mysql 语法：

```
CREATE DATABASE 数据库名称;

SHOW DATABASES;

USE 数据库名称;

DROP DATABASE 数据库名称;

create table table_name (列名1 属性,列名2 属性,...);

create table db_admin(
    id int auto_increment primarykey,
    user varchat(32) not null,
    password varchar(32) not null,
    createtime datetime
);

 查看数据表结构SHOW COLUMNS或DESCRIBE
SHOW [FULL] COLUMNS FROM 数据表名 [FROM 数据库名];
--或者
SHOW [FULL] COLUMNS FROM 数据库名.数据表名;
DESCRIBE 数据表名称 [列名];
--或者
DESC 数据表名称 [列名];

查看键表语句SHOW CREATE TABLE
--示例:查看user表的键表语句
show create table user;
CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`),
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

在tb_admin添加一个新的email，类型varchar(50)，not null，将字段user的类型由varchar(30)改为varchar(40)

alter table tb_admin add email varchat(50) not null,modify user varchar(40);

alter_specification:
ADD [COLUM] create_definition [FIRSR | AFTER column_name ]  --添加新字段
| ADD INDEX [index_name] (index_col_name,...)                    --添加索引名称
| ADD PRIMARY KEY (index_col_name,...)                           --添加主键名称
| ADD UNIQUE [index_name] (index_col_name,...)                   --添加唯一索引
| ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}  --修改字段名称
| CHANGE [COLUMN] old_col_name create_definition                 --修改字段类型
| MODIFY [COLUMN] create_definition                              --修改子句
| DROP [COLUMN] col_name                                         --删除子段名称
| DROP PRIMARY KEY                                               --删除主键名称
| DROP INDEX index_name                                          --删除索引名称
| RENAME [AS] new_tbl_name                                       --更改表名
```

```
| table_options

重命名
RENAME TABLE 数据表名1 To 数据表名2

删除：
DROP TABLE 数据表名;
--或
DROP TABLE IF EXISTS 数据表名;

--实现克隆表
--1.方法一 LIKE
CREATE TABLE 新表名 LIKE 被克隆表名;
INSERT INTO 新表名 SELECT * FROM 被克隆表名;

--2.方法二
CREATE TABLE 新表名 (SELECT * FROM 被克隆表名);


--清空表
--1.记录自增ID未删除
DELETE FORM 表名;
--DELETE是一行一行的删除记录。若有自增字段，再次添加会从原记录最大的自增字段开始写入记录

--2.记录自增ID删除
TRUNCATE TABLE 表名;
--TRUNCATE不会返回被删除条目，TRUNCATE是将表重建，速度快于DELETE，自增值从初始值开始

临时表：
CREATE TEMPORARY TABLE 表名(...);
--示例：
create temporary table tb_admin(
    id int(4) zerofill primary key auto_increment,
    name varchar(32) not null,
    sex char(2) not null
);

insert into 数据表名[(column_name1,column_name2,...)] value(v1,v2,...)[,
(v1,v2,...)];
--后面给定多个括号进行批量添加，中间使用逗号隔开

SELECT [DISTINCT] [CONCAT (col1,":",col2) as col] selection_list --要查询的内容,选
择哪些列
FROM 数据表名tb_list                                    --指定数据表
WHERE primary_constraint                              --查询时需要满足的内容
GROUP BY grouping_columns                             --如何对结果进行分组
ORDER BY sorting_columns                              --如何对结果进行排序
HAVING secondary_constraint                           --查询时满足的第二条件
LIMIT count                                           --限定输出的查询结果


select tb_book.id,tb_book.name,tb_bookinfo.author,tb_bookinfo.price --查询的列
from tb_book,tb_bookinfo                                            --数据表
where tb_book.name = tb_bookinfo.name                               --匹配条件
and tb_bookinfo.name = '演员的自我修养';                              --外部条件
```

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 数据表名[(create_definition,...)] [table_options] [select_statement]
```

CREATE TABLE语句的参数说明如下：

参数	说明
TEMPORARY	使用该关键字创建一个临时表
IF NOT EXISTS	用于避免表存在时MySQL报错
create_definition	表的列属性部分，至少一列
table_options	表的一些特性参数
select_statement	SELECT语句描述部分，用它可以快速的创建表

下面是create_definition部分，每一列定义的具体格式：

```
col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [PRIMARY KEY] [reference_definition]
```

列属性create_definition参数说明：

参数	说明
col_name	字段名
type	字段类型
NOT NULL | NULL	指出该列是否允许是空值
DEFAULT default_value	默认值
AUTO_INCREMENT	是否为自动编号，表中只能有一个AUTO_INCREMENT列，并且必须被索引
PRIMARY KEY	是否为主键，表中只能有一个PRIMARY KEY
reference_definition	为字段添加注释

> PRIMARY KEY：如果一个表中没有PRIMARY KEY，而某些应用程序需要PRIMARY KEY，MySQL将返回第一个没有任何NULL列的UNIQUE键作为PRIMARY KEY。

4.修改表结构ALTER TABLE

> 通过alter修改表结构必须是表中数据全部删除之后才能进行修改

修改表结构指的是：增加或删除字段、修改字段名称或类型、取消设置主键外键、取消设置索引以及修改表的注释等。语法：

示例

在tb_admin添加一个新的email，类型varchar(50)，not null，将字段user的类型由varchar(30) 改为varchar(40)

```
alter table tb_admin add email varchat(50) not null,modify user varchar(40);
```

详细讲解

```
ALTER[IGNORE] TABLE 数据表名 alter_spec[,alter_spec]...
```

> 当指定IGNORE时，如果出现重复关键的行，则只执行一行，其他重复的行被删除

where:

设定查询条件，可以通过它实现很多复杂的条件查询。where子句中常用的比较运算符：

运算符	名称	示例
=	等于	id=2
>	大于	id>2
>=	大于等于	id>=2
<	小于	id<2
<=	小于等于	id<=2
!= 或 <>	不大于	id!=2或id<>2
IS NULL	n/a(Not applicable)	id IS NULL
IS NOT NULL	n/a	id IS NOT NULL
BETWEEN ... AND ...	n/a	id BETWEEN 2 AND 12
IN	n/a	id IN(1,2,3)
NOT IN	n/a	id NOT IN(1,2,3)
LIKE	模式匹配	name LIKE('shi%')
NOT LIKE	模式匹配	name NOT LIKE('shi%')
REGEXP	正则表达式	name 正则表达式

4.GROUP BY

将数据划分到不同的组中，实现对记录的分组查询，在与AVG()和SUM()函数一起使用时，GROUP BY子句发挥最大作用。示例：

```
--查询tb_book表，按照type分组，求每类图书的平均价格，
select type,avg(price) from tb_book group by type;
```

5.DISTINCT

去除重复的行。

```
--查询tb_book表，除去重复的type数据
select distinct type from tb_book;
```

6.ORDER BY

对查询结果进行升序（默认）或降序（DESC）排列。升序NULL在最前面，降序NULL在最后面。

```
--查询tb_book表中的所有记录，按照id降序排列，显示三条记录
select * from tb_book order by id desc limit 3;
```

7.LIKE模糊查询

可实现模糊查询，它有两种通配符：`%`和`_`，`%`可以匹配一个或多个字符，`_`可以匹配一个字符。示例：

```
--查找所有第二个字符是h的图书
select * from tb_book where name like('_h%');
```

> 单个中文也算一个字符，比如`卷`就算一个字符。

8.CONCAT联合多列

联合多个字段，构成一个总的字符串。

```
--把name和price合并成一个新的字符串输出
select id,concat(name,":",price) as info,type from tb_book;
```

9.LIMIT限定结果行数

示例：

```
--1.降序显示3条记录
select * from tb_book order by price desc limit 3;

--2.从id=1显示4条记录
select * from tb_book where id limit 1,4;
```

```
--计算tb_book表中各类图书的总价格
select sum(price) as total,type from tb_book group by type;
--price打八折
select *,(price * 0.8) as "八折" from tb_book;

 3.修改记录UPDATE
UPDATE 数据表名 SET column_name1 = new_v1,column_name2 = new_v2,... where
condition;
1
```

其中set子句指出要修改的列和它们所给定的值；where可选，给出则指定某一行更新，否则全部记录行被更新。

```
--将username=admin的密码修改为123
update tb_user set password = '123' where username = 'admin';
1
2
```
更新时一定要保证WHERE子句正确，一旦WHERE子句出错，将会破坏所有改变的数据

删除记录DELETE
```
DELETE FROM 数据表名 WHERE condition;
1
```
删除需谨慎，一般where条件以id为判断，减少很多不必要的麻烦

名称	说明
avg(字段名)	获取指定列的平均值
count()	如果指定了一个字段会统计出该字段非空记录； 在前面加上DISTINCT会统计不同的记录； 使用COUNT(*)会统计包括NULL的所有记录数
min()	获取指定字段的最小值
max()	获取指定字段的最大值
std()	指定字段的标准背离值
stdtev()	与STD相同，为了兼容Oracle
sum()	指定字段所有记录的总和

> 还可使用算术运算符、字符串运算符和逻辑运算符构成表达式

数据库用户操作：

```
CREATE USER '用户名'@'来源地址' [IDENTIFIED BY [PASSWORD] '密码'];

----------------------------------解释--------------------------------
'用户名'：指定将创建的用户名
'来源地址'：指定新创建的用户可在哪些主机上登录，可使用IP地址、网段、主机名的形式，
           本地用户可用localhost，允许任意主机登录可用通配符%
'密码'：若使用明文密码，直接输入'密码'，插入到数据库时由Mysql自动加密；
        若使用加密密码，需要先使用SELECT PASSWORD('密码'); 获取密文，再在语句中添加
PASSWORD '密文';
        若省略“IDENTIFIED BY”部分，则用户的密码将为空（不建议使用）
----------------------------------------------------------------------
例如：
create user 'test1'@'localhost' IDENTIFIED BY '123456';

select password('123456');
create user 'test2'@'localhost' IDENTIFIED
BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9';


查看用户信息
创建后的用户保存在mysql数据库的user表里

--切换数据库
use mysql;
select user,authentication_string,Host from user;


重命名用户名
RENAME USER '用户名'@'来源地址' TO '新用户名'@'新来源地址';
```

```
DROP USER '用户名'@'来源地址';

--1.修改当前登陆用户密码
SET PASSWORD = PASSWORD('123');

--2.修改其他用户密码
SET PASSWORD FOR '用户名'@'来源地址' = PASSWORD('123');

--重置密码
update mysql.user set AUTHENTICATION_STRING = PASSWORD('abc123') where
user='root';
--刷新
FLUSH PRIVILEGES;
```

在 /etc/my.cnf配置文件下的[mysqld]添加：`skip-grant-tables`，之后重启服务，直接输入mysql就可以登陆

```
Bye
[root@192 ~]# vim /etc/my.cnf

[client]
port = 3306
socket=/usr/local/mysql/mysql.sock

[mysqld]
user = mysql
basedir=/usr/local/mysql
datadir=/usr/local/mysql/data
port = 3306
character-set-server=utf8
pid-file = /usr/local/mysql/mysqld.pid
socket=/usr/local/mysql/mysql.sock
bind-address = 0.0.0.0
skip-name-resolve
max_connections=2048
default-storage-engine=INNODB
max_allowed_packet=16M
server-id = 1
skip-grant-tables   <-- 添加模块

[root@192 ~]# systemctl restart mysqld.service   重启服务
[root@192 ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3   直接登录
Server version: 5.7.20 Source distribution

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.
```