|  |
| --- |
|  |

**数据库原理与实践期末项目**

**学号：**

**姓名：**



数据库原理与实践期末项目

20 24 —20 25 学年 一 学期

|  |  |
| --- | --- |
| 课程名称： | 数据库原理及应用 |
| 专业学院： | 现代技术学院 |
| 专业班级： | 计科B2215 |
| 学 号： |  |
| 姓 名： |  |
| 指导教师： | 李小蓉 |

目 录

[1、 项目背景及需求分析 1](#_Toc153742448)

[1.1项目背景 1](#_Toc153742449)

[1.2需求分析 1](#_Toc153742450)

[2、 概念结构设计 2](#_Toc153742451)

[2.1抽象出系统实体 2](#_Toc153742452)

[2.2局部E-R图 2](#_Toc153742453)

[2.3全局E-R图 2](#_Toc153742454)

[3、 逻辑结构设计 3](#_Toc153742455)

[3.1简单说明 3](#_Toc153742456)

[3.2关系模式 3](#_Toc153742457)

[3.3表结构 3](#_Toc153742458)

[4、 创建数据库及SQL相关操作 4](#_Toc153742459)

[4.1 创建数据库 4](#_Toc153742460)

[4.1.1 创建表 4](#_Toc153742461)

[4.1.2 添加记录 4](#_Toc153742462)

[4.1.3 修改记录 5](#_Toc153742463)

[4.2 删除记录 5](#_Toc153742464)

[4.3查询问题及查询结果 5](#_Toc153742465)

[4.3.1 选择列 5](#_Toc153742466)

[4.3.2函数使用 5](#_Toc153742467)

[4.3.3 LIKE子句 6](#_Toc153742468)

[4.3.4 条件查询 6](#_Toc153742469)

[4.4.5多表查询 6](#_Toc153742470)

[4.4.6子查询 6](#_Toc153742471)

[4.5建立视图 7](#_Toc153742472)

[4.5.1 定义视图 7](#_Toc153742473)

[4.5.2利用视图插入数据 7](#_Toc153742474)

[5、建立存储过程及函数 8](#_Toc153742475)

[5.1创建存储过程 8](#_Toc153742476)

[5.2存储过程应用 8](#_Toc153742477)

[6、建立触发器 9](#_Toc153742478)

[7、项目设计总结 10](#_Toc153742479)

# 1、 项目背景及需求分析

## 1.1项目背景

题目：健身房信息管理系统

健身房会员管理系统是为了方便线下门店管理业务，提高工作效率而设计的。具有会员，教练，工作人员三种角色，每个角色有相应的功能，让会员可以方便的购买课程，教练可以发布和查看自己的教学课程，工作人员可以在后台查看这些数据。总的来说，这个系统极大的方便了健身房的会员管理，具有很强的现实意义。

## 1.2需求分析

需求分析如图：



图2.1 会员用例图



图2.2 教练用例图



图2.3 员工用例图

# 2、 概念结构设计

## 2.1抽象出系统实体

实体包括：会员，课程、教练、健身房。

## 2.2局部E-R图



图3.1 会员er图



图3.2 课程er图



图3.3 健身房er图



图3.4 教练er图

## 2.3全局E-R图



图3.5 总ER图

# 3、 逻辑结构设计

## 3.1简单说明

**健身房与教练（1对多）**:

* 一个健身房可以拥有多个教练 (Gym.gym\_id -> Coach.gym\_id)。
* 一个教练只能隶属于一个健身房。

**健身房与课程（1对多）**:

* 一个健身房可以提供多个课程 (Gym.gym\_id -> Course.gym\_id)。
* 一个课程只能属于一个健身房。

**教练与课程（1对多）**:

* 一个教练可以教授多个课程 (Coach.coach\_id -> Course.coach\_id)。
* 一个课程只能由一个教练教授。

**会员与课程购买（1对多）**:

* 一个会员可以购买多个课程 (Member.member\_id -> Course\_Purchase.member\_id)。
* 一条购买记录对应一个课程。

**课程与课程购买（1对多）**:

* 一个课程可以被多个会员购买 (Course.course\_id -> Course\_Purchase.course\_id)。
* 一条购买记录对应一个课程。

## 3.2关系模式

**1. 健身房表（gym）**

* **描述**:
  + gym\_id: 健身房唯一标识符，主键。
  + name: 健身房名称。
  + phone: 健身房联系电话。
  + address: 健身房地址。
  + username: 健身房管理员的用户名。
  + password: 健身房管理员的密码。

**2. 教练表（coach）**

* **描述**:
  + coach\_id: 教练唯一标识符，主键，自增。
  + username: 教练用户名。
  + password: 教练密码。
  + phone: 教练联系电话。
  + gender: 教练性别，枚举值（Male, Female）。
  + birthdate: 教练出生日期。
  + coach\_type: 教练类型，枚举值（Regular, Premium）。
  + gym\_id: 所属健身房的外键，关联到 Gym 表的 gym\_id。

**3. 课程表（course）**

* **描述**:
  + course\_id: 课程唯一标识符，主键。
  + name: 课程名称。
  + description: 课程描述。
  + price: 课程价格（保留两位小数）。
  + max\_capacity: 课程的最大容量（人数上限）。
  + coach\_id: 外键，关联到 Coach 表的 coach\_id。
  + gym\_id: 外键，关联到 Gym 表的 gym\_id。

**4. 会员表（member）**

* **描述**:
  + member\_id: 会员唯一标识符，主键，自增。
  + username: 会员用户名。
  + password: 会员密码。
  + phone: 会员联系电话。
  + gender: 会员性别，枚举值（Male, Female）。
  + birthdate: 会员出生日期。
  + member\_type: 会员类型，枚举值（Regular, Premium）。

**5. 购买课程表（course\_purchase）**

* **描述**:
  + purchase\_id: 购买记录唯一标识符，主键，自增。
  + member\_id: 外键，关联到 Member 表的 member\_id。
  + course\_id: 外键，关联到 Course 表的 course\_id。
  + purchase\_date: 课程购买时间戳。
  + price: 购买时支付的课程价格（保留两位小数）。

## 3.3表结构

1. 健身房表（gym）

| 列名 | 数据类型 | 约束 | 描述 |
| --- | --- | --- | --- |
| gym\_id | INT | PRIMARY KEY | 健身房唯一标识符 |
| name | VARCHAR(20) | NOT NULL | 健身房名称 |
| phone | VARCHAR(20) | NOT NULL | 健身房联系电话 |
| address | VARCHAR(100) | NOT NULL | 健身房地址 |
| username | VARCHAR(20) | NOT NULL | 健身房管理员的用户名 |
| password | VARCHAR(20) | NOT NULL | 健身房管理员的密码 |

**2. 教练表（coach）**

| 列名 | 数据类型 | 约束 | 描述 |
| --- | --- | --- | --- |
| coach\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 教练唯一标识符 |
| username | VARCHAR(20) | NOT NULL | 教练用户名 |
| password | VARCHAR(20) | NOT NULL | 教练密码 |
| phone | VARCHAR(20) | NOT NULL | 教练联系电话 |
| gender | ENUM('Male', 'Female') | NOT NULL | 教练性别 |
| birthdate | DATE | NOT NULL | 教练出生日期 |
| coach\_type | ENUM('Regular', 'Premium') | NOT NULL | 教练类型（普通/高级） |
| gym\_id | INT | 外键 | 所属健身房，关联 gym.gym\_id |

3. 课程表（course）

| 列名 | 数据类型 | 约束 | 描述 |
| --- | --- | --- | --- |
| course\_id | INT | PRIMARY KEY | 课程唯一标识符 |
| name | VARCHAR(20) | NOT NULL | 课程名称 |
| description | VARCHAR(255) | NOT NULL | 课程描述 |
| price | DECIMAL(8, 2) | NOT NULL | 课程价格 |
| max\_capacity | INT | NOT NULL | 课程最大容量 |
| coach\_id | INT | 外键 | 教练，关联 coach.coach\_id |
| gym\_id | INT | 外键 | 健身房，关联 gym.gym\_id |

4. 会员表（member）

| 列名 | 数据类型 | 约束 | 描述 |
| --- | --- | --- | --- |
| member\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 会员唯一标识符 |
| username | VARCHAR(20) | NOT NULL | 会员用户名 |
| password | VARCHAR(20) | NOT NULL | 会员密码 |
| phone | VARCHAR(20) | NOT NULL | 会员联系电话 |
| gender | ENUM('Male', 'Female') | NOT NULL | 会员性别 |
| birthdate | DATE | NOT NULL | 会员出生日期 |
| member\_type | ENUM('Regular', 'Premium') | NOT NULL | 会员类型（普通/高级） |

5. 购买课程表（course\_purchase）

| 列名 | 数据类型 | 约束 | 描述 |
| --- | --- | --- | --- |
| purchase\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | 购买记录唯一标识符 |
| member\_id | INT | 外键 | 会员ID，关联 member.member\_id |
| course\_id | INT | 外键 | 课程ID，关联 course.course\_id |
| purchase\_date | TIMESTAMP | NOT NULL | 购买日期 |
| price | DECIMAL(8, 2) | NOT NULL | 购买时支付的课程价格 |

# 4、 创建数据库及SQL相关操作

## 4.1 创建数据库

建立数据库，数据库名称自取，设置字符集为UTF-8,排序规则为utf8\_general\_ci。

代码：

CREATE DATABASE gym

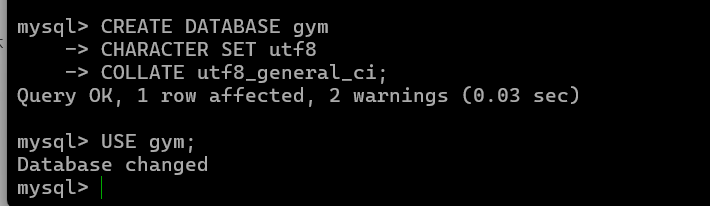
CHARACTER SET utf8

COLLATE utf8\_general\_ci;

-- 使用数据库

USE gym;

截图结果：



### 4.1.1 创建表

根据需要，在你们创建的数据库中创建你们的每个数据表。

-- 创建健身房表

CREATE TABLE gym (

gym\_id INT PRIMARY KEY,

name VARCHAR(20) NOT NULL,

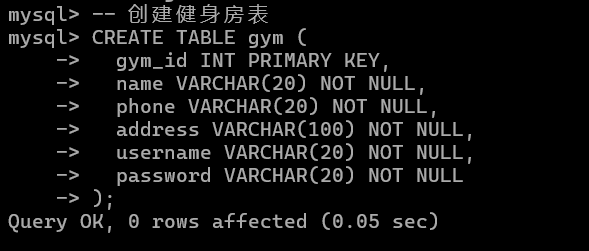
phone VARCHAR(20) NOT NULL,

address VARCHAR(100) NOT NULL,

username VARCHAR(20) NOT NULL,

password VARCHAR(20) NOT NULL

);



-- 创建教练表

CREATE TABLE coach (

coach\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(20) NOT NULL,

password VARCHAR(20) NOT NULL,

phone VARCHAR(20) NOT NULL,

gender ENUM('Male', 'Female') NOT NULL,

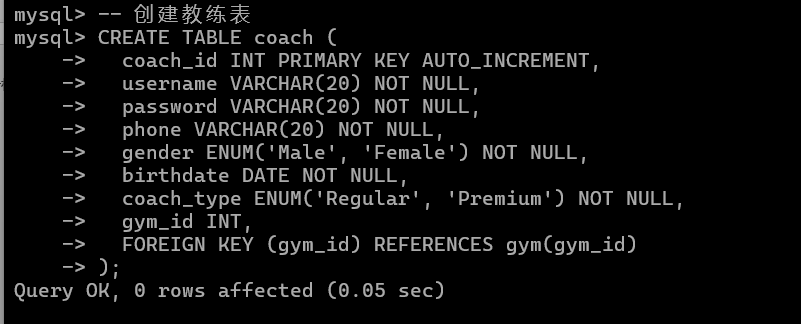
birthdate DATE NOT NULL,

coach\_type ENUM('Regular', 'Premium') NOT NULL,

gym\_id INT,

FOREIGN KEY (gym\_id) REFERENCES gym(gym\_id)

);



-- 创建课程表

CREATE TABLE course (

course\_id INT PRIMARY KEY,

name VARCHAR(20) NOT NULL,

description VARCHAR(255) NOT NULL,

price DECIMAL(8, 2) NOT NULL,

max\_capacity INT NOT NULL,

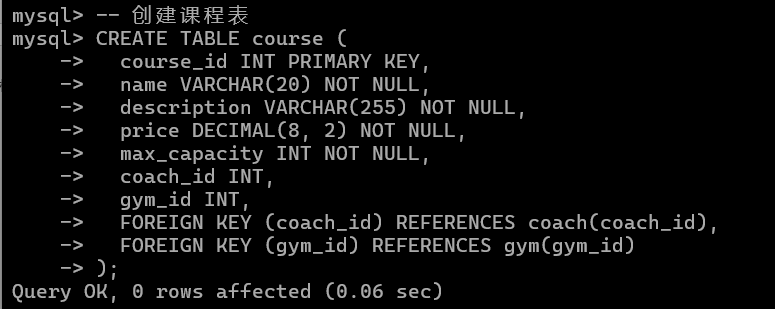
coach\_id INT,

gym\_id INT,

FOREIGN KEY (coach\_id) REFERENCES coach(coach\_id),

FOREIGN KEY (gym\_id) REFERENCES gym(gym\_id)

);



-- 创建会员表

CREATE TABLE member (

member\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(20) NOT NULL,

password VARCHAR(20) NOT NULL,

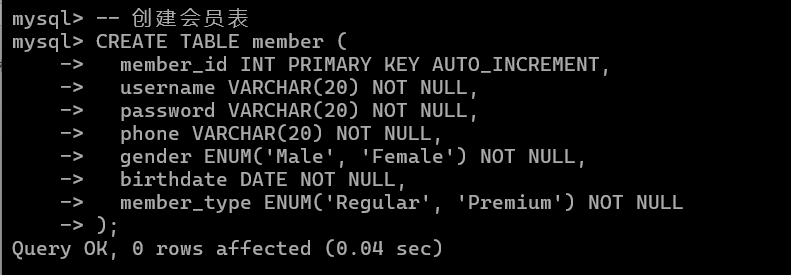
phone VARCHAR(20) NOT NULL,

gender ENUM('Male', 'Female') NOT NULL,

birthdate DATE NOT NULL,

member\_type ENUM('Regular', 'Premium') NOT NULL

);



-- 创建购买课程表

CREATE TABLE course\_purchase (

purchase\_id INT PRIMARY KEY AUTO\_INCREMENT,

member\_id INT,

course\_id INT,

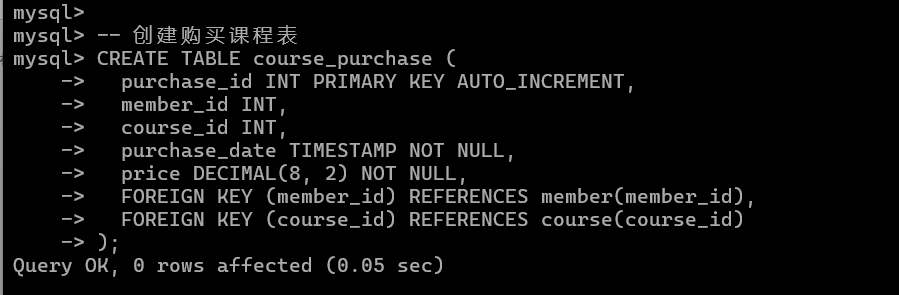
purchase\_date TIMESTAMP NOT NULL,

price DECIMAL(8, 2) NOT NULL,

FOREIGN KEY (member\_id) REFERENCES member(member\_id),

FOREIGN KEY (course\_id) REFERENCES course(course\_id)

);



### 4.1.2 添加记录

**-- 插入数据到 健身房表 (gym)**

INSERT INTO gym (gym\_id, name, phone, address, username, password) VALUES

(1, 'Fitness One', '1234567890', '123 Main St', 'admin1', 'pass1'),

(2, 'Power Gym', '1234567891', '456 Maple St', 'admin2', 'pass2'),

(3, 'Elite Gym', '1234567892', '789 Oak St', 'admin3', 'pass3'),

(4, 'ProFit', '1234567893', '101 Pine St', 'admin4', 'pass4'),

(5, 'Next Level', '1234567894', '202 Cedar St', 'admin5', 'pass5'),

(6, 'Flex Gym', '1234567895', '303 Birch St', 'admin6', 'pass6'),

(7, 'BodyWorks', '1234567896', '404 Ash St', 'admin7', 'pass7'),

(8, 'Iron Paradise', '1234567897', '505 Elm St', 'admin8', 'pass8'),

(9, 'Muscle Zone', '1234567898', '606 Cherry St', 'admin9', 'pass9'),

(10, 'BeFit Gym', '1234567899', '707 Walnut St', 'admin10', 'pass10');



**-- 插入数据到 教练表 (coach)**

INSERT INTO coach (username, password, phone, gender, birthdate, coach\_type, gym\_id) VALUES

('coach1', 'pass1', '9876543210', 'Male', '1990-01-01', 'Regular', 1),

('coach2', 'pass2', '9876543211', 'Female', '1988-05-12', 'Premium', 2),

('coach3', 'pass3', '9876543212', 'Male', '1995-07-24', 'Regular', 3),

('coach4', 'pass4', '9876543213', 'Female', '1987-03-15', 'Premium', 4),

('coach5', 'pass5', '9876543214', 'Male', '1992-09-10', 'Regular', 5),

('coach6', 'pass6', '9876543215', 'Female', '1993-12-30', 'Premium', 6),

('coach7', 'pass7', '9876543216', 'Male', '1991-11-18', 'Regular', 7),

('coach8', 'pass8', '9876543217', 'Female', '1989-06-08', 'Premium', 8),

('coach9', 'pass9', '9876543218', 'Male', '1994-02-22', 'Regular', 9),

('coach10', 'pass10', '9876543219', 'Female', '1990-10-05', 'Premium', 10);



**-- 插入数据到 课程表 (course)**

INSERT INTO course (course\_id, name, description, price, max\_capacity, coach\_id, gym\_id) VALUES

(1, 'Yoga', 'Beginner yoga class', 100.00, 15, 1, 1),

(2, 'Pilates', 'Core strength training', 120.00, 10, 2, 2),

(3, 'Zumba', 'Dance-based fitness class', 80.00, 20, 3, 3),

(4, 'HIIT', 'High-intensity interval training', 150.00, 12, 4, 4),

(5, 'Weightlifting', 'Strength training', 200.00, 8, 5, 5),

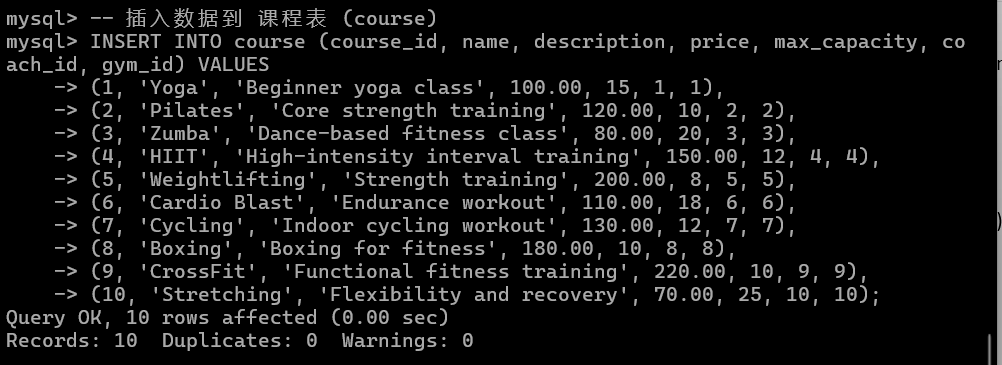
(6, 'Cardio Blast', 'Endurance workout', 110.00, 18, 6, 6),

(7, 'Cycling', 'Indoor cycling workout', 130.00, 12, 7, 7),

(8, 'Boxing', 'Boxing for fitness', 180.00, 10, 8, 8),

(9, 'CrossFit', 'Functional fitness training', 220.00, 10, 9, 9),

(10, 'Stretching', 'Flexibility and recovery', 70.00, 25, 10, 10);



**-- 插入数据到 会员表 (member)**

INSERT INTO member (username, password, phone, gender, birthdate, member\_type) VALUES

('member1', 'pass1', '5551112222', 'Male', '1990-01-15', 'Regular'),

('member2', 'pass2', '5551113333', 'Female', '1992-02-25', 'Premium'),

('member3', 'pass3', '5551114444', 'Male', '1995-03-10', 'Regular'),

('member4', 'pass4', '5551115555', 'Female', '1988-04-05', 'Premium'),

('member5', 'pass5', '5551116666', 'Male', '1993-05-20', 'Regular'),

('member6', 'pass6', '5551117777', 'Female', '1994-06-15', 'Premium'),

('member7', 'pass7', '5551118888', 'Male', '1991-07-30', 'Regular'),

('member8', 'pass8', '5551119999', 'Female', '1989-08-10', 'Premium'),

('member9', 'pass9', '5552220000', 'Male', '1992-09-25', 'Regular'),

('member10', 'pass10', '5552221111', 'Female', '1990-10-15', 'Premium');



**-- 插入数据到 购买课程表 (course\_purchase)**

INSERT INTO course\_purchase (member\_id, course\_id, purchase\_date, price) VALUES

(1, 1, '2023-01-01 10:00:00', 100.00),

(2, 2, '2023-01-05 14:00:00', 120.00),

(3, 3, '2023-01-10 18:00:00', 80.00),

(4, 4, '2023-01-15 12:00:00', 150.00),

(5, 5, '2023-01-20 08:00:00', 200.00),

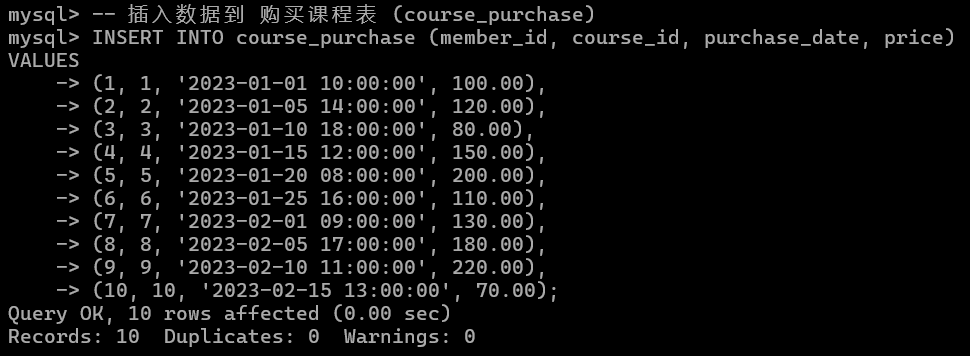
(6, 6, '2023-01-25 16:00:00', 110.00),

(7, 7, '2023-02-01 09:00:00', 130.00),

(8, 8, '2023-02-05 17:00:00', 180.00),

(9, 9, '2023-02-10 11:00:00', 220.00),

(10, 10, '2023-02-15 13:00:00', 70.00);



### 4.1.3 修改记录

-- 修改健身房 ID 为 1 的名称为 'New Fitness One' 和联系电话为 '9998887777'

UPDATE gym

SET name = 'New Fitness One', phone = '9998887777'

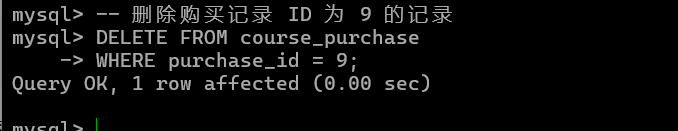
WHERE gym\_id = 1;



## 4.2 删除记录

-- 删除购买记录 ID 为 9 的记录

DELETE FROM course\_purchase

WHERE purchase\_id = 9; 

## 4.3查询问题及查询结果

### 4.3.1 选择列

自行设计查询单表中的几列，并用别名代替。

代码：

-- 查询课程的名称和价格，分别使用别名 "课程名称" 和 "价格"

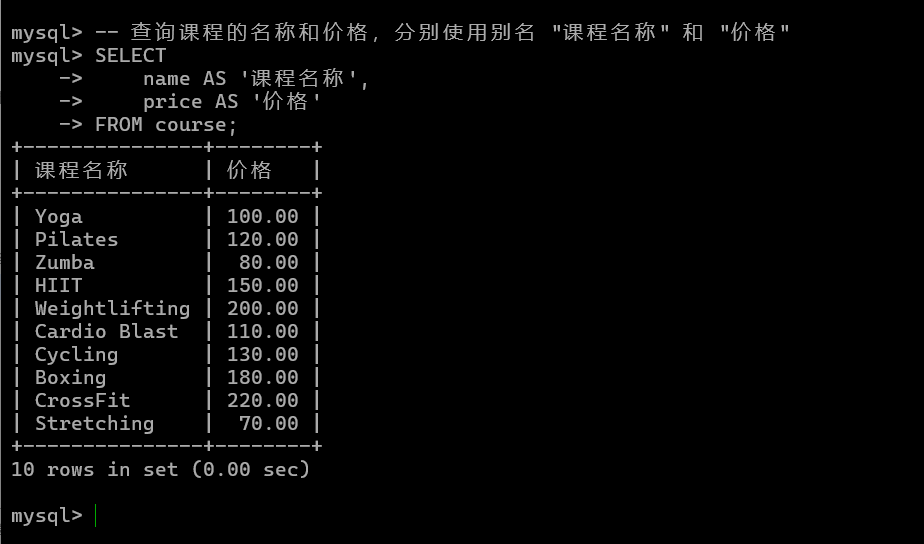
SELECT

name AS '课程名称',

price AS '价格'

FROM course;

结果截图：



### 4.3.2函数使用

计算满足条件的平均值，最大值，最小值。

代码：

-- 查询课程价格的平均值、最大值和最小值

SELECT

AVG(price) AS '平均价格',

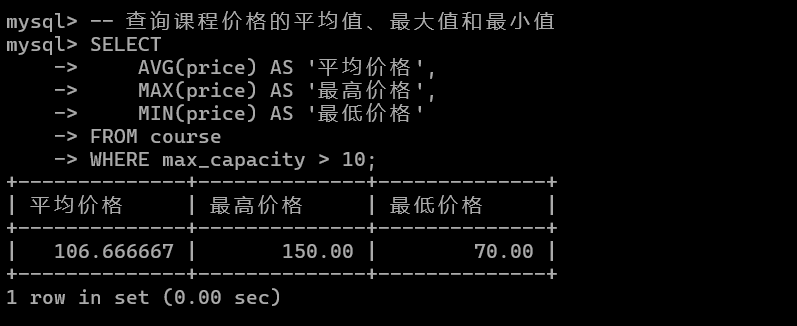
MAX(price) AS '最高价格',

MIN(price) AS '最低价格'

FROM course

WHERE max\_capacity > 10; -- 条件：课程的最大容量大于 10

结果截图：



### 4.3.3 LIKE子句

用LIKE子句实现模糊查询

说明实现的功能：

代码：

-- 查询用户名中包含 'member' 的会员

SELECT

member\_id,

username,

phone

FROM member

WHERE username LIKE '%member%';

结果截图：



### 4.3.4 条件查询

用给定条件，查询结果

说明实现的功能：

代码：

-- 查询符合条件的课程

SELECT

course\_id,

name,

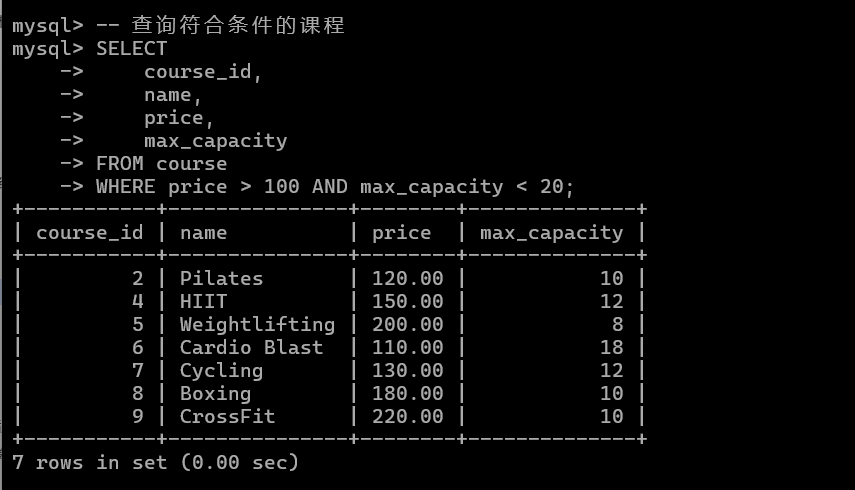
price,

max\_capacity

FROM course

WHERE price > 100 AND max\_capacity < 20;

结果截图：



### 4.4.5多表查询

至少牵扯两个表，根据给定条件，查询结果

说明实现的功能：

代码：

-- 查询购买记录中会员用户名和课程名称

SELECT

member.username AS '会员用户名',

course.name AS '课程名称',

course\_purchase.price AS '购买价格'

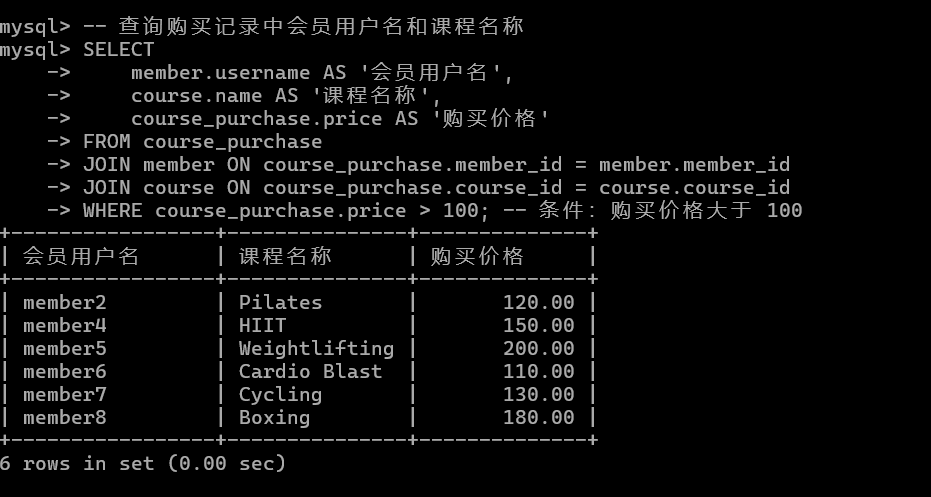
FROM course\_purchase

JOIN member ON course\_purchase.member\_id = member.member\_id

JOIN course ON course\_purchase.course\_id = course.course\_id

WHERE course\_purchase.price > 100; -- 条件：购买价格大于 100

结果截图：



### 4.4.6子查询

单行子查询举例

说明实现的功能：

代码：

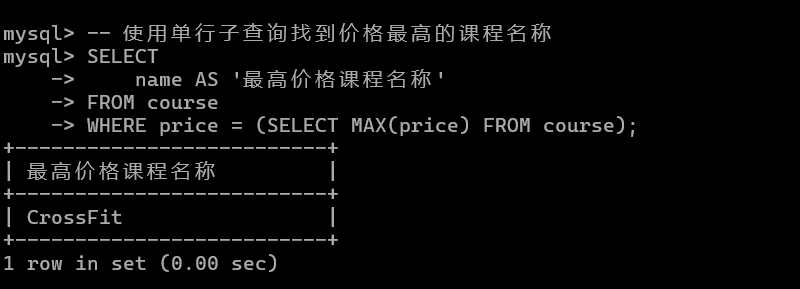
-- 使用单行子查询找到价格最高的课程名称

SELECT

name AS '最高价格课程名称'

FROM course

WHERE price = (SELECT MAX(price) FROM course);

结果截图：

多行子查询举例（用in，any,all任意一个均可）

说明实现的功能：

代码：

-- 查询价格高于所有 Yoga 类课程价格的课程

SELECT

course\_id,

name,

price

FROM course

WHERE price > ALL (

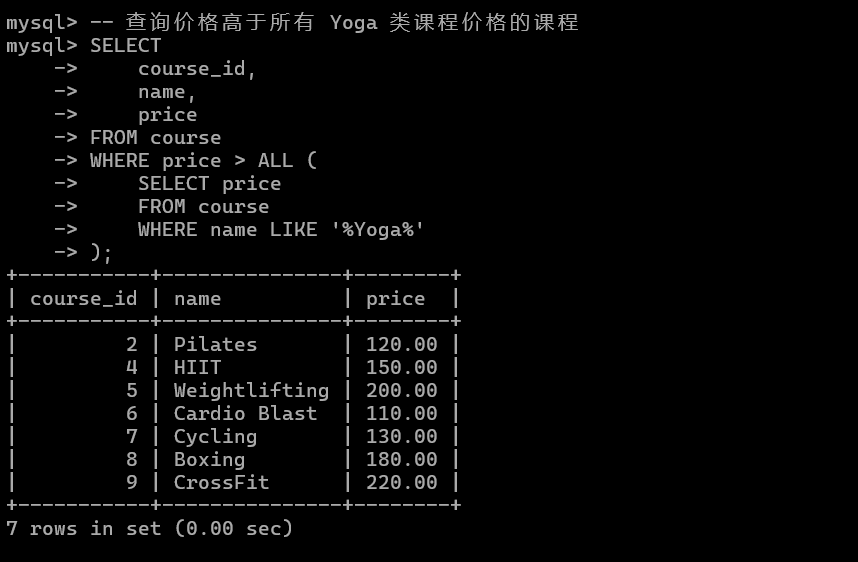
SELECT price

FROM course

WHERE name LIKE '%Yoga%'

);

结果截图：



## 4.5建立视图

建立一个用户user01，给user01账户授予建立视图的权限。

代码：

-- 创建用户 user01，设置密码为 'password123'

CREATE USER 'user01'@'localhost' IDENTIFIED BY 'password123';

-- 授予 user01 创建视图的权限

GRANT CREATE VIEW ON \*.\* TO 'user01'@'localhost';

-- 刷新权限

FLUSH PRIVILEGES;

结果截图：

### 4.5.1 定义视图

自定义一个视图，要求实现多表查询。

代码：

-- 创建视图，包含会员的用户名、电话、课程名称和价格

CREATE VIEW purchase\_summary AS

SELECT

member.username AS '会员用户名',

member.phone AS '会员电话',

course.name AS '课程名称',

course.price AS '课程价格'

FROM course\_purchase

JOIN member ON course\_purchase.member\_id = member.member\_id

JOIN course ON course\_purchase.course\_id = course.course\_id;

查询自定义的视图代码：

-- 查询自定义的视图

SELECT \* FROM purchase\_summary;

结果截图：



### 4.5.2利用视图插入数据

创建简单视图代码：

-- 创建一个简单的视图，展示会员的用户名和类型

CREATE VIEW member\_view AS

SELECT

username AS '会员用户名',

member\_type AS '会员类型'

FROM member;

利用视图向表中插入一条记录代码：

-- 向视图中插入一条记录

INSERT INTO member\_view (会员用户名, 会员类型)

VALUES ('test\_user', 'Regular');

查询插入的记录代码：

-- 查询刚插入的记录

SELECT \* FROM member\_view WHERE 会员用户名 = 'test\_user';查询结果截图：

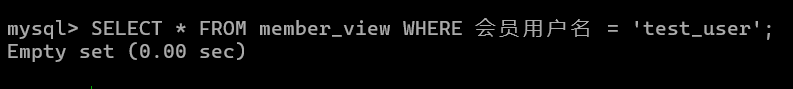
删除插入的结果代码：

-- 删除插入的记录

DELETE FROM member WHERE username = 'test\_user';

删除结果截图：

-- 确认删除的记录不存在

SELECT \* FROM member\_view WHERE 会员用户名 = 'test\_user'; 

# 5、建立存储过程及函数

## 5.1创建存储过程

创建一个存储过程，该存储过程实现输入某一个参数可以返回对应行上另外一个参数的值（如给定学号，返回选修课程信息等）。

代码：

-- 根据输入的 member\_id 返回对应的用户名。

DELIMITER $$

CREATE PROCEDURE GetMemberUsername(IN member\_id\_input INT, OUT username\_output VARCHAR(20))

BEGIN

SELECT username INTO username\_output

FROM member

WHERE member\_id = member\_id\_input;

END $$

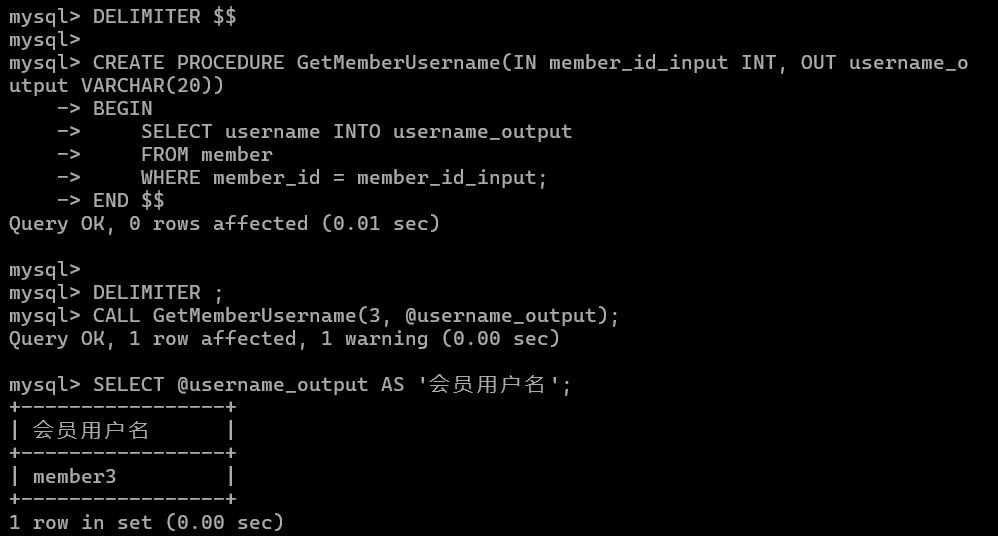
DELIMITER ;

执行存储过程代码：

CALL GetMemberUsername(3, @username\_output);

SELECT @username\_output AS '会员用户名';

结果截图：



## 5.2存储过程应用

创建一个函数过程，实现统计功能（统计个数或平均值等）。

代码：

-- 创建一个函数，统计 course 表中价格高于指定值的课程数量。

DELIMITER $$

CREATE FUNCTION CountExpensiveCourses(min\_price DECIMAL(8, 2))

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE course\_count INT;

SELECT COUNT(\*) INTO course\_count

FROM course

WHERE price > min\_price;

RETURN course\_count;

END $$

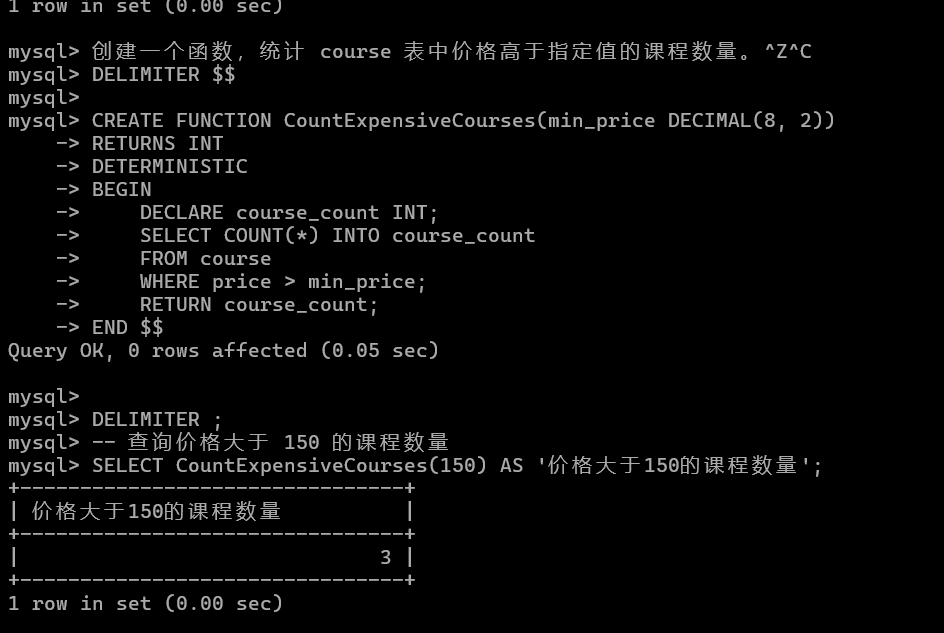
DELIMITER ;

执行函数过程代码：

-- 查询价格大于 150 的课程数量

SELECT CountExpensiveCourses(150) AS '价格大于150的课程数量';

结果截图：



# 6、建立触发器

-- 在 course\_purchase 表中插入新记录时，触发器会将购买价格高于 200 的记录插入日志表。

-- 创建日志表

CREATE TABLE purchase\_log (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

member\_id INT,

course\_id INT,

purchase\_price DECIMAL(8, 2),

log\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- 创建触发器

DELIMITER $$

CREATE TRIGGER LogHighPricePurchase

AFTER INSERT ON course\_purchase

FOR EACH ROW

BEGIN

IF NEW.price > 200 THEN

INSERT INTO purchase\_log (member\_id, course\_id, purchase\_price)

VALUES (NEW.member\_id, NEW.course\_id, NEW.price);

END IF;

END $$

DELIMITER ;

-- 插入一条购买记录，触发器将自动记录到日志表

INSERT INTO course\_purchase (member\_id, course\_id, purchase\_date, price)

VALUES (1, 9, NOW(), 220.00);

-- 查看日志表

SELECT \* FROM purchase\_log;



# 7、项目设计总结

本项目涉及一个综合性健身房管理系统的数据库设计与实现，包括表结构定义、数据插入、查询操作、视图和存储过程、触发器等高级 SQL 功能的实现。项目全面涵盖了数据库开发的关键技术点。本次项目从需求分析、数据库设计到具体实现和优化，全面涵盖了数据库开发的核心环节。在实践中，我收获了扎实的 SQL 技能，并积累了处理复杂业务场景的经验。同时也发现了一些不足之处，为未来类似项目的改进指明了方向。