操作数据库

```
/*
    查询所有数据库
    标准语法:
        SHOW DATABASES;
*/
-- 查询所有数据库
SHOW DATABASES;


/*
    查询某个数据库的创建语句
    标准语法:
        SHOW CREATE DATABASE 数据库名称;
*/
-- 查询mysql数据库的创建语句
SHOW CREATE DATABASE mysql;



/*
    创建数据库
    标准语法:
        CREATE DATABASE 数据库名称;
*/
-- 创建db1数据库
CREATE DATABASE db1;


/*
    创建数据库，判断、如果不存在则创建
    标准语法:
        CREATE DATABASE IF NOT EXISTS 数据库名称;
*/
-- 创建数据库db2(判断，如果不存在则创建)
CREATE DATABASE IF NOT EXISTS db2;


/*
    创建数据库、并指定字符集
    标准语法:
        CREATE DATABASE 数据库名称 CHARACTER SET 字符集名称;
*/
-- 创建数据库db3、并指定字符集utf8
CREATE DATABASE db3 CHARACTER SET utf8;

-- 查看db3数据库的字符集
SHOW CREATE DATABASE db3;


-- 练习：创建db4数据库、如果不存在则创建，指定字符集为gbk
```

```
CREATE DATABASE IF NOT EXISTS db4 CHARACTER SET gbk;


-- 查看db4数据库的字符集
SHOW CREATE DATABASE db4;


/*
    修改数据库的字符集
    标准语法:
        ALTER DATABASE 数据库名称 CHARACTER SET 字符集名称;
*/
-- 修改数据库db4的字符集为utf8
ALTER DATABASE db4 CHARACTER SET utf8;

-- 查看db4数据库的字符集
SHOW CREATE DATABASE db4;


/*
    删除数据库
    标准语法:
        DROP DATABASE 数据库名称;
*/
-- 删除db1数据库
DROP DATABASE db1;

/*
    删除数据库，判断、如果存在则删除
    标准语法:
        DROP DATABASE IF EXISTS 数据库名称;
*/
-- 删除数据库db2，如果存在
DROP DATABASE IF EXISTS db2;

/*
    使用数据库
    标准语法:
        USE 数据库名称;
*/
-- 使用db4数据库
USE db4;

/*
    查询当前使用的数据库
    标准语法:
        SELECT DATABASE();
*/
-- 查询当前正在使用的数据库
SELECT DATABASE();
```

操作数据表

```
-- 使用mysql数据库
```

```sql
USE mysql;

/*
    查询所有数据表
    标准语法:
        SHOW TABLES;
*/
-- 查询库中所有的表
SHOW TABLES;

/*
    查询表结构
    标准语法:
        DESC 表名;
*/
-- 查询user表结构
DESC USER;

/*
    查询数据表的字符集
    标准语法:
        SHOW TABLE STATUS FROM 数据库名称 LIKE '表名';
*/
-- 查看mysql数据库中user表字符集
SHOW TABLE STATUS FROM mysql LIKE 'user';

/*
    创建数据表
    标准语法:
        CREATE TABLE 表名(
            列名 数据类型 约束,
            列名 数据类型 约束,
            ...
            列名 数据类型 约束
        );
*/
-- 创建一个product商品表(商品编号、商品名称、商品价格、商品库存、上架时间)
CREATE TABLE product(
    id INT,
    NAME VARCHAR(20),
    price DOUBLE,
    stock INT,
    insert_time DATE
);

-- 查看product表详细结构
DESC product;



/*
    修改表名
    标准语法:
        ALTER TABLE 旧表名 RENAME TO 新表名;
*/
```

```sql
-- 修改product表名为product2
ALTER TABLE product RENAME TO product2;



/*
    修改表的字符集
    标准语法:
        ALTER TABLE 表名 CHARACTER SET 字符集名称;
*/
-- 查看db3数据库中product2数据表字符集
SHOW TABLE STATUS FROM db3 LIKE 'product2';

-- 修改product2数据表字符集为gbk
ALTER TABLE product2 CHARACTER SET gbk;


/*
    给表添加列
    标准语法:
        ALTER TABLE 表名 ADD 列名 数据类型;
*/
-- 给product2表添加一列color
ALTER TABLE product2 ADD color VARCHAR(10);


/*
    修改表中列的数据类型
    标准语法:
        ALTER TABLE 表名 MODIFY 列名 数据类型;
*/
-- 将color数据类型修改为int
ALTER TABLE product2 MODIFY color INT;

-- 查看product2表详细信息
DESC product2;

/*
    修改表中列的名称和数据类型
    标准语法:
        ALTER TABLE 表名 CHANGE 旧列名 新列名 数据类型;
*/
-- 将color修改为address
ALTER TABLE product2 CHANGE color address VARCHAR(200);

-- 查看product2表详细信息


/*
    删除表中的列
    标准语法:
        ALTER TABLE 表名 DROP 列名;
*/
-- 删除address列
ALTER TABLE product2 DROP address;
```

```
/*
    删除表
    标准语法：
        DROP TABLE 表名;
*/
-- 删除product2表
DROP TABLE product2;

/*
    删除表，判断、如果存在则删除
    标准语法：
        DROP TABLE IF EXISTS 表名;
*/
-- 删除product2表，如果存在则删除
DROP TABLE IF EXISTS product2;
```

新增表数据

```
/*
    给指定列添加数据
    标准语法：
        INSERT INTO 表名(列名1,列名2,...) VALUES (值1,值2,...);
*/
-- 向product表添加一条数据
INSERT INTO product (id,NAME,price,stock,insert_time) VALUES (1,'手
机',1999.99,25,'2020-02-02');


-- 向product表添加指定列数据
INSERT INTO product (id,NAME,price) VALUES (2,'电脑',3999.99);

/*
    给全部列添加数据
    标准语法：
        INSERT INTO 表名 VALUES (值1,值2,值3,...);
*/
-- 默认给全部列添加数据
INSERT INTO product VALUES (3,'冰箱',1500,35,'2030-03-03');

/*
    批量添加所有列数据
    标准语法：
        INSERT INTO 表名 VALUES (值1,值2,值3,...),(值1,值2,值3,...),(值1,值2,值
3,...);
*/
-- 批量添加数据
INSERT INTO product VALUES (4,'洗衣机',800,15,'2030-05-05'),(5,'微波
炉',300,45,'2030-06-06');
```

修改和删除表数据

```
/*
    修改表数据
    标准语法：
        UPDATE 表名 SET 列名1 = 值1,列名2 = 值2,... [where 条件];
*/
-- 修改手机的价格为3500
UPDATE product SET price=3500 WHERE NAME='手机';

-- 修改电脑的价格为1800、库存为36
UPDATE product SET price=1800,stock=36 WHERE NAME='电脑';


/*
    删除表数据
    标准语法：
        DELETE FROM 表名 [WHERE 条件];
*/
-- 删除product表中的微波炉信息
DELETE FROM product WHERE NAME='微波炉';

-- 删除product表中库存为10的商品信息
DELETE FROM product WHERE stock=10;
```

查询_数据准备

```
-- 创建db1数据库
CREATE DATABASE db1;

-- 使用db1数据库
USE db1;

-- 创建数据表
CREATE TABLE product(
    id INT,          -- 商品编号
    NAME VARCHAR(20),    -- 商品名称
    price DOUBLE,        -- 商品价格
    brand VARCHAR(10),   -- 商品品牌
    stock INT,       -- 商品库存
    insert_time DATE         -- 添加时间
);

-- 添加数据
INSERT INTO product VALUES
(1,'华为手机',3999,'华为',23,'2088-03-10'),
(2,'小米手机',2999,'小米',30,'2088-05-15'),
(3,'苹果手机',5999,'苹果',18,'2088-08-20'),
(4,'华为电脑',6999,'华为',14,'2088-06-16'),
(5,'小米电脑',4999,'小米',26,'2088-07-08'),
(6,'苹果电脑',8999,'苹果',15,'2088-10-25'),
(7,'联想电脑',7999,'联想',NULL,'2088-11-11');
```

查询_查询全部

```
/*
    查询全部数据
    标准语法：
        SELECT * FROM 表名;
*/
-- 查询product表所有数据
SELECT * FROM product;


/*
    查询指定列
    标准语法：
        SELECT 列名1,列名2,... FROM 表名;
*/
-- 查询名称、价格、品牌
SELECT NAME,price,brand FROM product;


/*
    去除重复查询
    标准语法：
        SELECT DISTINCT 列名1,列名2,... FROM 表名;
*/
-- 查询品牌
SELECT brand FROM product;
-- 查询品牌，去除重复
SELECT DISTINCT brand FROM product;



/*
    计算列的值
    标准语法：
        SELECT 列名1 运算符(+ - * /) 列名2 FROM 表名;

    如果某一列为null，可以进行替换
    ifnull(表达式1,表达式2)
    表达式1：想替换的列
    表达式2：想替换的值
*/
-- 查询商品名称和库存，库存数量在原有基础上加10
SELECT NAME,stock+10 FROM product;

-- 查询商品名称和库存，库存数量在原有基础上加10。进行null值判断
SELECT NAME,IFNULL(stock,0)+10 FROM product;



/*
    起别名
    标准语法：
        SELECT 列名1,列名2,... AS 别名 FROM 表名;
```

```
*/
-- 查询商品名称和库存，库存数量在原有基础上加10。进行null值判断。起别名为getSum
SELECT NAME,IFNULL(stock,0)+10 AS getSum FROM product;
SELECT NAME,IFNULL(stock,0)+10 getSum FROM product;
```

查询_条件查询

```
/*
    条件查询
    标准语法：
        SELECT 列名列表 FROM 表名 WHERE 条件;
*/
-- 查询库存大于20的商品信息
SELECT * FROM product WHERE stock > 20;


-- 查询品牌为华为的商品信息
SELECT * FROM product WHERE brand='华为';

-- 查询金额在4000 ~ 6000之间的商品信息
SELECT * FROM product WHERE price >= 4000 AND price <= 6000;
SELECT * FROM product WHERE price BETWEEN 4000 AND 6000;


-- 查询库存为14、30、23的商品信息
SELECT * FROM product WHERE stock=14 OR stock=30 OR stock=23;
SELECT * FROM product WHERE stock IN(14,30,23);

-- 查询库存为null的商品信息
SELECT * FROM product WHERE stock IS NULL;

-- 查询库存不为null的商品信息
SELECT * FROM product WHERE stock IS NOT NULL;


-- 查询名称以小米为开头的商品信息
SELECT * FROM product WHERE NAME LIKE '小米%';

-- 查询名称第二个字是为的商品信息
SELECT * FROM product WHERE NAME LIKE '_为%';

-- 查询名称为四个字符的商品信息
SELECT * FROM product WHERE NAME LIKE '____';

-- 查询名称中包含电脑的商品信息
SELECT * FROM product WHERE NAME LIKE '%电脑%';
```

查询_聚合函数

```
/*
    聚合函数
    标准语法：
        SELECT 函数名(列名) FROM 表名 [WHERE 条件];
```

```
*/
-- 计算product表中总记录条数
SELECT COUNT(*) FROM product;

-- 获取最高价格
SELECT MAX(price) FROM product;


-- 获取最低库存
SELECT MIN(stock) FROM product;


-- 获取总库存数量
SELECT SUM(stock) FROM product;


-- 获取品牌为苹果的总库存数量
SELECT SUM(stock) FROM product WHERE brand='苹果';

-- 获取品牌为小米的平均商品价格
SELECT AVG(price) FROM product WHERE brand='小米';
```

查询_排序查询

```
/*
    排序查询
    标准语法:
        SELECT 列名 FROM 表名 [WHERE 条件] ORDER BY 列名1 排序方式1,列名2 排序方式2;
*/
-- 按照库存升序排序
SELECT * FROM product ORDER BY stock ASC;

-- 查询名称中包含手机的商品信息。按照金额降序排序
SELECT * FROM product WHERE NAME LIKE '%手机%' ORDER BY price DESC;

-- 按照金额升序排序，如果金额相同，按照库存降序排列
SELECT * FROM product ORDER BY price ASC,stock DESC;
```

查询_分组查询

```
/*
    分组查询
    标准语法:
        SELECT 列名 FROM 表名 [WHERE 条件] GROUP BY 分组列名 [HAVING 分组后条件过滤]
[ORDER BY 排序列名 排序方式];
*/
-- 按照品牌分组，获取每组商品的总金额
SELECT brand,SUM(price) FROM product GROUP BY brand;

-- 对金额大于4000元的商品，按照品牌分组,获取每组商品的总金额
SELECT brand,SUM(price) FROM product WHERE price > 4000 GROUP BY brand;
```

```sql
-- 对金额大于4000元的商品，按照品牌分组，获取每组商品的总金额，只显示总金额大于7000元的
SELECT brand,SUM(price) getSum FROM product WHERE price > 4000 GROUP BY brand
HAVING getSum > 7000;

-- 对金额大于4000元的商品，按照品牌分组，获取每组商品的总金额，只显示总金额大于7000元的、并按照总金额的降序排列
SELECT brand,SUM(price) getSum FROM product
WHERE price > 4000
GROUP BY brand
HAVING getSum > 7000
ORDER BY getSum DESC;
```

查询_分页查询

```sql
/*
    分页查询
    标准语法：
        SELECT 列名 FROM 表名
        [WHERE 条件]
        [GROUP BY 分组列名]
        [HAVING 分组后条件过滤]
        [ORDER BY 排序列名 排序方式]
        LIMIT 当前页数,每页显示的条数;

    LIMIT 当前页数,每页显示的条数;
    公式：当前页数 = (当前页数-1) * 每页显示的条数
*/
-- 每页显示3条数据

-- 第1页  当前页数=(1-1) * 3
SELECT * FROM product LIMIT 0,3;

-- 第2页  当前页数=(2-1) * 3
SELECT * FROM product LIMIT 3,3;

-- 第3页  当前页数=(3-1) * 3
SELECT * FROM product LIMIT 6,3;
```

约束_主键约束

```sql
-- 创建学生表(编号、姓名、年龄)  编号设为主键
CREATE TABLE student(
    id INT PRIMARY KEY,
    NAME VARCHAR(30),
    age INT
);

-- 查询学生表的详细信息
DESC student;

-- 添加数据
INSERT INTO student VALUES (1,'张三',23);
INSERT INTO student VALUES (2,'李四',24);
```

```sql
-- 删除主键
ALTER TABLE student DROP PRIMARY KEY;


-- 建表后单独添加主键约束
ALTER TABLE student MODIFY id INT PRIMARY KEY;
```

约束_主键自增约束

```sql
-- 创建学生表(编号、姓名、年龄)  编号设为主键自增
CREATE TABLE student(
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(30),
    age INT
);

-- 查询学生表的详细信息
DESC student;

-- 添加数据
INSERT INTO student VALUES (NULL,'张三',23),(NULL,'李四',24);

-- 删除自增约束
ALTER TABLE student MODIFY id INT;
INSERT INTO student VALUES (NULL,'张三',23);

-- 建表后单独添加自增约束
ALTER TABLE student MODIFY id INT AUTO_INCREMENT;
```

约束_唯一约束

```sql
-- 创建学生表(编号、姓名、年龄)  编号设为主键自增，年龄设为唯一
CREATE TABLE student(
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(30),
    age INT UNIQUE
);

-- 查询学生表的详细信息
DESC student;

-- 添加数据
INSERT INTO student VALUES (NULL,'张三',23);
INSERT INTO student VALUES (NULL,'李四',23);

-- 删除唯一约束
ALTER TABLE student DROP INDEX age;

-- 建表后单独添加唯一约束
ALTER TABLE student MODIFY age INT UNIQUE;
```

约束_非空约束

```
-- 创建学生表(编号、姓名、年龄)  编号设为主键自增，姓名设为非空，年龄设为唯一
CREATE TABLE student(
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(30) NOT NULL,
    age INT UNIQUE
);

-- 查询学生表的详细信息
DESC student;

-- 添加数据
INSERT INTO student VALUES (NULL,'张三',23);

-- 删除非空约束
ALTER TABLE student MODIFY NAME VARCHAR(30);
INSERT INTO student VALUES (NULL,NULL,25);

-- 建表后单独添加非空约束
ALTER TABLE student MODIFY NAME VARCHAR(30) NOT NULL;
```

约束_外键约束

```
-- 创建db2数据库
CREATE DATABASE db2;

-- 使用db2数据库
USE db2;

/*
    外键约束
    标准语法:
        CONSTRAINT 外键名 FOREIGN KEY (本表外键列名) REFERENCES 主表名(主表主键列名)
*/
-- 建表时添加外键约束
-- 创建user用户表
CREATE TABLE USER(
    id INT PRIMARY KEY AUTO_INCREMENT,    -- id
    NAME VARCHAR(20) NOT NULL             -- 姓名
);
-- 添加用户数据
INSERT INTO USER VALUES (NULL,'张三'),(NULL,'李四');


-- 创建orderlist订单表
CREATE TABLE orderlist(
    id INT PRIMARY KEY AUTO_INCREMENT,    -- id
    number VARCHAR(20) NOT NULL,          -- 订单编号
    uid INT,                    -- 外键列
    CONSTRAINT ou_fk1 FOREIGN KEY (uid) REFERENCES USER(id)
);
-- 添加订单数据
```

```
INSERT INTO orderlist VALUES (NULL,'hm001',1),(NULL,'hm002',1),
(NULL,'hm003',2),(NULL,'hm004',2);


-- 添加一个订单，但是没有真实用户。添加失败
INSERT INTO orderlist VALUES (NULL,'hm005',3);

-- 删除李四用户。删除失败
DELETE FROM USER WHERE NAME='李四';




/*
    删除外键约束
    标准语法：
        ALTER TABLE 表名 DROP FOREIGN KEY 外键名;
*/
-- 删除外键约束
ALTER TABLE orderlist DROP FOREIGN KEY ou_fk1;



/*
    建表后单独添加外键约束
    标准语法：
        ALTER TABLE 表名 ADD CONSTRAINT 外键名 FOREIGN KEY (本表外键列名) REFERENCES
主表名(主键列名);
*/
-- 添加外键约束
ALTER TABLE orderlist ADD CONSTRAINT ou_fk1 FOREIGN KEY (uid) REFERENCES
USER(id);
```

外键级联操作

```
/*
    添加外键约束，同时添加级联更新   标准语法：
    ALTER TABLE 表名 ADD CONSTRAINT 外键名 FOREIGN KEY (本表外键列名) REFERENCES 主表
名(主键列名)
    ON UPDATE CASCADE;

    添加外键约束，同时添加级联删除   标准语法：
    ALTER TABLE 表名 ADD CONSTRAINT 外键名 FOREIGN KEY (本表外键列名) REFERENCES 主表
名(主键列名)
    ON DELETE CASCADE;

    添加外键约束，同时添加级联更新和级联删除   标准语法：
    ALTER TABLE 表名 ADD CONSTRAINT 外键名 FOREIGN KEY (本表外键列名) REFERENCES 主表
名(主键列名)
    ON UPDATE CASCADE ON DELETE CASCADE;
*/
-- 删除外键约束
ALTER TABLE orderlist DROP FOREIGN KEY ou_fk1;
```

```sql
-- 添加外键约束，同时添加级联更新和级联删除
ALTER TABLE orderlist ADD CONSTRAINT ou_fk1 FOREIGN KEY (uid) REFERENCES
USER(id)
ON UPDATE CASCADE ON DELETE CASCADE;


-- 将李四这个用户的id修改为3,订单表中的uid也自动修改
UPDATE USER SET id=3 WHERE id=2;

-- 将李四这个用户删除,订单表中的该用户所属的订单也自动删除
DELETE FROM USER WHERE id=3;
```

表关系_一对一

```sql
-- 创建db3数据库
CREATE DATABASE db3;

-- 使用db3数据库
USE db3;

-- 创建person表
CREATE TABLE person(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(20)                        -- 姓名
);
-- 添加数据
INSERT INTO person VALUES (NULL,'张三'),(NULL,'李四');

-- 创建card表
CREATE TABLE card(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    number VARCHAR(20) UNIQUE NOT NULL, -- 身份证号
    pid INT UNIQUE,                         -- 外键列
    CONSTRAINT cp_fk1 FOREIGN KEY (pid) REFERENCES person(id)
);
-- 添加数据
INSERT INTO card VALUES (NULL,'12345',1),(NULL,'56789',2);
```

表关系_一对多

```sql
-- 创建user表
CREATE TABLE USER(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(20)                        -- 姓名
);
-- 添加数据
INSERT INTO USER VALUES (NULL,'张三'),(NULL,'李四');

-- 创建orderlist表
CREATE TABLE orderlist(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    number VARCHAR(20),                     -- 订单编号
    uid INT,                -- 外键列
```

```
    CONSTRAINT ou_fk1 FOREIGN KEY (uid) REFERENCES USER(id)
);
-- 添加数据
INSERT INTO orderlist VALUES (NULL,'hm001',1),(NULL,'hm002',1),(NULL,'hm003',2),
(NULL,'hm004',2);



/*
    商品分类和商品
*/
-- 创建category表
CREATE TABLE category(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(10)                        -- 分类名称
);
-- 添加数据
INSERT INTO category VALUES (NULL,'手机数码'),(NULL,'电脑办公');

-- 创建product表
CREATE TABLE product(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(30),           -- 商品名称
    cid INT,                -- 外键列
    CONSTRAINT pc_fk1 FOREIGN KEY (cid) REFERENCES category(id)
);
-- 添加数据
INSERT INTO product VALUES (NULL,'华为P30',1),(NULL,'小米note3',1),
(NULL,'联想电脑',2),(NULL,'苹果电脑',2);
```

表关系_多对多

```
-- 创建student表
CREATE TABLE student(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(20)            -- 学生姓名
);
-- 添加数据
INSERT INTO student VALUES (NULL,'张三'),(NULL,'李四');

-- 创建course表
CREATE TABLE course(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    NAME VARCHAR(10)            -- 课程名称
);
-- 添加数据
INSERT INTO course VALUES (NULL,'语文'),(NULL,'数学');


-- 创建中间表
CREATE TABLE stu_course(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 主键id
    sid INT,  -- 用于和student表中的id进行外键关联
    cid INT,  -- 用于和course表中的id进行外键关联
```

```
    CONSTRAINT sc_fk1 FOREIGN KEY (sid) REFERENCES student(id), -- 添加外键约束
    CONSTRAINT sc_fk2 FOREIGN KEY (cid) REFERENCES course(id)   -- 添加外键约束
);
-- 添加数据
INSERT INTO stu_course VALUES (NULL,1,1),(NULL,1,2),(NULL,2,1),(NULL,2,2);
```

多表查询_数据准备

```
-- 创建db4数据库
CREATE DATABASE db4;
-- 使用db4数据库
USE db4;

-- 创建user表
CREATE TABLE USER(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 用户id
    NAME VARCHAR(20),           -- 用户姓名
    age INT                                 -- 用户年龄
);
-- 添加数据
INSERT INTO USER VALUES (1,'张三',23);
INSERT INTO USER VALUES (2,'李四',24);
INSERT INTO USER VALUES (3,'王五',25);
INSERT INTO USER VALUES (4,'赵六',26);


-- 订单表
CREATE TABLE orderlist(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 订单id
    number VARCHAR(30),         -- 订单编号
    uid INT,    -- 外键字段
    CONSTRAINT ou_fk1 FOREIGN KEY (uid) REFERENCES USER(id)
);
-- 添加数据
INSERT INTO orderlist VALUES (1,'hm001',1);
INSERT INTO orderlist VALUES (2,'hm002',1);
INSERT INTO orderlist VALUES (3,'hm003',2);
INSERT INTO orderlist VALUES (4,'hm004',2);
INSERT INTO orderlist VALUES (5,'hm005',3);
INSERT INTO orderlist VALUES (6,'hm006',3);
INSERT INTO orderlist VALUES (7,'hm007',NULL);


-- 商品分类表
CREATE TABLE category(
    id INT PRIMARY KEY AUTO_INCREMENT,  -- 商品分类id
    NAME VARCHAR(10)                    -- 商品分类名称
);
-- 添加数据
INSERT INTO category VALUES (1,'手机数码');
INSERT INTO category VALUES (2,'电脑办公');
INSERT INTO category VALUES (3,'烟酒茶糖');
INSERT INTO category VALUES (4,'鞋靴箱包');
```

```sql
-- 商品表
CREATE TABLE product(
    id INT PRIMARY KEY AUTO_INCREMENT,   -- 商品id
    NAME VARCHAR(30),                    -- 商品名称
    cid INT, -- 外键字段
    CONSTRAINT cp_fk1 FOREIGN KEY (cid) REFERENCES category(id)
);
-- 添加数据
INSERT INTO product VALUES (1,'华为手机',1);
INSERT INTO product VALUES (2,'小米手机',1);
INSERT INTO product VALUES (3,'联想电脑',2);
INSERT INTO product VALUES (4,'苹果电脑',2);
INSERT INTO product VALUES (5,'中华香烟',3);
INSERT INTO product VALUES (6,'玉溪香烟',3);
INSERT INTO product VALUES (7,'计生用品',NULL);


-- 中间表
CREATE TABLE us_pro(
    upid INT PRIMARY KEY AUTO_INCREMENT,  -- 中间表id
    uid INT, -- 外键字段。需要和用户表的主键产生关联
    pid INT, -- 外键字段。需要和商品表的主键产生关联
    CONSTRAINT up_fk1 FOREIGN KEY (uid) REFERENCES USER(id),
    CONSTRAINT up_fk2 FOREIGN KEY (pid) REFERENCES product(id)
);
-- 添加数据
INSERT INTO us_pro VALUES (NULL,1,1);
INSERT INTO us_pro VALUES (NULL,1,2);
INSERT INTO us_pro VALUES (NULL,1,3);
INSERT INTO us_pro VALUES (NULL,1,4);
INSERT INTO us_pro VALUES (NULL,1,5);
INSERT INTO us_pro VALUES (NULL,1,6);
INSERT INTO us_pro VALUES (NULL,1,7);
INSERT INTO us_pro VALUES (NULL,2,1);
INSERT INTO us_pro VALUES (NULL,2,2);
INSERT INTO us_pro VALUES (NULL,2,3);
INSERT INTO us_pro VALUES (NULL,2,4);
INSERT INTO us_pro VALUES (NULL,2,5);
INSERT INTO us_pro VALUES (NULL,2,6);
INSERT INTO us_pro VALUES (NULL,2,7);
INSERT INTO us_pro VALUES (NULL,3,1);
INSERT INTO us_pro VALUES (NULL,3,2);
INSERT INTO us_pro VALUES (NULL,3,3);
INSERT INTO us_pro VALUES (NULL,3,4);
INSERT INTO us_pro VALUES (NULL,3,5);
INSERT INTO us_pro VALUES (NULL,3,6);
INSERT INTO us_pro VALUES (NULL,3,7);
INSERT INTO us_pro VALUES (NULL,4,1);
INSERT INTO us_pro VALUES (NULL,4,2);
INSERT INTO us_pro VALUES (NULL,4,3);
INSERT INTO us_pro VALUES (NULL,4,4);
INSERT INTO us_pro VALUES (NULL,4,5);
INSERT INTO us_pro VALUES (NULL,4,6);
INSERT INTO us_pro VALUES (NULL,4,7);
```

多表查询_内连接查询

```
/*
    显示内连接
    标准语法:
        SELECT 列名 FROM 表名1 [INNER] JOIN 表名2 ON 关联条件;
*/
-- 查询用户信息和对应的订单信息
SELECT * FROM USER INNER JOIN orderlist ON orderlist.uid = user.id;


-- 查询用户信息和对应的订单信息，起别名
SELECT * FROM USER u INNER JOIN orderlist o ON o.uid=u.id;


-- 查询用户姓名，年龄。和订单编号
SELECT
    u.name,      -- 用户姓名
    u.age,       -- 用户年龄
    o.number     -- 订单编号
FROM
    USER u           -- 用户表
INNER JOIN
    orderlist o      -- 订单表
ON
    o.uid=u.id;



/*
    隐式内连接
    标准语法:
        SELECT 列名 FROM 表名1,表名2 WHERE 关联条件;
*/
-- 查询用户姓名，年龄。和订单编号
SELECT
    u.name,      -- 用户姓名
    u.age,       -- 用户年龄
    o.number     -- 订单编号
FROM
    USER u,      -- 用户表
    orderlist o      -- 订单表
WHERE
    o.uid=u.id;
```

多表查询_外连接查询

```
/*
    左外连接
    标准语法:
        SELECT 列名 FROM 表名1 LEFT [OUTER] JOIN 表名2 ON 条件;
*/
```

```sql
-- 查询所有用户信息，以及用户对应的订单信息
SELECT
    u.*,
    o.number
FROM
    USER u
LEFT OUTER JOIN
    orderlist o
ON
    o.uid=u.id;



/*
    右外连接
    标准语法：
        SELECT 列名 FROM 表名1 RIGHT [OUTER] JOIN 表名2 ON 条件;
*/
-- 查询所有订单信息，以及订单所属的用户信息
SELECT
    o.*,
    u.name
FROM
    USER u
RIGHT OUTER JOIN
    orderlist o
ON
    o.uid=u.id;
```

多表查询_子查询

```sql
/*
    结果是单行单列的
    标准语法：
        SELECT 列名 FROM 表名 WHERE 列名=(SELECT 列名 FROM 表名 [WHERE 条件]);
*/
-- 查询年龄最高的用户姓名
SELECT MAX(age) FROM USER;
SELECT NAME,age FROM USER WHERE age=(SELECT MAX(age) FROM USER);


/*
    结果是多行单列的
    标准语法：
        SELECT 列名 FROM 表名 WHERE 列名 [NOT] IN (SELECT 列名 FROM 表名 [WHERE 条
件]);
*/
-- 查询张三和李四的订单信息
SELECT * FROM orderlist WHERE uid IN (1,2);
SELECT id FROM USER WHERE NAME IN ('张三','李四');
SELECT * FROM orderlist WHERE uid IN (SELECT id FROM USER WHERE NAME IN ('张
三','李四'));
```

```
/*
    结果是多行多列的
    标准语法：
        SELECT 列名 FROM 表名 [别名],(SELECT 列名 FROM 表名 [WHERE 条件]) [别名]
[WHERE 条件];
*/
-- 查询订单表中id大于4的订单信息和所属用户信息
SELECT * FROM orderlist WHERE id > 4;
SELECT
    u.name,
    o.number
FROM
    USER u,
    (SELECT * FROM orderlist WHERE id > 4) o
WHERE
    o.uid=u.id;
```

多表查询_多表查询练习

```
-- 1.查询用户的编号、姓名、年龄。订单编号
/*
分析
    用户的编号、姓名、年龄  user表       订单编号 orderlist表
    条件：user.id=orderlist.uid
*/
SELECT
    u.id,
    u.name,
    u.age,
    o.number
FROM
    USER u,
    orderlist o
WHERE
    u.id=o.uid;



-- 2.查询所有的用户。用户的编号、姓名、年龄。订单编号
/*
分析
    用户的编号、姓名、年龄  user表     订单编号 orderlist表
    条件：user.id=orderlist.uid
    查询所有的用户，左外连接
*/
SELECT
    u.id,
    u.name,
    u.age,
    o.number
FROM
    USER u
LEFT OUTER JOIN
    orderlist o
```

```
ON
    u.id=o.uid;



-- 3.查询所有的订单。用户的编号、姓名、年龄。订单编号
/*
分析
    用户的编号、姓名、年龄 user表     订单编号 orderlist表
    条件：user.id=orderlist.uid
    查询所有的订单，右外连接
*/
SELECT
    u.id,
    u.name,
    u.age,
    o.number
FROM
    USER u
RIGHT OUTER JOIN
    orderlist o
ON
    u.id=o.uid;



-- 4.查询用户年龄大于23岁的信息。显示用户的编号、姓名、年龄。订单编号
/*
分析
    用户的编号、姓名、年龄 user表     订单编号 orderlist表
    条件：user.id=orderlist.uid AND user.age > 23
*/
SELECT
    u.id,
    u.name,
    u.age,
    o.number
FROM
    USER u,
    orderlist o
WHERE
    u.id=o.uid
    AND
    u.age > 23;


-- 5.查询张三和李四用户的信息。显示用户的编号、姓名、年龄。订单编号
/*
分析
    用户的编号、姓名、年龄 user表    订单编号 orderlist表
    条件：user.id=orderlist.uid AND user.name IN ('张三','李四')
*/
SELECT
    u.id,
    u.name,
```

```
    u.age,
    o.number
FROM
    USER u,
    orderlist o
WHERE
    u.id=o.uid
    AND
    u.name IN ('张三','李四');


-- 6.查询商品分类的编号、分类名称。分类下的商品名称
/*
分析
    商品分类的编号、分类名称 category表    商品名称 product表
    条件：category.id=product.cid
*/
SELECT
    c.id,
    c.name,
    p.name
FROM
    category c,
    product p
WHERE
    c.id=p.cid;


-- 7.查询所有的商品分类。商品分类的编号、分类名称。分类下的商品名称
/*
分析
    商品分类的编号、分类名称 category表    商品名称 product表
    条件：category.id=product.cid
    查询所有的商品分类，左外连接
*/
SELECT
    c.id,
    c.name,
    p.name
FROM
    category c
LEFT OUTER JOIN
    product p
ON
    c.id=p.cid;


-- 8.查询所有的商品信息。商品分类的编号、分类名称。分类下的商品名称
/*
分析
    商品分类的编号、分类名称  category表    商品名称 product表
    条件：category.id=product.cid
    查询所有的商品信息，右外连接
*/
SELECT
```

```
    c.id,
    c.name,
    p.name
FROM
    category c
RIGHT OUTER JOIN
    product p
ON
    c.id=p.cid;



-- 9.查询所有的用户和该用户能查看的所有的商品。显示用户的编号、姓名、年龄。商品名称
/*
分析
    用户的编号、姓名、年龄 user表    商品名称 product表     中间表 us_pro
    条件：us_pro.uid=user.id AND us_pro.pid=product.id
*/
SELECT
    u.id,
    u.name,
    u.age,
    p.name
FROM
    USER u,
    product p,
    us_pro up
WHERE
    up.uid=u.id
    AND
    up.pid=p.id;



-- 10.查询张三和李四这两个用户可以看到的商品。显示用户的编号、姓名、年龄。商品名称
/*
分析
    用户的编号、姓名、年龄 user表    商品名称 product表    中间表 us_pro
    条件：us_pro.uid=user.id AND us_pro.pid=product.id AND user.name IN ('张三','李
四')
*/
SELECT
    u.id,
    u.name,
    u.age,
    p.name
FROM
    USER u,
    product p,
    us_pro up
WHERE
    up.uid=u.id
    AND
    up.pid=p.id
    AND
```

```
    u.name IN ('张三','李四');
```