特殊情况下修改表中单独一个字段的字符集编码

```
alter table `表名` change `要更改的字段名` `要改成的字段名` 要改成的数据类型（字符宽度）
character set 要改成的字符集编码 要改成的字段属性;
```

查看表中字段的字符集编码只能用

```
show create table `字段名`;
```

和默认表的字符编码不同的字段会显示出来

例：