--单行注释

#单行注释

/\*\*/多行注释

--创建并连接数据库

cd C:\xampp\mysql\bin

--登录账号密码

--学校密码123456，自己电脑空白

--mysql -uroot -p123456 mydb;

mysql -u root -p

--创建数据库(mydb为自定义数据库名称)

CREATE databases mydb;

--防止创建的数据库存在，存在会返回警告信息

CREATE DATABASE IF NOT EXISTS mydb;

--查看警告信息

show warnings;

--查看数据库

show databases;

--查看指定的数据库

show CREATE DATABASE  mydb;

--选择数据库

USE mydb;

--删除数据库

DROP DATABASE mydb;

DROP DATABASE IF EXISTS mydb;

--查看所有数据表

show tables;

--查看名称中有new的数据表show tables[like 匹配模式]

--%匹配一个或多个字符，\_仅匹配一个字符

show tables  LIKE '%new%';

--创建数据表

CREATE TABLE 数据表名();

--temporary创建临时表，仅当前字段可见

--字段属性，某些约束属性

CREATE [temporary] TABLE [IF NOT EXISTS] 数据表名 (

字段名 字段类型 [字段属性]，

字段名 字段类型 [字段属性]

)[表选项];

--查看数据表详细内容

SELECT \* FROM 数据表名;

--查看表结构

--DESCRIBE 数据表名;

DESC 数据表名;

show CREATE TABLE 数据表名;

--修改数据表

--[to/as]可省略

ALTER TABLE 旧表名 rename [to/as] 新表名;

--同时修改多个数据表

rename TABLE 旧表名1 TO 新表名1[,旧表名2 TO 新表名2]...

--修改表结构

--将数据表goods中名为aaa的字段修改为des

--ALTER TABLE goods change aaa des varchar(255);

ALTER TABLE 数据表名 change[字段属性] 旧字段名 新字段名 字段类型[字段属性];

--修改表内容字段类型

--将数据表goods中des的字段的类型varchar(255)修改为char(55)

--ALTER TABLE goods modify des char(55);

ALTER TABLE 数据表名  modify[column] 字段名 新类型 [字段属性];

--移动表字段

--将goods表中最后一个字段des移动到name后

--ALTER TABLE goods modify des VAR(55) after name;

--将字段名1调整为数据表的第一个字段

ALTER TABLE 数据表名 modify[column] 字段名1 数据类型 [字段属性] first 字段名2;

--将字段名1插入字段名2的后面

ALTER TABLE 数据表名 modify[column] 字段名1 数据类型 [字段属性] after 字段名2;

--新增一个字段(默认添加到表的最后)

ALTER TABLE 数据表名 ADD[column] 新字段名 字段类型 [first 字段名];

ALTER TABLE 数据表名 ADD[column] 新字段名 字段类型 [after 字段名];

--新增多个字段(多个字段无法指定添加位置)

ALTER TABLE 数据表名 add[column] (新字段名1 字段类型1,新字段名2 字段类型2,...)

--删除字段

ALTER TABLE 数据表名 DROP 字段名;

--删除数据表

--IF EXISTS用于删除一个不存在的数据表时防止产生错误

DROP[temporary] TABLE[IF EXISTS] 数据表1,数据表2;

--添加数据

INSERT INTO 数据表名 VALUES

('1','2');

--只查看数据表编号(sno)为u004的详细信息

SELECT \* FROM goods WHERE sno = 'u004';

--修改u004的会员信息(svip)

UPDATE goods SET svip = 1000 WHERE sno = 'u004';

--删除表数据编号(sno)为u005信息

DELETE FROM goods WHERE sno = 'u005';

--查询表中年龄超过60岁的抗疫英雄姓名(name)和年龄(age)。

--SELECT name,age  FROM goods WHERE age >= '60';

SELECT name,age  FROM goods WHERE age >60;

--查询表中最大年龄的抗疫英雄

--全部信息

SELECT \* FROM goods WHERE age = (SELECT max(age) FROM goods);

--单独姓名

SELECT name FROM goods WHERE age = (SELECT max(age) FROM goods);

--查询表中获得“人民英雄”国家荣誉称号(honor)的抗疫英雄人数

--\*代表全部

SELECT COUNT(\*) FROM goods WHERE honor = '人民英雄';

--查询年龄(age)的最大值

SELECT max(age) FROM goods；

--允许中文输入

engine=innodb DEFAULT charset = utf8;

--数据类型 字节数 无符号的取值范围 有符号取值范围

TINYINT   1  0～255                    -128～127

SMALLINT  2  0～65535                  -32768～32767

MEDIUMINT 3  0～16777215               -8388608～8388607

INT       4  0～4294967295             -2147483648～2147483647

BIGINT    8  0～18446744073709551615   -9223372036854775808～9223372036854775807

--设置零填充

--若数值宽度小于显示宽度，会在左侧填充0

ZEROFILL

--数据类型浮点数 字节数 负数取值范围 非负数取值范围

--FLOAT的精度大约6~7位，DOUBLE的精度大约15位左右。

FLOAT     4  -3.402823466E+38 ~-1.175494351E-38                               0和1.175 494 351E-38 ~3.402 823 466E+38

DOUBLE    8  -1.797 693 134 862 315 7E+308 ~ -2.225 073 858 507 201 4E-308    0 和2.225 073 858 507 201 4E-308 ~1.797 693 134 862 315 7E+308

--定点数类型通过DECIMAL(M,D)设置位数和精度

--DECIMAL(5,2)表示的取值范围是-999.99~999.99。

DECIMAL

--二进制数：在二进制字符串前加前缀b。b'1000001'

--十六进制数：有两种表示方式，形如“x'41'”和“0x41”。

--转义字符：在字符前加“\”转义。

--"ab\"c"  ab"c

--时间和日期类型 取值范围 日期格式 零值

YEAR        1901~2155                                YYYY                0000

DATE        1000-01-01~9999-12-3                     YYYY-MM-DD          0000-00-00

TIME        -838:59:59~838:59:59                     HH:MM:SS            00:00:00

DATETIME    1000-01-01 00:00:00~9999-12-31 23:59:59  YYYY-MM-DD HH:MM:SS 0000-00-00 00:00:00

TIMESTAMP   1970-01-01 00:00:01~2038-01-19 03:14:07  YYYY-MM-DD HH:MM:SS 0000-00-00 00:00:00

CURRENT\_DATE 或者 NOW() 输入当前系统日期

--数据类型 类型说明

CHAR()                                 固定长度字符串

VARCHAR()                              可变长度字符串

TEXT                                   大文本数据

ENUM('值1', '值2', '值3', …, '值n')    枚举类型                            ENUM类型的数据只能从枚举列表中取，并且只能取一个

SET('值1', '值2', '值3', …, '值n')     字符串对象                          set可以从列表中选择一个或多个值来保存，多个值之间用逗号“,”分隔

BINARY                                 固定长度的二进制数据

VARBINARY                              可变长度的二进制数据

BLOB                                   二进制大对象（Binary Large Object）

--默认约束：为数据表中的字段指定默认值。

字段名 数据类型  DEFAULT 默认值;

--删除默认约束UNSIGNED

ALTER TABLE 表名 MODIFY age INT UNSIGNED;

--添加默认约束

ALTER TABLE 表名 MODIFY age INT UNSIGNED DEFAULT 18;

--非空约束：字段的值不能为NULL。

字段名 数据类型 NOT NULL;

--唯一约束：保证数据表中字段的唯一性，即表中字段的值不能重复出现。

--列级约束

字段名 数据类型 UNIQUE;

--表级约束

UNIQUE(字段名1, 字段名2, …);

--添加唯一约束

ALTER TABLE 表名 ADD UNIQUE 字段名;

--删除唯一约束

ALTER TABLE 表名 DROP INDEX 字段名;

--创建测试表，添加复合唯一键

UNIQUE(字段名1, 字段名2)

--只有当两个字段同时发生重复时，插入记录失败

--(1,4)(1,3)可以录入，再次录入(1.4)失败

--主键约束，主键的作用：唯一标识表中的记录。

--列级约束

字段名 数据类型 PRIMARY KEY

--表级约束

PRIMARY KEY (字段名1, 字段名2, …)

--删除主键约束

ALTER TABLE 表名 DROP PRIMARY KEY;

--添加主键约束

ALTER TABLE 表名 ADD PRIMARY KEY (字段名);

----自动增长

字段名 数据类型 AUTO\_INCREMENT

--修改自动增长值

ALTER TABLE 表名 AUTO\_INCREMENT = 10;

--删除id自动增长

ALTER TABLE 表名 MODIFY id 字段类型 UNSIGNED;

--重新为id添加自动增长

ALTER TABLE 表名 MODIFY id 字段类型 UNSIGNED AUTO\_INCREMENT;

--select语句

SELECT [ALL|DISTINCT] \* | 列名1[,列名2,……列名n] FROM 表名

[WHERE条件表达式] [GROUP BY 列名 [ASC | DESC]

[HAVING 条件表达式]]

[ORDER BY 列名  [ASC | DESC] , ...]

[LIMIT [OFFSET] 记录数];

--WHERE子句：用于指定查询筛选条件。

--GROUP BY子句：用于将查询结果按指定的列进行分组；其中HAVING为可选参数，用于对分组后的结果集进行筛选。

--ORDER BY子句：用于对查询结果集按指定的列进行排序。

--LIMIT子句：用于限制查询结果集的行数。参数OFFSET为偏移量，当OFFSET值为0时，表示从查询结果的第1条记录开始，如果OFFSET为1时，表示查询结果从第2条记录开始。

--as可以用于备注表列名

--查询Goods中每件商品的销售总价，其中销售总价=销售数量\*价格，显示商品名称和销售总价。

SELECT gdName AS 商品名称,gdPrice\*gdSaleQty AS 销售总价 FROM goods;

--查询Users（用户信息表）中，用户名和年龄。

SELECT uName AS 用户名,YEAR(NOW())-YEAR(uBirth) AS 年龄 FROM users;

--可使用BETWEEN AND来限制查询数据的范围

WHERE 表达式 [NOT] BETWEEN 初始值 AND 终止值

--查询Users表中2000年后出生的，且性别为“男”的用户姓名，电话号码，出生年月。

SELECT  uName AS 姓名,uPhone AS 电话号码, uBirth AS 出生日期 FROM Users WHERE uBirth>=2000-1-1 AND uSex='男';

--查询Goods表中gdCity值为“长沙或“西安”，且gdPriced小于等于50的商品名称,商品价格。

SELECT gdName AS 商品名称 ,gdPrice AS 商品价格 FROM goods WHERE  gdPrice<=50  AND  (gdCity='长沙' OR gdCity='西安' );

--查询Goods表中gdPriced在100到500元的商品名称,商品价格

SELECT gdName AS 商品名称 ,gdPrice AS 商品价格 FROM goods WHERE gdPrice BETWEEN 100 AND 500;

--IN运算符与BETWEEN…AND运算符类似，用来限制查询数据的范围

WHERE 表达式 [NOT] IN (值1,值2…值N)

--查询Goods表中gdCity为长沙、西安、上海三个城市的商品名称,商品价格

SELECT gdName AS 商品名称 ,gdPrice AS 商品价格  FROM goods WHERE gdcity IN('长沙','西安','上海');

--使用LIKE运算符，实际中当需要查询的条件只能提供不完全确定的部分信息时，就需要使用LIKE运算符实现字符串的模糊查询

WHERE 列名 [NOT] LIKE ‘字符串’ [ESCAPE ‘转义字符’]

--通配符         说明              示例

                任意字符串        s%：表示查询以s开头的任意字符串，如small

                                  %s：表示查询以s结尾的任意字符串，如address

     %                            %s%：表示查询包含s的任意字符串，如super、course

   \_             任何单个字符     \_s：表示查询以s 结尾且长度为2的字符串，如as

                                  s\_：表示查询以s开头且长度为2的字符串，如sa

--查询Users表中gdName为“李”开头的用户姓名、性别和手机号

SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users WHERE uName LIKE '李%';

--查询Users表中gdName第2个字为“湘”的用户姓名、性别和手机号

SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users WHERE uName LIKE '\_湘%';

--查询Goods表中gdName以“华为P9\_”开头的gdCode

SELECT gdCode FROM goods WHERE gdName LIKE '华为P9\\_%';

--使用IS NULL运算符，系统自动将其设置为空值

WHERE 列名 IS [NOT] NULL

--查询Users表中uImage为空的用户姓名和性别

SELECT uName AS 用户姓名,usex AS 性别 FROM users WHERE uimage IS NULL;

--使用DISTINCT消除重复结果集

--查询Goods表中gdPrice大于200的商品来源哪些城市

SELECT DISTINCT gdname AS 商品名称,gdCity AS 来源城市 FROM goods WHERE gdPrice >200;

--使用REGEXP运算符

WHERE 列名 REGEXP ‘模式串’

--模式    说明                                            示例

^         匹配字符串的开始位置                          '^d'：匹配以字母d开头的字条串，如dear，do

$         匹配字符串的结束位置                          'st$'：匹配以st结束的字符串，如test，resist

.         匹配除 "\n" 之外的任何单个字符                'h.t'：匹配任何h和t间的一个字符，如hit，hot

[...]     匹配字符集合中的任意一个字符                  '[ab]'：匹配ab中的任意一个字符， 如：plain，hobby

[^..]     匹配非字符集合中的任意一个字符                '[^ab]'：匹配任何不包含a或b的字符串

p1|p2|p3  匹配 p1 或 p2 或 p3                           'z|food' ：匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。

\*         匹配零个或多个在它前面的字符                   zo\* ：匹配 "z" 以及 "zoo"。\* 等价于{0,}。

+         匹配前面的字符1次或多次                       'zo+' ：匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。

{n}       匹配前面的字符串至少n次，n 是一个非负整数      'o{2}' ：匹配 "food" 中的两个o，但不能匹配 "Bob" 中的 'o'

{n,m}     匹配前面的字符串至少n次，至多m次，m 和 n 均为非负整数，其中n <= m       'o{2,4}' ：匹配至少2个o，最多4个o的字符串。如oo，oooo

--查询Users表中uPhone以“5”结尾用户的姓名，性别和电话

SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users WHERE uphone REGEXP '5$';

--查询Users表中uPhone以“16,17,18”开头用户的姓名，性别和电话

SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users WHERE uphone REGEXP '^16|17|18';

--数据排序

ORDER BY {列名 | 表达式 | 正整数} [ASC | DESC] [,…n]

--查询Goods表tID为1的商品编号、名称和价格，并按价格升序排列

SELECT gdCode AS 商品编号 ,gdName AS 商品名称,gdPrice AS 商品价格 FROM goods WHERE tID=1 ORDER BY gdPrice;

--查询Goods表tID为1的商品编号、名称、价格和销售量，并先按销售量降序，再价格升序排列

SELECT gdCode AS 商品编号 ,gdName AS 商品名称,gdPrice AS 商品价格,gdSaleQty AS 销售量 FROM goods WHERE tID=1 ORDER BY gdSaleQty DESC,gdPrice;

--使用LIMIT限制结果集返回的行数

--SELECT语句中使用LIMIT子句来指定查询结果从哪一条记录开始以及一共查询多少行记录

--查询Goods表前3行记录的商品编号、名称和价格。

SELECT gdCode AS 商品编号,gdName AS 商品名称,gdPrice AS 商品价格 FROM goods LIMIT 3;

--聚合函数

-- 函数名      说明

SUM            返回表达式中所有值的和

MAX            返回表达式中的最大值

COUNT          返回组中的项数

AVG            返回组中各值的平均值

MIN            返回表达式中的最小值

GROUP\_CONCAT   返回一个字符串结果，该结果由分组中的值连接组合而成

-- 查询Goods表，统计所有商品的总销售量。

SELECT SUM(gdSaleQty) FROM goods;

-- 查询Goods表，显示商品的最高价格。

SELECT MAX(gdprice) FROM goods;

-- 查询Orders表，显示购买过商品的用户人数。

SELECT COUNT(DISTINCT uID) FROM orders;

--数据分组统计GROUP BY子句

GROUP BY [ ALL ] 列名1,列名2 [ ,...n ] [ WITH ROLLUP] [HAVING 条件表达式]

--GROUP BY和GROUP\_CONCAT函数一起使用

--DISTINCT 可以排除重复值

GROUP\_CONCAT([DISTINCT] 表达式 [ORDER BY 列名] [SEPARATOR 分隔符])

-- 查询Users表，按uCity列进行分组

SELECT uID,uName,uSex,uCity FROM users GROUP BY uCity;

-- 查询Users表，统计各城市的用户人数。

SELECT uCity,COUNT(\*) FROM users GROUP BY uCity;

-- 查询Users表，将同一城市的uID值用逗号“,”连接起来，列名为uIDs。

SELECT uCity,GROUP\_CONCAT(uID) AS uIDs FROM users GROUP BY uCity;

-- 查询Users表，将同一城市的uID值用下划线“\_”连接起来，列名为uIDs。

SELECT uCity,GROUP\_CONCAT(uid ORDER BY uid SEPARATOR '\_')  AS uids FROM users GROUP BY ucity;

--GROUP BY和WITH ROLLUP一起使用，可以输出每一类分组的汇总值

-- 查询Users表，统计“上海”和“长沙”两地用户人数。

SELECT uCity,COUNT(\*) FROM users WHERE uCity IN ('长沙','上海') GROUP BY uCity WITH ROLLUP;

--GROUP BY和HAVING一起使用

--HAVING关键字和WHERE关键字都用于设置条件表达式对查询结果集进行筛选

--HAVING关键字后可以跟聚合函数，且只能跟GROUP BY一起使用

-- 查询Users表，统计各城市的用户人数，显示人数在3人以上的城市。

SELECT uCity,COUNT(\*) FROM users GROUP BY uCity HAVING COUNT(\*)>=3;

-- 视图操作

-- 插入视图

CREATE VIEW v\_student

AS

SELECT sno,sname,ssex FROM student;

-- 插入一条语句

INSERT INTO v\_student

VALUE('22140339','xmy','男');

-- 查询视图

SELECT \* FROM v\_student WHERE sno = '22140339';

-- 删除视图

-- 删除单个

DROP VIEW v\_student;

-- 删除多个

DROP VIEW v\_student,v\_app,v\_sst;

-- 删除视图指定内容

DELETE FROM v\_student WHERE sno = '2007050122';

-- 修改视图

ALTER VIEW v\_student

AS

SELECT sno,sname FROM student;

-- 查询20070304班选修大学英语的课程且成绩在80-90分的学生姓名，学号，班级号及成绩

CREATE VIEW v\_stu2

AS

SELECT sname,A.sno,classno,degree FROM

student A, sc B, course C

WHERE A.sno = B.sno AND B.cno = C.cno

AND classno = '20070304'

AND cname = '大学英语'

AND degree BETWEEN 80 AND 90;

-- 创建一个名为sc\_view1的视图，从数据库gradem的sc表中查询出成绩大于90分的所有学生选修课程成绩的信息。

CREATE VIEW sc\_view1

AS

SELECT \* FROM sc WHERE degree > 90;

-- 创建一个名为sc\_view2的视图，从数据库gradem的sc表中查询出成绩小于80分的所有学生的学号、课程号、成绩等信息。

CREATE VIEW sc\_view2

AS

SELECT sno,cno,degree FROM sc WHERE degree <80;

-- 创建一个名为sc\_view3的视图，由数据库gradem的student、course、sc表创建一个显示“20070304”班学生选修课程（包括学生姓名、课程名称、成绩等信息）的视图。

CREATE VIEW sc\_view3

AS

SELECT sname,cname,degree

FROM student a ,sc b,course c

WHERE a.sno = b.sno

AND b.cno = c.cno

AND classno = '20070304';

-- 创建一个从视图sc\_view1中查询出课程号“c01”的所有学生的视图。

CREATE VIEW view\_new

AS

SELECT \* FROM sc\_view1 WHERE cno='c01';

CREATE INDEX u\_1 ON users (uname);

SHOW INDEX FROM users;

-- 索引操作

-- 显示索引

SHOW INDEX FROM student;

-- 为商品类别表的名称字段建立普通索引。

CREATE INDEX i\_1 ON goodstype (tname);

-- 为用户表的用户名字段建立唯一索引UNIQUE。

CREATE UNIQUE INDEX u\_1 ON users(uname);

-- 为商品表的商品编号和商品名称创建普通的复合索引。

CREATE INDEX id\_1 ON goods (gdid,gdname);

-- 为用户表的电子邮箱建立索引，降序。

CREATE INDEX e\_1 ON users(uEmail DESC);

-- 删除以上索引。

DROP INDEX i\_1 ON goodstype;

DROP INDEX id\_1 ON goods;

DROP INDEX e\_1 ON users;

-- 事务处理操作

-- 开启事务

START TRANSACTION;

-- 将Alex用户的500元钱转给bill用户，转账

-- 失败ROLLBACK

START TRANSACTION;

UPDATE sh\_user SET money = money - 500 WHERE NAME = 'Alex';

UPDATE sh\_user SET money = money + 500 WHERE NAME = 'Bill';

ROLLBACK;

-- 成功commit;

START TRANSACTION;

UPDATE sh\_user SET money = money - 50 WHERE NAME = 'Alex';

UPDATE sh\_user SET money = money + 50 WHERE NAME = 'Bill';

COMMIT;

-- 开启事务

START TRANSACTION;

-- 录入订单信息sh\_order\_goods，购买120台笔记本电脑，id为4

INSERT INTO sh\_order\_goods(id,goods\_id,goods\_num)

VALUE (1,4,120);

-- 更新商品表sh\_goods，看看库存够不够

UPDATE sh\_goods SET stock = stock - 120 WHERE id = 4;

-- 确认事务（确认/回滚）

ROLLBACK;

-- 存储过程操作

-- 创建存储过程(//)($$)

DELIMITER//

CREATE PROCEDURE aaa()

BEGIN

SELECT sno,sname,classno,saddress FROM student WHERE saddress LIKE '%青岛%';

END//

-- 调用存储过程

CALL aaa;

-- rj347x中获取名字相同人的信息

DELIMITER//

CREATE PROCEDURE bbb()

BEGIN

SELECT 学号,姓名,城市 FROM rj347x

WHERE 姓名 IN (

SELECT 姓名 FROM rj347x

GROUP BY 姓名

HAVING COUNT(\*)>1

);

END//

CALL bbb;

-- 创建一个存储过程T1，统计某位同学的考试门数（参数in out）

-- in

DELIMITER //

CREATE PROCEDURE t1( IN T1\_sno CHAR(10))

BEGIN

SELECT COUNT(\*) FROM sc WHERE sno = T1\_sno;

END //

CALL t1('2007010101');

-- out

DELIMITER //

CREATE PROCEDURE t2( IN t\_sno CHAR(10),OUT t\_num INT)

BEGIN

SELECT COUNT(\*) INTO t\_num FROM sc WHERE sno = t\_sno;

END //

CALL t2('2007010101',@num);

SELECT @num;

-- 创建触发器T\_1,当向orderdetail表插入一条记录时，order表对应的ototal的值增加，增加的值为订单详细中对应商品的数量

 DELIMITER//

 CREATE TRIGGER T\_1

 AFTER INSERT ON orderdetail FOR EACH ROW

 BEGIN

 UPDATE orders SET oTotal = oTotal + new.odnum

 WHERE orders.oid = new.oid;

 END//

 INSERT INTO orderdetail

 VALUES (13,1,1,1,'sdfsdf',NOW());

-- ----------------------------------------------------------------------------------------------------------------

-- 1查询所有学生的基本信息、所有课程的基本信息和所有学生的成绩信息（用三条SQL语句）

SELECT \* FROM student;

SELECT \* FROM course;

SELECT \* FROM score;

-- 2查询所有学生的学号、姓名、性别和出生日期

SELECT 学号,姓名,性别, 出生日期 FROM student;

-- 3查询所有课程的课程名称

SELECT DISTINCT 课程名称 FROM course;

-- 4查询前10门课程的课号及课程名称

SELECT 课号,课程名称 FROM course LIMIT 10;

-- 5查询所有学生的姓名及年龄

SELECT 姓名,YEAR(now())-YEAR(出生日期) AS 年龄 FROM student;

-- 6查询所有年龄大于18岁的女生的学号和姓名

SELECT 姓名,学号 FROM student WHERE YEAR(NOW())-YEAR(出生日期)>18 AND 性别='女';

-- 7查询所有男生的信息

SELECT \* FROM student WHERE 性别='男';

-- 8查询所有任课教师的姓名和所在系别

SELECT 教师姓名,所在系别 FROM teacher;

-- 9查询“电子商务”专业的学生姓名、性别和出生日期

SELECT 姓名, 性别,出生日期 FROM student WHERE 专业='电子商务';

-- 10查询Student表中的所有系名

SELECT DISTINCT 系名 FROM student;

-- 11查询“C01”课程的开课学期

SELECT 开课学期 FROM teaching WHERE 序号='C01';

-- 12查询成绩在80～90分之间的学生学号及课号

SELECT 学生学号,课号 FROM score WHERE 成绩 BETWEEN 80 AND 90;

-- 13查询在1970年1月1日之前出生的男教师信息

SELECT \* FROM teacher WHERE 性别='男' AND 出生日期<'1970-01-01';

-- 14输出有成绩的学生学号

SELECT 学生学号 FROM score WHERE 成绩 IS NOT NULL;

-- 15查询所有姓“刘”的学生信息

SELECT \* FROM student WHERE 姓名 LIKE '%刘%';

-- 16查询生源地不是山东省的学生信息

SELECT \* FROM student WHERE 生源地 NOT LIKE '%山东省%';

-- 17查询成绩为79分、89分或99分的记录

SELECT \* FROM score WHERE 成绩 = 79 OR 成绩= 89 OR 成绩= 99;

-- 18查询名字中第二个字是“小”字的男生的学生姓名和地址

SELECT 姓名,地址 FROM student WHERE sname 姓名 '\_小%' AND 性别 = '男';

-- 19查询名称以“计算机”开头的课程名称

SELECT 课程名称 FROM course WHERE 课程名称 LIKE '^计算机';

-- 20查询计算机工程系和软件工程系的学生信息

SELECT \* FROM student WHERE 系别='计算机工程系' OR 系别='软件工程系';

-- 多表操作

-- 查询goods表中商品类别为服饰的商品id，名称，价格，销售数量

SELECT gdid,gdName,gdPrice,gdSaleQty,tname FROM

goods JOIN goodstype

ON goods.tid=goodstype.tid

WHERE goodstype.tid = 1;

-- 查询段湘林购买订单的总价格

SELECT uname AS 购买人,SUM(oTotal) FROM

orders JOIN users

ON orders.uid = users.uid

WHERE uname = '段湘林';

-- 查询谁没有买过东西

SELECT \* FROM users

WHERE uid NOT IN(SELECT DISTINCT uid FROM orders);

-- 查询与用户在同一城市的uname和uphone

SELECT uName,uPhone,ucity FROM

users WHERE ucity =

(SELECT ucity FROM users WHERE uname = '蔡准');

SELECT A.uname,A.uphone,A.ucity FROM

 users A JOIN users B

ON A.ucity = B.ucity

WHERE B.uname = '蔡准';

-- 根据taobao.sql提供的数据，查询服饰类的商品卖出总数

SELECT SUM(gdSaleQty) AS 商品卖出总数 FROM

goods JOIN goodstype

ON goods.tid=goodstype.tid

WHERE goodstype.tname = '服饰';

-- 平均年龄

SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) FROM student;

-- 年龄大于平均年龄的学生姓名

SELECT sname FROM student

WHERE YEAR(CURDATE())-YEAR(sbirthday)

> (SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) FROM student);

-- 查询“LED小台灯"的2017年全年销售量

SELECT SUM(odNum) AS 全年销售量 FROM

orderdetail JOIN goods

ON orderdetail.gdid = goods.gdid

WHERE goods.gdname = 'LED小台灯';

-- 查询“LED小台灯"的2017年全年销售量

SELECT SUM(odnum) AS 全年销售量 FROM

orders,orderdetail,goods

WHERE orders.oID= orderdetail.oID AND goods.gdID= orderdetail.gdID

AND gdname='LED小台灯' AND YEAR(otime)=2017;

-- 查询购买过“LED小台灯"的用户姓名、联系电话、电子邮箱

SELECT uname AS 用户姓名,uphone AS 联系电话,uemail AS 电子邮箱

FROM users,goods,orderdetail,orders

WHERE goods.gdID=orderdetail.gdID

AND orderdetail.oID=orders.oID

AND users.uID=orders.uID

AND gdname='LED小台灯';

-- 查询用户段湘林的所有订单信息

SELECT \* FROM orders

WHERE uid IN(

SELECT uid FROM users WHERE uname = '段湘林');

-- 查询2017年全年购买金额在1000元以上用户信息，显示用户名、性别和联系电话

SELECT uname AS 用户名,usex AS 性别,uphone AS 联系电话,SUM(ototal) AS 购买金额

FROM users,orders

WHERE users.uid = orders.uid

AND YEAR(otime) = 2017

GROUP BY orders.uID

HAVING SUM(ototal)>1000;

-- 按性别统计，2017年全年男性和女性分别购买商品的订单总价

SELECT usex AS 性别,SUM(ototal) AS 商品订单总价

FROM users JOIN orders

ON users.uID=orders.uID

WHERE YEAR(otime)=2017 GROUP BY usex;

SELECT usex,SUM(ototal)

FROM users,orders

WHERE users.uID=orders.uID

AND YEAR(otime)=2017

GROUP BY usex;

-- 统计各种类别商品2017年全年卖出的总数，显示类别名称和数量，并按数量从高到低排序

SELECT tname AS 类别名称,SUM(odnum) AS 卖出总数量

FROM goods,goodstype,orders,orderdetail

WHERE goods.tID=goodstype.tID

AND orders.oID=orderdetail.oID

AND goods.gdID=orderdetail.gdID

AND YEAR(otime)=2017

GROUP BY goods.tID

ORDER BY SUM(odnum) DESC;

-- 多表操作2

-- (1) 查询电子工程系女学生的学生学号、姓名及考试成绩。

SELECT a.sno AS 学号,sname AS 姓名,degree as成绩 FROM

student a JOIN sc b ON a.sno = b.sno

WHERE  ssex = '女'

AND sdept = '电子工程系';

--

SELECT a.sno,sname,SUM(degree) FROM

student a JOIN sc b ON a.sno = b.sno

WHERE  ssex = '女'

AND sdept = '电子工程系'

GROUP BY sname;

--查询计算机工程系女生的学生学号、姓名及考试成绩:SELECT A.sno 学号,sname 姓名,degree 成绩

SELECT A.sno 学号,A.sname 姓名,B.degree 考试成绩

FROM student A,sc B

WHERE A.sno=B.sno AND ssex='女' AND sdept='计算机工程系';

-- (2) 查询“闫旭光”同学所选课程的成绩。(不考虑重名)

SELECT cname AS 姓名,degree AS 成绩 FROM

student a JOIN sc b ON a.sno = b.sno

  JOIN course c ON c.cno = b.cno

  WHERE sname = '闫旭光';

--查询“自己”所选课程的名称、成绩。

SELECT sname 姓名,cno 课程,degree 成绩

FROM student A,sc B

WHERE A.sno=B.sno AND sname='吴兵';

-- (3) 查询“李新”老师所授课程的课程名称。

SELECT cname AS 课程名称 FROM

course a JOIN teaching b ON a.cno = b.cno

    JOIN teacher c ON c.tno = b.tno

WHERE Tname = '李新';

-- 查询“李新”教师所授课程的课程名称。

SELECT tname 姓名,cname 课程

FROM teacher A,course B,teaching C

WHERE A.tno=C.tno AND B.cno=C.cno AND tname='李新';

-- (4) 查询女教师所授课程的课程号及课程名称。

SELECT a.cno AS 课程号,cname AS 课程名称 FROM

course a JOIN teaching b ON a.cno = b.cno

    JOIN teacher c ON c.tno = b.tno

WHERE Tsex = '女';

-- 查询女教师所授课程的课程号及课程名称:

SELECT tname 姓名,cname 课程,B.cno 课程号

FROM teacher A,course B,teaching C

WHERE A.tno=C.tno AND B.cno=C.cno AND tsex='女';

-- (5) 查询至少选修一门课程的女学生姓名。

SELECT sname AS 姓名 FROM

student a JOIN sc b ON a.sno = b.sno

WHERE ssex = '女'

GROUP BY a.sno

HAVING COUNT(cno)>1;

-- 查询至少选修了一门课程的女生姓名，

SELECT sname

FROM student A,sc B

WHERE A.sno=B.sno AND ssex='女'

GROUP BY A.sno

HAVING COUNT(cno)>=1;

-- (6) 查询姓“王”的学生所学的课程名称。

SELECT a.sname AS 姓名,cname AS 课程名称 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN course c ON c.cno = b.cno

WHERE sname LIKE '王%';

-- 查询姓“王”的学生所学的课称名称。

SELECT sname 姓名,cname 课程

FROM student A,course B,sc C

WHERE A.sno=C.sno AND B.cno=C.cno AND sname LIKE '%王%';

-- (7) 查询选修“高等数学”课程且成绩在80～90分之间的学生学号及成绩。

-- (8) 查询选修“高等数学”课程且成绩在80～90分之间的学生姓名及成绩。

SELECT a.sno AS 学号,degree AS 成绩 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN course c ON c.cno = b.cno

WHERE cname = '高等数学'

HAVING degree BETWEEN 80 AND 90;

-- 查询选修“数据库”课程且成绩在80-90分的学生学号及成绩

SELECT sno 学号,degree 成绩

FROM sc A,course B

WHERE  A.cno=B.cno AND B.cname LIKE '%数据库%'

HAVING degree BETWEEN 80 AND 90;

-- (9) 查询课程成绩及格的男同学的姓名及课程号与成绩。

SELECT sname AS 姓名,c.cno AS 课程号,degree AS 成绩 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN course c ON c.cno = b.cno

WHERE ssex = '男'

AND degree>60;

-- 查询课程成绩及格的男生的学生信息、课程号与成绩。

SELECT A.\*,cno 课程号,degree 成绩

FROM student A,sc B

WHERE A.sno=B.sno AND degree>60 AND ssex='男';

-- (10) 查询选修“c04”课程的学生的平均年龄。

SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) AS 平均年龄 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN course c ON c.cno = b.cno

WHERE c.cno = 'c04';

-- 查询选修“C04”课程的学生的平均年龄。

SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) 平均年龄

FROM student a,sc b

WHERE a.sno=b.sno AND cno='c04';

-- (11) 查询学习课程名为“大学英语”的学生学号和姓名。

SELECT a.sno AS 学号,sname AS 姓名 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN course c ON c.cno = b.cno

WHERE cname = '大学英语';

-- 查询选修课程名为“数学”的学生学号和姓名。

SELECT A.sno 学号,sname 姓名

FROM student A,sc B,course C

WHERE A.sno=B.sno AND B.cno=C.cno AND cname LIKE '%数学';

-- (12) 查询“钱军”教师任课的课程号，选修其课程的学生的学号和成绩。

SELECT c.tno AS 课程号 ,a.sno AS 学号,degree AS 成绩 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN teaching c ON c.cno = b.cno

JOIN teacher d ON d.tno = c.tno

WHERE tname = '钱军';

-- 查询“钱军”教师任课的课程号、选修其课程的学生的学号和和成绩。

SELECT B.cno 课程号,cname 课程名,sno 学号,degree 成绩

FROM sc A,course B,teacher C,teaching D

WHERE A.cno=B.cno AND B.cno=D.cno AND C.tno=D.tno

AND tname='钱军';

-- (13)查询“钱军”教师任课的课程号，选修其课程的学生的姓名。

SELECT c.tno AS 课程号 ,sname AS 学生姓名 FROM

student a JOIN sc b ON a.sno = b.sno

JOIN teaching c ON c.cno = b.cno

JOIN teacher d ON d.tno = c.tno

WHERE tname = '钱军';

-- (14)查询在第1学期所开课程的课程名称及成绩。

SELECT cname AS 课程名称,degree AS 成绩 FROM

teaching a JOIN sc b ON  a.cno = b.cno

JOIN course c ON c.cno = b.cno

WHERE cterm ='1';

-- 查询在第1学期所开课程的课程名称及学生的成绩。

SELECT degree,cname

FROM teaching a,sc b,course c

WHERE a.cno=b.cno AND b.cno=c.cno

AND cterm='1'

GROUP BY sno;

-- (15)查询“c02”号课程不及格的学生信息。

SELECT \* FROM

sc a JOIN student b ON a.sno=b.sno

WHERE cno='C02'

HAVING degree<60;

-- 查询“C02”课程不及格的学生信息。

SELECT \*

FROM sc a,student b

WHERE a.sno=b.sno AND cno='C02'

HAVING degree<60;

-- (16)查询软件工程系成绩在90分以上的学生姓名、性别和课程名称。

SELECT sname,ssex,cname,deptname,degree FROM

department a JOIN class b ON a.deptno = b.deptno

JOIN student c ON c.classno = b.classno

JOIN sc d ON d.sno = c.sno

JOIN course e ON e.cno = d.cno

WHERE degree>90

 AND deptname = '软件工程系';

--

SELECT sname,ssex,cname FROM

student a JOIN sc b ON a.sno = b.sno

 JOIN course c ON c.cno = b.cno

 WHERE degree>90

 AND sdept = '软件工程系';

 -- 查询信息工程系成绩在90分以上的学生姓名、性别和课程名称。

SELECT sname 姓名,ssex 性别,cname 课程名称

FROM sc a,student b,course c

WHERE a.sno=b.sno AND a.cno=c.cno

AND sdept='信息工程系'

AND degree>90;

 -- (17) 查询同时选修了“c04”和“c02”课程的学生姓名。

SELECT sname AS 姓名 FROM

student a JOIN sc b ON a.sno=b.sno

JOIN sc c ON c.sno = b.sno

WHERE b.cno='C04' AND c.cno='C02'

GROUP BY sname;

-- 查询同时选修了“C04”和“C02”课程的学生姓名和成绩。

SELECT sname 姓名,b.degree 成绩,c.degree 成绩

FROM student a,sc b,sc c

WHERE a.sno=b.sno AND b.sno=c.sno

AND b.cno='C04' AND c.cno='C02'

GROUP BY sname;

-- 查询所有平均年龄大于平均年龄的学生姓名和年龄gradem1

SELECT sname AS 姓名, YEAR(NOW()) - YEAR(sbirthday) AS 年龄 FROM student

WHERE YEAR(NOW()) - YEAR(sbirthday) >

(SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) FROM student);

-- 查询没有选修高等数学的学生学号和姓名gradem1

SELECT sno,sname FROM student

WHERE sno NOT IN

(SELECT sno FROM sc WHERE cno IN(

SELECT cno FROM course WHERE cname = '高等数学'

));

-- 查询其他系中比计算机工程系某一学生年龄小的学生姓名和年龄gradem

SELECT sname,YEAR(NOW())-YEAR(sbirthday) AS 年龄 FROM student

WHERE YEAR(NOW())-YEAR(sbirthday)< ANY(

SELECT YEAR(NOW())-YEAR(sbirthday) FROM student WHERE sdept = '计算机工程系')

AND sdept <> '计算机工程系';

-- 子查询操作

-- 查询所有选修了c01号课程的学生姓名gradem1

SELECT sname FROM student

WHERE EXISTS (

SELECT \* FROM sc WHERE sno = student.sno

AND cno ='c01'

);

-- 查询选修了全部课程的学生姓名gradem1

SELECT sname FROM student

WHERE NOT EXISTS(SELECT \* FROM course WHERE NOT EXISTS(

SELECT \* FROM sc WHERE sno=student.sno AND cno=course.cno)

);

-- 查询张忠同学所选课程的成绩gradem

SELECT degree FROM sc

WHERE sno IN (

SELECT sno FROM student WHERE sname = '张忠'

);

-- 子查询“李新”老师所授课程的课程名称gradem1

SELECT cname FROM course

WHERE cno IN (SELECT cno FROM teaching WHERE tno IN(SELECT tno FROM teacher WHERE tname = '李新')

);

-- 子查询方法，查询女教师所授课程的课程号及课程名称

SELECT cno,cname FROM course

WHERE cno IN(SELECT cno FROM teaching

WHERE tno IN(SELECT tno FROM teacher WHERE tsex='女'));

-- （1）查询“李佳丽”同学所选课程的成绩

SELECT degree FROM sc

WHERE sno IN (

SELECT sno FROM student WHERE sname = '李佳丽'

);

-- （2）查询“李新”老师所授课程的课程名称。

SELECT cname FROM course

WHERE cno IN (SELECT cno FROM teaching WHERE tno IN(SELECT tno FROM teacher WHERE tname = '李新')

);

-- （3）查询女教师所授课程的课程号及课程名称。

SELECT cno,cname FROM course

WHERE cno IN(SELECT cno FROM teaching

WHERE tno IN(SELECT tno FROM teacher WHERE tsex='女'));

-- （4）查询姓“王”的学生所学的课程名称。

SELECT cname FROM course

WHERE cno IN (SELECT cno FROM sc

WHERE sno IN(SELECT sno FROM student

WHERE sname LIKE '王%'));

-- （5）查询“C01”课程不及格的学生信息。

SELECT \* FROM student

WHERE sno IN (SELECT sno FROM sc WHERE degree <60 AND cno='c01');

-- （6）查询选修“高等数学”课程且成绩在80～90分的学生学号及成绩。

SELECT sno,degree FROM sc

WHERE cno IN(SELECT cno FROM course

WHERE cname='高等数学') AND degree BETWEEN 80 AND 90;

-- （7）查询选修“C04”课程的学生的平均年龄。

SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) AS 平均年龄 FROM

student WHERE sno IN(SELECT sno FROM sc WHERE cno = 'c04');

-- （8）查询选修课程名为“高等数学”的学生学号和姓名。

SELECT sno,sname FROM student

WHERE sno IN(SELECT sno FROM sc

WHERE cno IN(SELECT cno FROM course WHERE cname='高等数学'));

-- （9）查询“钱军”教师任课的课程号，选修其课程的学生的学号和成绩。

SELECT tno,sno,degree FROM sc,teacher

WHERE cno IN(SELECT cno FROM teaching

WHERE tno IN(SELECT tno FROM teacher WHERE tname='钱军'));

-- （10）查询在第3学期所开课程的课程名称及学生的成绩。

SELECT cname,degree FROM sc,course

WHERE sno IN(SELECT sno FROM teaching WHERE

cno IN(SELECT cno FROM course WHERE teaching.cterm = 3));

-- （11）查询与“李佳丽”同一个系的同学姓名。

SELECT sname FROM student

WHERE sdept IN(SELECT sdept FROM student WHERE sname='李佳丽');

-- （12）查询学号比“李佳丽”同学大，而出生日期比她小的学生姓名。

SELECT sname FROM student

WHERE YEAR(NOW())-YEAR(sbirthday)< ALL(SELECT YEAR(NOW())-YEAR(sbirthday) FROM student

WHERE sname='李佳丽') AND sno>ALL (SELECT sno FROM student WHERE sname='李佳丽');

-- （13）查询出生日期大于所有女同学出生日期的男同学的姓名及系别。

SELECT sname,sdept FROM student

WHERE sbirthday > ALL (SELECT sbirthday FROM student WHERE ssex='女')

AND ssex='男';

-- （14）查询成绩比该课程平均成绩高的学生的学号及成绩。

SELECT sno,degree FROM sc a

WHERE degree > ALL (SELECT AVG(degree) FROM sc b WHERE a.sno = b.sno);

-- （15）查询不讲授“C01”课的教师姓名。

SELECT tname FROM teacher

WHERE tno NOT IN (SELECT tno FROM teaching

WHERE cno='c01');

-- （16）查询没有选修“C02”课程的学生学号及姓名。

SELECT sno,sname FROM student

WHERE sno NOT IN (SELECT sno FROM sc WHERE cno='C02');

-- （17）查询选修了“高等数学”课程的学生学号、姓名及系别。

SELECT sno,sname,sdept FROM student

WHERE sno IN (SELECT sno FROM sc

WHERE cno IN(SELECT cno FROM course WHERE cname='高等数学'));

--

-- 把平均成绩大于80分的学生的学号和平均成绩存入另一个已知的基本表s\_grade(sno,avg\_grade)中

INSERT INTO s\_grade(sno,avg\_grade)

SELECT sno,AVG(degree) FROM sc

GROUP BY sno

HAVING AVG(degree)>80;

-- 挑选出全班的女生信息，放在新表girls

CREATE TABLE girls LIKE student;

INSERT INTO girls

SELECT \* FROM student

WHERE ssex = '女';

-- 将电子工程系全体学生的成绩设置为0

UPDATE sc SET degree=0

WHERE sno IN(SELECT sno FROM student WHERE sdept = '电子工程系');

-- 删除王小龙的信息

DELETE FROM student

WHERE sname = '王小龙';

-- （1）向student表中插入记录（"2005010203"，"张静"，"女"，"1981-3-21"，"软件工程系"，"软件技术"）

INSERT INTO student(sno,sname,ssex,sbirthday,sdept,speciality) VALUE

("2005010203","张静","女","1981-3-21","软件工程系","软件技术");

-- （2）插入学号为“2005010302”、姓名为“李四”的学生信息

INSERT INTO student(sno,sname)

VALUE ('2005010302','李四');

-- （3）把电子工程系的学生记录保存到表TS中（TS表已存在，表结构与student表相同）

CREATE TABLE ts LIKE student;

INSERT INTO ts

SELECT \* FROM student

WHERE sdept = '电子工程系';

-- （4）将学号为“2005010202”的学生姓名改为“张华”，系别改为“电子工程系”，专业改为“电子应用技术”。

UPDATE student SET sname='张华',sdept='电子工程系',speciality='电子应用技术'

WHERE sno ='2005010202';

-- （5）将“李勇”同学的专业改为“计算机信息管理”。

UPDATE student SET speciality='计算机信息管理'

WHERE sname ='李勇';

-- （6）删除学号为“2005010302”的学生记录。

DELETE FROM student

WHERE sno ='2005010302';

-- （7）删除“计算机工程系”所有学生的选课记录。

DELETE FROM student

WHERE sno IN(SELECT sno FROM student WHERE sdept = '计算机工程系');

-- （8）删除sc表中尚无成绩的选课记录。

DELETE FROM sc

WHERE degree IS NULL;

-- （9）把“刘晨”同学的选修记录全部删除。

DELETE FROM student

WHERE sname = '刘晨';

-- 视图

-- 1.创建视图v\_goods，显示商品ID，商品名称，商品价格，库存数量。并显示视图中数据。

CREATE VIEW v\_goods

AS

SELECT gdID,gdName,gdPrice,gdQuantity FROM goods;

-- 2.创建视图v\_orders，显示订单编号、会员姓名、商品名称、总金额信息。并显示视图中数据。

CREATE OR REPLACE VIEW v\_orders(订单编号,会员姓名,商品名称,总金额信息)

AS

SELECT c.oid,uName,gdName,oTotal FROM

users a,goods b,orders c,orderdetail d

WHERE a.uid = c.uid

AND c.oid = d.oid

AND b.gdid = d.gdid;

SELECT \* FROM v\_orders;

-- 3.使用UPDATE语句更新视图v\_goods，将所有商品单价提升10%。并显示视图中数据。

UPDATE v\_goods SET gdPrice = gdPrice\*1.1;

SELECT \* FROM v\_goods;

-- 4.使用DELETE语句，更新视图v\_goods，删除goods表中最后一行。并显示视图中数据

DELETE FROM v\_goods ORDER BY gdid DESC LIMIT 1;

SELECT \* FROM v\_goods;

-- 5.创建视图v3，显示没有购买商品的编号，用户名、电话信息。并显示视图中数据

CREATE OR REPLACE VIEW v3(编号,用户名,电话信息)

AS

SELECT uid,uname,uphone FROM users

WHERE uid NOT IN(

SELECT uid FROM orders);

SELECT \* FROM v3;

-- 6.删除以上创建的所有视图

DROP VIEW v\_goods;

DROP VIEW v\_orders;

DROP VIEW v3;

-- 储存过程

CREATE TABLE w(name1 CHAR(3),score INT);

INSERT w VALUE('xxx',82);

DELIMITER $$

CREATE PROCEDURE pw(IN mz CHAR(3),IN fen INT)

BEGIN

INSERT w VALUE(mz,fen);

SELECT \* FROM w;

END $$

CALL pw('bbb',100);

-- rj347x中获取名字相同人的信息

SELECT 学号,姓名,城市 FROM rj347x

WHERE 姓名 IN (

SELECT 姓名 FROM rj347x

GROUP BY 姓名

HAVING COUNT(\*)>1

);

-- 查询李珍珍购买的商品

DELIMITER//

CREATE PROCEDURE ccc()

BEGIN

SELECT gdname FROM users a,scar b,goods c

WHERE uname = '李珍珍'

AND a.uid = b.uid

AND b.gdid =c.gdid;

END//

CALL ccc;

-- 创建一个从SC表中查询某一门课程考试成绩总分的存储过程K2。

DELIMITER //

CREATE PROCEDURE k2(IN K2\_cno CHAR(3))

BEGIN

SELECT SUM(degree) FROM sc WHERE cno = K2\_cno;

END //

CALL k2('c01');

-- 创建存储过程k3,从student表查询通过输入班级号参数传递给存储过程，查看结果全班同学信息

DELIMITER //

CREATE PROCEDURE k3(IN k3\_classno CHAR(10))

BEGIN

SELECT \* FROM student WHERE classno = k3\_classno;

END //

CALL k3('20070101');

-- 创建一个从student表查询班级号为“20070301”班的学生资料的存储过程proc\_1，其中包括学号、姓名、性别、出生年月等。调用proc\_1存储过程，观察执行结果。

DELIMITER //

CREATE PROCEDURE proc\_1()

BEGIN

SELECT sno,sname,ssex,sbirthday FROM student WHERE classno = '20070301';

END //

CALL proc\_1();

-- 在gradem数据库中创建存储过程proc\_2，要求实现如下功能：存在不及格情况的学生选课情况列表，其中包括学号、姓名、性别、课程号、课程名、成绩、系别等。调用proc\_2存储过程，观察执行结果。

DELIMITER //

CREATE PROCEDURE proc\_2()

BEGIN

SELECT a.sno,sname,ssex,cno,degree,sdept FROM student a,sc b

WHERE degree<60

AND a.sno = b.sno;

END //

CALL proc\_2();

-- 创建存储过程spGetgoodsbygdID，根据商品ID查询指定的商品信息，显示gdCode，gdName，gdPrice，gdCity。

DELIMITER//

CREATE PROCEDURE spGetgoodsbygdID(IN goods\_id INT)

BEGIN

SELECT gdCode,gdName,gdPrice,gdCity FROM goods WHERE gdid = goods\_id;

END //

CALL spGetgoodsbygdID(4);

-- 创建存储过程spGetuIDbyuName，根据用户名返回用户ID（创建和调用带输入输出参数的存储过程）。

DELIMITER //

CREATE PROCEDURE spGetuIDbyuName(IN user\_name VARCHAR(20),OUT user\_id INT)

BEGIN

SELECT uid INTO user\_id FROM users WHERE uName = user\_name;

END//

CALL spGetuIDbyuName('段湘林',@uid);

SELECT @uid;

-- 为表sc创建一个插入触发器 student\_sc\_insert，当向表sc插入数据时，必须保证插入的学号有效地存在于student表中，

 -- 如果插入的学号在student表中不存在，插入新学号到student表。

 DELIMITER //

 CREATE TRIGGER student\_sc\_insert

 BEFORE INSERT ON sc FOR EACH ROW

 BEGIN

 IF(SELECT sno FROM student WHERE sno = new.sno) IS NULL

 THEN INSERT INTO student(sno) VALUES(new.sno);

 END IF;

 END//

 INSERT INTO sc VALUES('2007010199','a01',100);