作业

```
# 统计出0001和0002分别有多少人
select subject_number, count(*) from grades group by subject_number;

# 统计 students 中age 大于18的人数
select count(*) from students where age>18;
select count(*) from (select * from students where age>18)as t;
```

default 默认约束

```
# default 常在创建表格的时候，用于修饰某个字段，让这个字段有默认值。类似于函数的默认参数。

# 建表
create table t1(id int default 110, name varchar(10));

# 举例： 向t1表格中的 name 字段插入两条数据
insert into t1(name) value("thenew"),("ameng");
insert into t1 value(1,'tongyao'),(55,'talent');

# 举例查询结果
+------+---------+
| id   | name    |
+------+---------+
|  110 | thenew  |
|  110 | ameng   |
|    1 | tongyao |
|   55 | talent  |
+------+---------+


# 举例： 虎牙直播
# 在用户注册的时候，如果我们不手动设置账户的昵称，它会有一个默认昵称"我是一颗小虎牙"
create table hu_ya_user(user_name varchar(20) default '我是一颗小虎牙',phone_number
int);
select * from hu_ya_user;

# 模拟用户在注册的时候，服务器后台做的事情
insert into hu_ya_user(phone_number) values(1008611);
insert into hu_ya_user(user_name,phone_number) values('张三',1008611);
```

not null 非空约束

```
# 非空约束在创建表格的时候用于修饰字段，代表这个字段必须填值，不能为空
create table t2(id int not null, name varchar(20));
# 插入一条id字段没有值的数据
insert into t2(name)  values("haha");
# Field 'id' doesn't have a default value   字段'id'没有默认值
```

unique key 唯一约束

```
# 用于在创建表格的时候，修饰字段限定唯一，即不能重复值
create table t3(id int unique key, name varchar(20) not null);

insert into t3 value(1,'xiaoming');
insert into t3 value(1,'xiaoming'); # 报错

# 王者荣耀
# 在王者荣耀中修改账户名的时候，会有一个限制：修改的账户名名字不能重复。
create table wzry_user(id int,name varchar(20) unique key);

SELECT * FROM wzry_user;

insert into wzry_user(id,name) values(1,'张三');
insert into wzry_user(id,name) values(2,'张三');
```

primary key 主键约束

```
# 主键：非空、唯一、确定 （一张表中只能有一个主键）
# 非空且唯一 就是主键约束
create table t4(id int primary key,
                name varchar(20) not null unique key);

insert into t4 values(1,'yy');
```

auto_increment 自增长约束

```
# 和主键约束用，多用于主键自增长
create table t5(id int primary key auto_increment, name varchar(10));
insert into t5(name)  values('xiaowang'),('laowang');

create table tb5(id int primary key auto_increment, name
varchar(10))auto_increment=100;    # auto_increment=100 指定从100开始自增长
```

foreign key 外键约束

```
# 什么是外键？
# 概念：外键是关系型数据库中用于建立表与表之间关联关系的一种约束。它定义了一个字段，用于引用另一个表（被引用表）的主键或唯一键。外键的作用是维护表与表之间的一致性和完整性。
```

具体来说，外键有以下特点和作用：
1、关联关系建立：通过外键，可以在一个表中创建一个字段，该字段的值与另一个表中的主键或唯一键相对应。这样可以建立起表与表之间的关联关系，实现数据的引用和连接。
2、数据完整性：外键约束可以确保引用的数据在被引用表中存在。当试图插入或更新外键字段的值时，数据库会验证该值是否在被引用表中存在。这样可以避免插入无效的引用值，保持数据的完整性。
3、数据一致性：外键约束还可以保持表与表之间的数据一致性。当主表中的主键或唯一键发生变化时，引用了该键的外键字段会自动更新，保持关联数据的一致性。

外键约束语法:

```
# 一个表格只能有一个主键，但是可以有多个外键
# 语法：
foreign key(字段) references 表格名(字段)
#声明这个字段是一个外键字段，这个字段中的数据是跟另一个表格中的 主键字段数据进行绑定的。
```

外键约束案例:

```
create table a(a_id int primary key auto_increment,
               a_name varchar(20) not null
              );

insert into a values(1,'a1'),(2,'a2');

select * from a;

create table b(b_id int primary key,
               b_name varchar(20) not null,
               fy_id int not null,
               foreign key(fy_id) references a(a_id)
              );

# foreign key(fy_id) references a(a_id)
# 指定fy_id为外键，  这个外键对应a表的主键a_id

insert into b values(1,'haha',1),(2,'heihei',2);

select * from b;
```

表关系

one to one

```
案例1：  一个学校  只能  有一个校长
        一个校长  也只能担任一个学校
案例2：   用户与用户详情的关系
# 关系：一对一多用于多表拆分，将一张表格的基础字段放在一张表中，其他详情字段放在另一张表中，以提升操作效率。
```

one to many

```
 案例1： 一个学校可以有 多个老师  每个老师 只能在 一个学校里面任职
 案例2： 一个顾客可以有很多订单，但是一个订单只能对应一个顾客
# sql实现 ： 在多的一方建立外键，指向一的一方的主键
```

many to many

```
多对多 一定有一张 中间表 存的是多对多的关系

一个学生 可以上 N个课程
每个课程 可以有多个学员进行学习
中间表： 自增id,学员id外键，课程id外键
# sql实现： 建立第三张中间表，中间表至少包含两个外键，分别管理两方主键
```

表连接查询：

表连接查询是在关系型数据库中用于联合多个表以检索相关数据的一种查询方式。它可以通过将共同字段进行匹配，将多个表中的数据进行关联，从而获取更丰富和综合的结果集。常见的表连接查询有以下几种类型：

```
# 语法：
1. 内连接（INNER JOIN）：      #  INNER可以省略
   - 内连接返回两个或多个表中的匹配行。
#  - 语法：
   SELECT 列名 FROM 表1 INNER JOIN 表2 ON 表1.字段 = 表2.字段;

2. 左连接（LEFT JOIN）：
   - 左连接返回左表中的所有行，以及右表中与左表匹配的行。
#  - 语法：
   SELECT 列名 FROM 表1 LEFT JOIN 表2 ON 表1.字段 = 表2.字段;

3. 右连接（RIGHT JOIN）：
   - 右连接返回右表中的所有行，以及左表中与右表匹配的行。
#  - 语法：
   SELECT 列名 FROM 表1 RIGHT JOIN 表2 ON 表1.字段 = 表2.字段;
```

一对一关系表链接查询

```
#  创建一对一关系的两张表格
create table chinese(id int primary key,name varchar(10));
# 插入数据
insert into chinese(id,name) values(410410001,'张三');
insert into chinese(id,name) values(410410002,'李四');
insert into chinese(id,name) values(410410003,'王五');
insert into chinese(id,name) values(410410004,'宋六');
insert into chinese(id,name) values(410410005,'孙七');

select * from chinese;
```

```
# 创建一个中国人信息表
create table chinese_msg(msg_id int primary key auto_increment,msg varchar(255),
c_id int, foreign key(c_id) references chinese(id));
# 插入数据
insert into chinese_msg(msg,c_id)  values('湖南汉族喜欢吃臭豆腐',410410001);
insert into chinese_msg(msg,c_id)  values('新疆维吾尔族喜欢吃哈密瓜',410410002);
insert into chinese_msg(msg,c_id)  values('内蒙古汉族喜欢吃手把肉',410410003);
insert into chinese_msg(msg,c_id)  values('西藏藏族喜欢喝青稞酒',410410004);

select * from chinese_msg;


#  内连接查询 这两张表中，对应的数据信息
select * from chinese join chinese_msg on chinese_msg.c_id=chinese.id;
# 以左边这张表格为准--左连接
select * from chinese left join chinese_msg on  chinese_msg.c_id=chinese.id;
# 以右边这张表格为准--右链接
select * from chinese_msg right join chinese on  chinese.id=chinese_msg.c_id;
```

一对多关系表链接查询

```
-- 创建顾客表
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255)
);
# 插入数据
INSERT INTO Customers (CustomerID, CustomerName)
VALUES (1, 'Alice Johnson'),
       (2, 'Bob Smith'),
       (3, 'Carol Williams');


-- 创建订单表
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

# 插入数据
INSERT INTO Orders (OrderID, OrderDate, CustomerID)
VALUES (101, '2023-08-01', 1),
       (102, '2023-08-02', 2),
       (103, '2023-08-03', 1),
       (104, '2023-08-04', 3);


-- 内连接查询，返回顾客名字和订单日期
SELECT Customers.CustomerName, Orders.OrderDate
FROM Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

-- 左连接查询，返回顾客名字和订单日期
```

```sql
SELECT Customers.CustomerName, Orders.OrderDate
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

-- 右连接查询，返回所有订单的订单日期和对应顾客的名字（如果有的话）
SELECT Customers.CustomerName, Orders.OrderDate
FROM Customers
RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

 # 当两张表格中的数据都能对应产生关联的话，左右链接的查询结果是和内连接没区别的，如果说，两张表格中的数据无法产生联系，那么左连接以左边的表为准，右链接以右边的表格为准。
```

多对多关系表链接查询

```sql
-- 创建学生表
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    StudentName VARCHAR(255)
);
-- 插入学生数据
INSERT INTO Students (StudentID, StudentName)
VALUES (1, 'Alice Johnson'),
       (2, 'Bob Smith'),
       (3, 'Carol Williams');

-- 创建课程表
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(255)
);
-- 插入课程数据
INSERT INTO Courses (CourseID, CourseName)
VALUES (101, 'Math'),
       (102, 'History'),
       (103, 'Science');

-- 创建选课表
CREATE TABLE Enrollments (
    EnrollmentID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
-- 插入选课数据
INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID)
VALUES (1001, 1, 101),
       (1002, 1, 102),
       (1003, 2, 101),
       (1004, 3, 103);

-- 内连接查询，返回学生姓名和选修课程名
SELECT Students.StudentName, Courses.CourseName
FROM Students
INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
```

```sql
INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;


-- 左连接查询，返回所有学生的姓名和选修课程名（如果有的话）
SELECT Students.StudentName, Courses.CourseName
FROM Students
LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
LEFT JOIN Courses ON Enrollments.CourseID = Courses.CourseID;


-- 右连接查询，返回所有选修课程的名字和对应学生的姓名（如果有的话）
SELECT Students.StudentName, Courses.CourseName
FROM Students
RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
RIGHT JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```