6.1今日目标

6.2【记忆】limit限制记录

- limit 数据有很多，只取指定数量的数据
  - 表中的数据，位置默认从 0 开始
- limit 使用格式：

  ```
  limit start, count
  ```

- **limit 一定要写到 SQL 语句的最后面**

```
只返回固定个数：select * from 表名 limit 数量;
分页查询：select * from 表名 limit 开始索引 查询数量;
-- 1. start公式 count*(page-1)

-- 2. limit 放在最后面(注意)

-- 3. 坑点：当有排序规则，且做了分页查询．如果排序的数据有重复项，就可以导致返回的数据时随机的
    -- 解决方案：排序规则中，最后必须有一个唯一值
```

6.3【记忆】聚合函数

- 聚合函数：组函数，对一组数据进行运算和统计；
- 常见的聚合函数：
  - `count(col)`
  - `max(col)`
  - `min(col)`
  - `sum(col)`
  - `avg(col)`

```
-- 聚合函数默认忽略字段为null的记录
-- 计算平均年龄 sum(age)/count(*)
select avg(age) from students;


-- 四舍五入 round(数据 ，要保留的位数)
-- 计算所有人的平均年龄，保留2位小数
select round(avg(age), 2) from students;

-- 计算男性的平均身高 保留2位小数
-- ifnull函数：表示判断指定字段的值是否为null，如果为空使用自己提供的值。
select round(avg(ifnull(height, 0)), 2) from students where gender = '男';
```

6.4【记忆】group分组

- 用于分组（按照某个特定的字段进行分类）

> `group by`

```
-- 1. group by的使用
group by可用于单个字段分组，也可用于多个字段分组

-- 根据gender字段来分组
select gender from students group by gender;

-- 根据name和gender字段进行分组
select name, gender from students group by name, gender;

-- 2. group by + group_concat()的使用
group_concat(字段名)：统计每个分组指定字段的信息集合，每个信息之间使用逗号进行分割

-- 根据gender字段进行分组， 查询gender字段和分组的name字段信息
select gender,group_concat(name) from students group by gender;

-- 3. group by + 聚合函数的使用
-- 统计不同性别的人的平均年龄
select gender,avg(age) from students group by gender;

-- 统计不同性别的人的个数
select gender,count(*) from students group by gender;

-- 4. group by + having的使用
having作用和where类似都是过滤数据的，但having是过滤分组数据的，只能用于group by

-- 根据gender字段进行分组，统计分组条数大于2的
select gender,count(*) from students group by gender having count(*)>2;

-- 5. group by + with rollup的使用
with rollup的作用是：在最后记录后面新增一行，显示select查询时聚合函数的统计和计算结果

-- 根据gender字段进行分组，汇总总人数
select gender,count(*) from students group by gender with rollup;

-- 根据gender字段进行分组，汇总所有人的年龄
select gender,group_concat(age) from students group by gender with rollup;
```

注意点： 分组的字段要出现在 select 的后面

6.5【记忆】内连接

- 内连接：查询的结果为两个表匹配到的数据，默认是笛卡尔积算法（所有组合）
  - 关键字

    > `inner join ... on...`

  - 格式

```
select c.name, s.* from students s inner join classes c on s.cls_id = c.id order by
c.id;
```

6.6【记忆】左连接

- 左外连接 : 以左表为主表查询右表的内容;

  > `left join ... on ...`

  - 左外连接一个表，在从表中没有找到匹配，右侧补 NULL

```
select * from students s left join classes c on s.cls_id = c.id;

-- 查询找不到对应班级的学生
select * from students s left join classes c on s.cls_id = c.id where c.id is null;
```

6.7 【记忆】右连接

- 右外连接 : 以右表为主查询左表内容;

  > `right join ... on ...`

  - 右外连接一个表，在从表中没有找到匹配，左侧补 NULL

```
格式:select 字段 from 表1 right join 表2 on 表1.字段1 = 表2.字段2

  -- 将数据表名字互换位置，用left join完成
    select * from students s right join classes c on s.cls_id = c.id;

    -- 查询找不到对应学生的班级
    select * from students s right join classes c on s.cls_id = c.id where s.id is null;
```

6.8【记忆】自连接

- 自连接： 特殊的内连接; 左表和右表是同一个表;

```
select c.id, c.title, c.pid, p.title from areas as c inner join areas as p on c.pid = p.id
where p.title = '山西省';
```

6.9 【记忆】子查询

- 在一个 select 语句中,嵌入了另外一个 select 语句, 那么被嵌入的 select 语句称之为子查询语句,外部那个select 语句则称为主查询.
- 主查询和子查询的关系:
  - 子查询嵌入主查询中
  - 充当主查询的条件或者数据源
  - 子查询也是一条完整的查询语句

```
-- 查出高于平均身高的信息
select * from students where height > (select avg(height) from students);
```

6.10【了解】数据库三范式

- 范式，设计数据库的一些原则：
- 数据库设计的3范式：
  - 1NF : 列的原子性(能拆就拆)
  - 2NF : 1> 必须有主键 2> 非主键必须完全依赖于主键(双主键的某些字段, 可能只依赖其中1个主键)
  - 3NF :非主键列必须直接依赖于主键

6.11【了解】E-R模型及表间关系

- E-R模式: 即实体-关系模型(先想清楚, 有哪些表, 表有哪些字段, 表和表之间有哪些关系)
- 表间关系：
  - 一对一：在表A或表B中创建一个字段，存储另一个表的主键值
  - 一对多：在多的一方表(学生表)中创建一个字段，存储班级表的主键值
  - 多对多：

    > 新建一张表，这个表只有两个字段，一个用于存储A的主键值，一个用于存储B的主键值

6.12【理解】外键使用

- 外键的概念：另一个张关联表的主键
- 作用：关联2张表的数据
- 创建外键：
  - 已经存在的表建立外键：

    ```
    alter table students add foreign key(cls_id) references classes(id);
    ```

  - 创建表的时候建立外键

    ```
    create table teacher(
        id int not null primary key auto_increment,
        name varchar(10),
        s_id int not null,
        foreign key(s_id) references 另一张表名(id)
    );
    ```

- 删除外键：
  - 查看外键名称： `show create table teacher;`
  - 删除外键:`alter table teacher drop foreign key teacher_ibfk_1;`

6.13【应用】SQL演练

- 数据准备
- SQL语句演练
  - 演练-分组和聚合函数的组合使用

```
    -- 1> 查询类型 cate_name 为 '超级本' 的商品名称 name 、价格 price ( where )
    select name, price from goods where cate_name = '超级本';

    -- 2> 显示商品的种类
    -- 分组的方式( group by )
    select cate_name from goods group by cate_name;


    -- 去重的方法( distinct )
    select distinct cate_name from goods;


    -- 3> 求所有电脑产品的平均价格 avg ,并且保留两位小数( round )
    select round(avg(price), 2) from goods;


    -- 4> 显示 每种类型 cate_name (由此可知需要分组)的 平均价格
    select cate_name, avg(price) from goods group by cate_name;


    -- 5> 查询 每种类型 的商品中 最贵 max 、最便宜 min 、平均价 avg 、数量 count
    select cate_name, max(price), min(price), avg(price), count(*) from goods
group by cate_name;

    -- 6> 查询所有价格大于 平均价格 的商品，并且按 价格降序 排序 order desc

    -- 查询平局价格(avg_price)
    select avg(price) from goods;

    -- 使用子查询
    select * from goods where price > (select avg(price) from goods) order by
price desc;
```

6.14 知识点总结