# day03_基础查询

## 1.准备工作

```
# 操作库的前提: 先启动mysql服务并连接它
# 操作表的前提: 先有库并使用它
create database day03;
use day03;
# 创建商品表
CREATE TABLE product
(
    pid         INT PRIMARY KEY,
    pname       VARCHAR(20),
    price       DOUBLE,
    category_id VARCHAR(32)
);
# 插入数据
INSERT INTO product(pid,pname,price,category_id) VALUES(1,'联想',5000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(2,'海尔',3000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(3,'雷神',5000,'c001');
INSERT INTO product(pid,pname,price,category_id) VALUES(4,'杰克琼斯',800,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(5,'真维斯',200,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(6,'花花公子',440,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(7,'劲霸',2000,'c002');
INSERT INTO product(pid,pname,price,category_id) VALUES(8,'香奈儿',800,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(9,'相宜本草',200,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(10,'面霸',5,'c003');
INSERT INTO product(pid,pname,price,category_id) VALUES(11,'好想你枣',56,'c004');
INSERT INTO product(pid,pname,price,category_id) VALUES(12,'香飘飘奶茶',1,'c005');
INSERT INTO product(pid,pname,price,category_id) VALUES(13,'海澜之家',1,'c002');
```

## 2.简单查询

### 知识点:

```
简单查询关键字:  select 和 from

简单查询格式:  select [distinct]字段名...  from 表名;       select * from 表名;
    [] : 代表括号中内容可以省略
    ...: 省略号,代表可以指定多个字段
    *  : 代表查询所有字段

    distinct关键字: 给指定字段去重,此关键字必须放到select之后
    as关键字: 可以给表或者字段起别名,可以省略
```

## 示例:

```
# 2.简单查询练习
# 注意: 以下查询操作统一都使用day03库下的表
use day03;
# 格式: select [distinct]字段名|* from 表名;
# 需求1: 查询product表所有数据
select * from product;
# 注意: 如果想要查询其他库下的某个表,可以使用 库名.表名 锁定对应的表
select * from day02.product; # 注意: 如果day02下没有product表就会报错
select * from day03.product;

# 需求2: 只查询所有商品的名称和价格
select pname,price from product;

# 需求3: 只查询所有商品的名称和价格,要求pname起别名为商品名称,price起别名为商品价格
# 注意: 别名不建议用中文,以下仅仅为了演示操作,但是以后工作中别名用英文或者拼音
select pname as 商品名称,price as 商品价格 from product;
# 注意: as关键字可以省略
select pname  商品名称,price  商品价格 from product;

# 需求4: 查询商品的分类,要求去重
select distinct category_id from product;
```

# 3.条件查询

## 知识点:

```
条件查询关键字: where

条件查询基础格式: select 字段名 from 表名 where 条件;

        比较运算符:  >:大于  <:小于  >=:大于等于  <=:小于等于    !=和<> : 不等于

        逻辑运算符:  and:并且  or:或者  not:取反

        范围 查询:  between x and y: x到y的连续范围        in(x,y): x或者y

        模糊 查询:  like:模糊查询关键字   %:0个或者多个字符   _: 1个字符

        空 判 断 :  is null: 判断为空    is not null: 判断不为空
```

## 示例:

```
# 3.条件查询练习
# 3.1 比较运算符练习
# 比较运算符:  >:大于  <:小于  >=:大于等于  <=:小于等于    !=和<> : 不等于
# 需求1: 查询商品价格大于2000的商品信息
select * from product where price > 2000;
# 需求2: 查询商品价格小于2000的商品信息
select * from product where price < 2000;
# 需求3: 查询商品价格大于等于2000的商品信息
select * from product where price >= 2000;
# 需求4: 查询商品价格小于等于2000的商品信息
```

```
select * from product where price <= 2000;
# 需求5: 查询商品价格不等于2000的商品信息
select * from product where price != 2000;
select * from product where price <> 2000;

# 3.2逻辑运算符练习
# 逻辑运算符:  and:并且  or:或者  not:取反
# 需求6: 查询商品价格大于等于200并且小于等于2000的商品信息
select * from product where price >= 200 and price <=2000;
# 需求7: 查询商品价格等于3000 或者 等于5000的商品信息
select * from product where price = 3000 or price = 5000;
# 需求8: 查询商品价格不等于2000的商品信息
select * from product where not(price = 2000); # 此种方式相比!=是麻烦的,仅用于演示取反效果
# 需求9: 查询商品价格不在200到2000之间的商品信息
select * from product where not (price >= 200 and price <=2000);

# 3.3范围查询练习
# 范围查询: between x and y: x到y的连续范围        in(x,y): x或者y
# 需求6: 查询商品价格大于等于200并且小于等于2000的商品信息
select * from product where price between 200 and 2000;
# 需求7: 查询商品价格等于3000 或者 等于5000的商品信息
select * from product where price in(3000,5000);
# 需求8: 查询商品价格不等于2000的商品信息
select * from product where price not in(2000);
# 需求9: 查询商品价格不在200到2000之间的商品信息
select * from product where price not between 200 and 2000;


# 3.4 模糊查询练习
# 模糊 查询:  like:模糊查询关键字   %:0个或者多个字符   _: 1个字符
# 需求10: 查询商品名称以'香'开头的所有商品信息
select * from product where pname like '香%';
# 需求11: 查询商品名称以'香'开头并且名称是3个字的商品信息
select * from product where pname like '香__';
# 需求12: 查询包含'想'字的所有商品信息
select * from product where pname like '%想%';
# 需求13: 查询商品名称以'斯'结尾的所有商品信息
select * from product where pname like '%斯';
# 需求14: 查询商品名称以'斯'结尾并且名称是3个字的所有商品信息
select * from product where pname like '__斯';


# 3.5 空判断练习
# 空 判 断 :  is null: 判断为空    is not null: 判断不为空
# null: mysql中就代表空值,没有任何意义 ,它并不等于空字符串'',也不等于字符串'null'
# 因为原始数据没有null值,为了方便演示插入一条
INSERT INTO product(pid,pname,price,category_id) VALUES(14,'小米',1999,null);
INSERT INTO product(pid,pname,price,category_id) VALUES(15,'红米',1999,'null');
INSERT INTO product(pid,pname,price,category_id) VALUES(16,'华为',1999,'');
# 需求1: 查询没有分类的商品
select * from product where category_id is null;
# 需求2: 查询有分类的商品
# 注意: 不要写成not is null,这种写法是错误的
select * from product where category_id is not null;
```

# 4.排序查询

## 知识点:

```
排序查询关键字: order by

排序查询基础格式: select 字段名 from 表名 order by 排序字段名 asc|desc , 排序字段名
asc|desc;
        asc: 升序  注意: 是默认升序,order by后不写排序规则默认就是升序   123456
        desc: 降序   654321
        注意: 如果order by 后面跟了多个字段,先按照前面的字段排序,如果前面的字段值相同再按照后
面的字段排序
```

## 示例:

```
# 4.排序查询
# 排序查询格式: select 字段名 from 表名 order by 排序字段名 asc|desc;
# 需求1: 查询所有商品信息,并且按照价格升序排序
select * from product order by price ; -- 此处默认升序
select * from product order by price asc; -- 注意: 以后如果是升序就不要写asc,因为已经默
认了

# 需求2: 查询所有商品信息,并且按照价格降序排序
select * from product order by price desc;

# 需求3: 查询所有商品信息,并且按照价格降序排序,如果价格相同再按照分类编号升序排序
select * from product order by price desc , category_id;

# 需求4: 查询所有商品信息,并且按照价格降序排序,如果价格相同再按照分类编号降序排序
select * from product order by price desc , category_id desc;
```

# 5.聚合查询

## 知识点:

```
聚合查询函数:  count()数量  sum()求和   avg()平均值  max()最大值  min()最小值

聚合查询基础格式: select 聚合函数 from 表名;

注意1: 默认一个表就是一个大的分组

注意2: 聚合函数又叫统计函数,也叫分组函数

注意3: 聚合函数(字段名)方式会自动忽略null值 , 而count(*)方式不会忽略null值
```

## 示例:

```
# 5.聚合查询
# 聚合查询基础格式: select 聚合函数 from 表名;
# 需求1: 查询所有商品的个数
select count(pid) from product; -- 结果16
# 注意: 以后只要查个数建议首选count(*),因为它不会忽略null值
select count(*) from product; -- -- 结果16

# 需求2: 查询有分类的商品个数
```

```
select count(*) from product where category_id is not null; -- 结果15
# 小技巧: 利用count(字段名)自动忽略null值的特点
select count(category_id) from product;-- 结果15

# 需求3: 查询商品价格总和
select sum(price) from product;

# 需求4: 查询商品价格平均值
select avg(price) from product;

# 需求5: 查询商品最大价格
select max(price) from product;

# 需求6: 查询商品最小价格
select min(price) from product;
```

# 6.分组查询

## 知识点:

```
分组查询关键字:  group by

分组查询基础格式: select 分组字段名,聚合函数(字段名)  from 表名 group by 分组字段名;

分组查询基础+条件格式:  select 分组字段名,聚合函数(字段名) from 表名 [where 非聚合条件]
group by 分组字段名 [having 聚合条件];
```

## 示例1:

```
# 6.分组查询
# 分组查询基础格式: select 聚合函数 from 表名 group by 分组字段名;
# 需求1: 查询每个商品分类内商品的个数
# 方式1: 笨方法
select count(*) from product where category_id = 'c001'; -- c001 3
select count(*) from product where category_id = 'c002'; -- c002 5
select count(*) from product where category_id = 'c003'; -- c003 3
select count(*) from product where category_id = 'c004'; -- c004 1
select count(*) from product where category_id = 'c005'; -- c005 1
select count(*) from product where category_id is null; -- <null> 1
select count(*) from product where category_id = 'null'; -- 'null' 1
select count(*) from product where category_id = '';     -- '' 1
# 方式2: 用group by先分组,再组内聚合
select category_id,count(*) from product group by category_id;

# 需求2: 查询每个商品分类内商品价格总和
select category_id,sum(price) from product group by category_id;

# 需求3: 查询每个商品分类内商品的最大价格
select category_id,max(price) from product group by category_id;

# 需求4: 查询每个商品分类内商品的最小价格
select category_id,min(price) from product group by category_id;

# 需求5: 查询每个商品分类内商品平均价格
select category_id,avg(price) from product group by category_id;
```

```
# 需求6: 查询每个商品分类内商品平均价格,要求小数点后保留2位
# 拓展round(数据,保留位数)
select round(3.1415926,2);
# round实际应用
select category_id,round(avg(price),2) from product group by category_id;
```

### 示例2:

```
# 分组查询基础+条件格式:  select 聚合函数 from 表名 [where 非聚合条件] group by 分组字段名 [having 聚合条件];
# 需求1: 查询有商品分类的所有商品
select * from product where category_id is not null;

# 需求2: 查询每个商品分类的个数,但是分类为null的除外
# 分析: 先查询有商品分类的商品,然后按照商品分类编号进行分组统计每个分类的个数
select category_id,count(*) from product
where category_id is not null
group by category_id;

# 需求3: 查询每个商品分类的个数,但是分类为null的除外,并且只显示分组商品商品个数大于1的分组信息
select category_id,count(*) from product
where category_id is not null
group by category_id
having count(*) > 1;

# 注意: where只能筛选非聚合条件,having可以筛选聚合条件,也可以筛选非聚合条件,
# 但是不建议用having去筛选非聚合条件因为效率会降低,以后所有非聚合条件使用where筛选
select category_id,count(*) from product
group by category_id
having category_id is not null and count(*) > 1; -- 此种方式不建议,仅用于演示
```

## 7.分页查询

### 知识点:

```
分页查询关键字:  limit

分页查询基础格式: select 字段名 from 表名 limit x,y;
        x: 整数,代表查询的起始索引 注意: 索引默认是从0开始数
        y: 整数,代表查询的条数(每页展示的数量)
```

### 分页练习:

```
# 7.分页查询
# 分页查询基础格式: select 字段名 from 表名 limit x,y;
# 需求1: 查询表中前五条数据
select * from product limit 0,5;
# 注意: 如果起始索引是0,那么可以省略不写
select * from product limit 5;

# 需求: 数据共有16条, 每页显示4条,依次查询出每页展示的数据
# 查询第1页数据
select * from product limit 0,4;
```

```
# 查询第2页数据
select * from product limit 4,4;
# 查询第3页数据
select * from product limit 8,4;
# 查询第4页数据
select * from product limit 12,4;
# 推出起始索引计算结论: (页数-1)*每页展示的条数
```

## topN练习:

```
# topN练习
# select 字段名 from 表名 order by 排序字段 asc|desc limit x,y;
# 需求1: 查询商品价格最低的三件商品
# 分析: 先按照价格升序排序,然后用limit取前3个
select * from product order by price limit 3;

# 需求2: 查询商品价格最高的3件商品
# 分析: 先按照价格降序排序,然后用limit取前3个
select * from product order by price desc limit 3;
```