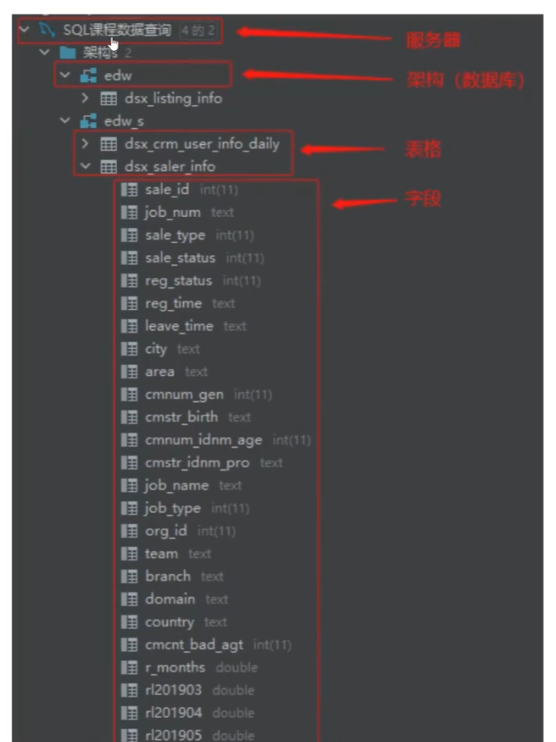
# 0628 十道题搞定SQL

1. SQL自学原则
   1. **不看这个视频SQL等于白学（必看** <https://www.bilibili.com/video/BV1zX4y1u7ZH)>
   2. **课程知识点环环相扣，不睡觉也要想办法弄懂，千万别掩饰！**
   3. **代码能力只能靠独立思考和勤学苦练，没有任何捷径，有依赖心理是学不好任何高级工具的！！！**
   4. **不会先想，想完再想，最后再问！**
   5. 要有自己的sql笔记本，记下自己总是遗忘或者错误的关键用法和函数，积累常用语句
   6. 一定要学会自己看报错Debug，这是你巩固提升，跟别人拉开代码效率和准确性的唯一机会（写得多，错的多，该的多，该犯的错都犯过，才能把代码写得又快又好）
      1. 不要害怕报错
      2. 先检查基础语法、大小写、空格、**书写错误**
      3. 先定位报错的位置linex
      4. 认真翻译报错信息
      5. 思考可能的报错原因（80%的报错都是因为语句用法不熟和粗心），排除粗心原因
      6. 百度寻找对应解决方法
      7. 加强积累，把你报错的位置和思路记录下来，巩固报错部分较为生疏的语法，很快你就能一眼知道报错原因，报错会越来越少
      8. 不会Debug，不是懒就是基本功不扎实
2. 熟悉数据库的服务器、架构、表格、字段
   1. 示例



* 1. 服务器：数据库中一切数据和信息所存储的地方，可以理解为你的电脑硬盘
     1. 服务器往往有对应的物理主机和访问地址，连接数据库时主机后面所填的就是访问地址
  2. 架构：服务器中具体的数据库，用于存储不同用途的表格，可以理解为电脑中的文件夹
     1. 常见的架构有ods/edw/edw\_s/ddm他们各自有不同的储存特点
        1. ods(Operational Data Store) : 操作型数据仓库
           1. 同步业务生产过程中所有数据，比edw更加详细，冗余和复杂度更高
        2. edw（Enterprise Data Warehouse）：企业数据库
           1. 已处理完成并进行汇总的，可供业务直接使用的企业数据库
        3. edw\_s：安全级别更高的企业数据库
           1. 存储安全级别更高的敏感或机密数据，例如用户的身份证、手机号、真实姓名等
           2. 加密性更高，一般需要独立授权
        4. ddm（Distributed Database Middleware）：局部数据库
           1. 专门正对某个具体的应用或需求建设的局部数据库，只关心自己需要的数据库，不会全盘考虑企业整体的数据架构和应用，每个应用都有自己的DM，所以DM可以基于仓库建设也可以独立建设。
  3. 表格：存在于不用架构中具体的数据表格，可以理解为文件家中的Excel
     1. 课程中共有3个表格
        1. [ddm.shop](http://ddm.shop/)
        2. ddm.cpc
        3. ods.orders
     2. 真实场景中“架构-表格”的形式存储表名更加方便跨数据库取数
     3. 对表格使用默认统一的别名，更方便后续的代码移植（用a b c的话还学要修改select后的字段前缀）
  4. 字段：表格中不同列，可以理解为Excel中的列
     1. 难道数据库权限后的第一件事就是整理自己的Dictionary，梳理常用的表格和字段含义
     2. 课程数据的简易字典（详细内容看课程资料网盘链接：<https://pan.baidu.com/s/1-jOwaXWArtv8h21B0-uWGg> 提取码：wsra）



1. SQL基础语法与运行原理：select-from-where-group by-having-order by-limit
   1. 场景：刚来公司的第一天，想看一下门店营业表有哪些字段【describe】
      1. 方法
         1. 用describe查看数据字典，看看表格有哪些字段，字段什么类型，有没有注释
      2. 语法
         1. 语法
            1. describe 数据库名称.表格名称
         2. 注释
            1. describe后表格名称唯一时，无需声明数据库名称
            2. describe后表格名称不唯一时，需声明数据库名称，并以.隔开
         3. 举例
            1. describe ddm.cpc
            2. describe edw\_sdsx\_saler\_info
      3. 代码
      4. 小练习
         1. 制作自己的数据字典 （乱码请改用ANSI）
         2. 查看所有数据库字段，并导出exce表格，重新命名子表，并将子表合并到同一excel文件中
   2. 场景：运营需要查看旗下所有品牌和门店12月1日至7日在美团上的GMV和下单人数（2019年）【select&from&where】
      1. 原理
         1. 【Excel演示】
            1. 在shop表中用筛选器选出时间在2019-12-01和2019-12-07之间且平台为meituan
            2. 显示指定列显示
      2. 语法
         1. from表格名
         2. where条件
            1. 基于from复制的表格，按照where后的条件对表格中的行进行筛选
            2. = 单个数值精确匹配
            3. between A and B 匹配包含AB及AB之间的所有值
            4. in(A,B,C,D) 可以实现多选，not in 可以反选
            5. and/or 可以并列多个条件

and 且

where 日期between ‘2019-12-01’ and ‘2019-12-07’ and 平台 = ‘meituan’

or 或

* + - * 1. like 配合通配进行模糊匹配

‘%’ 代表任意数量字符

‘-’ 只代表单个字符

‘[]’ 指定字符集，必须匹配该位置的一个字符

‘^’ 在[] 中使用，表示否定

* + - 1. select字段名
         1. SQL

基于from、join、where、group by、having处理好的数据表格，新增计算字段

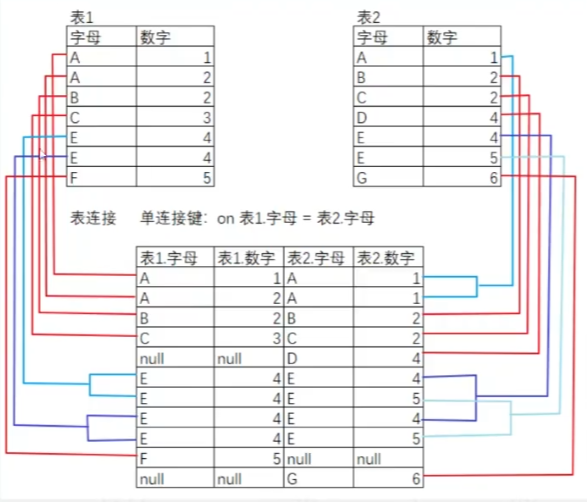
选择显示哪些字段

* + - * 1. Excel
    1. 代码
       1. select
       2. 品牌名称
       3. ,门店名称
       4. ,日期
       5. ,GMV
       6. ,平台
       7. ,下单人数
       8. from [ddm.shop](http://ddm.shop/)
       9. where 日期 between '2019-12-01' and '2019-12-07'
       10. and 平台 = 'meituan'
  1. 场景：查询所有门店每天的GMV和CPC消耗【join表链接】（GMV和CPC在两个不同的表格里，如何把它们放到一个表格中）
     1. 方法
        1. 使用表连接 -join
     2. 语法
        1. select 字段名
        2. from 表格名1
        3. join 表格名2
           1. join 到底是什么

表面上，join是各种逻辑的表连接

实际上，join是一种运算逻辑（算法）

**遍历匹配所有，不断向下增添**



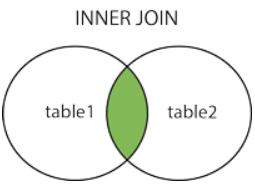
【Excel演示】 理解左/右/内 连接

单连接键

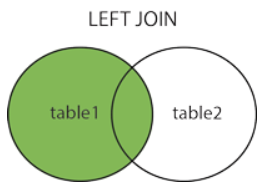
双连接键

inner/left/right/full 只是基于连接键是否为空值的筛选方式

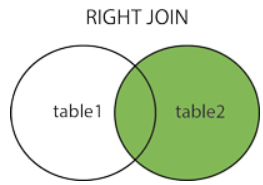
join = inner join



left join



right join

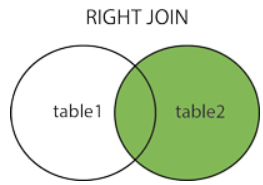


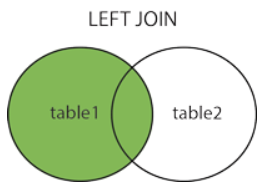
full join

mysql 不支持完全外部连接

那要如何实现呢？

需要union 连个左右连接





union 和 union all 的区别

1、对重复结果的处理不同

union 去重，union all 不去重

2、对排序的处理不同

注意

on 后面的筛选条件和where的筛选条件有什么不同？

on比where运行的早，数据库会根据on 的条件对数据进行筛选再进行连接

而where 则是连接完成后，再进行筛选

* + - 1. on 连接键1
      2. and 连接键2
      3. 注
         1. from 表格1 left join 表格2 on 连接键
         2. 等价于
         3. from 表格2 right join 表格1 on 连接键
    1. 代码
       1. select
       2. 门店名称
       3. ,shop.日期
       4. ,GMV
       5. ,cpc总费用
       6. from [ddm.shop.shop](http://ddm.shop.shop/)
       7. join ddm.cpc.cpc
       8. on shop.门店ID = cpc.门店ID
       9. and shop.日期 = cpc.日期
  1. 场景：业务提新需求了，还要看旗下所有品牌各门店12月1日至7日期间在所有平台上的总GMV和总下单人数【group by】
     1. 原理
        1. 【Excel演示】 数据透视表
     2. 语法
        1. 语法
           1. select
           2. 字段名
           3. ,聚合函数
           4. from 表名
           5. where 条件
           6. group by 字段名1

SQL

对数据进行分组 = 对group by 后的字段进行去重合并，并作为后续聚合运算的依据

因为group by先运行，并且运行后，表格中的非聚合字段已经形成了，所以**select 后的非聚合字段一定要与group by 后指定的字段一致**

为什么group by 可以使用1 ，2 等数字直接引用select 后的字段？ select不是最后运行的吗？

因为，select 会运行两次

第一次最先运行：按照字段原始名称和别名新建表头

代码还没运行完毕，表头就已经生成了

第二次最后运行：基于from、join、where、group by、having处理好的数据表格，按照select后的计算规则，计算并在表头后显示字段对应的数值

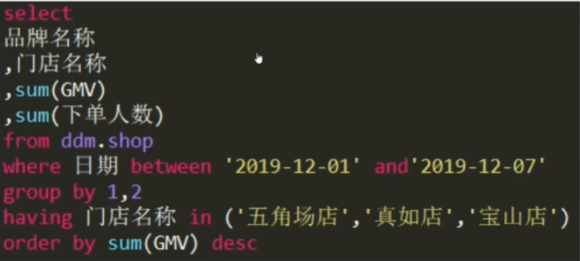
所以group by 可以使用1,2等数字直接引用select后的字段名称，直接用于去重合并

顺便提一句，order by 也可以直接使用数字引用字段

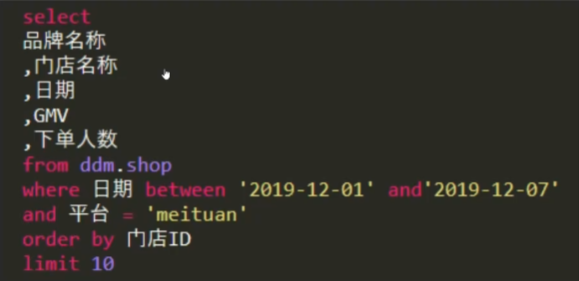
Excel

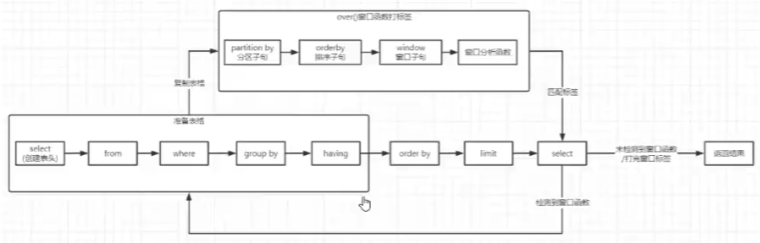
数据透视表 - 拖拽字段到**行** （去重合并）

* + - 1. 用法举例
    1. 代码
       1. select
       2. 品牌名称
       3. ,门店名称
       4. ,sum(GMV)
       5. ,max(GMV)
       6. from [ddm.shop](http://ddm.shop/)
       7. group by 品牌名称, 门店名称, 平台
  1. 场景：业务看了眼刚才的数据，表示只想看累计GMV在3万以上，并且下单人数在200人以上的门店【having】
     1. 原理
        1. 【Excel演示】
           1. 在数据透视表中对聚合的值进行筛选
     2. 语法
        1. select
        2. 字段名1
        3. ,聚合函数
        4. from表格名
        5. where条件
        6. group by 字段名1
        7. having 条件
           1. 由于SQL的单向执行，而where已经在group by之前执行过了，如果再对聚合运算的结果进行筛选，需要使用新语句having
           2. 并且，having是在分组（去重合并）后运行的，因此**having只可以对作为分组依据的非聚合字段和任意分组后聚合运算的结果**进行筛选，除了对象，having具体的语法和where完全一致
           3. **只可以对作为分组依据的group by 后的非聚合字段**进行筛选(浪费资源，不建议这样做，可以直接在where里做)



* + - * 1. 对**任意分组后聚合运算的结果**进行筛选
    1. 代码
       1. select
       2. 品牌名称
       3. ,门店名称
       4. ,sum(GMV)
       5. ,sum(下单人数)
       6. from [ddm.shop](http://ddm.shop/)
       7. where 日期 between '2019-12-01' and '2019-12-07'
       8. group by 1, 2
       9. having sum(GMV) > 30000
       10. and sum(下单人数) > 200
  1. 场景：对表格进行排序，显示指定行数的数据【order by & limit】
     1. order by
        1. 按照字段的顺序对表格的行进行排序，默认升序，字段后加DESC为降序
        2. order by 在select前运行，因此可以依据select后没有，但表格中有的字段进行排序

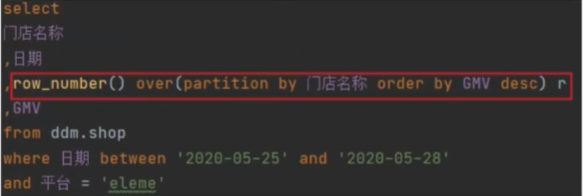




* + - 1. 注意：order by 字段内容为中文时，可能会出现排序混乱的情况
         1. 原因是编译MySQL时一般以ISO-8859字符集作为默认的字符集，比较过程中中文编码字符大小写转换造成了这种现象
         2. 解决方法是写成order by convert (字段名 using gbk)
         3. 例如：order by convert(门店名称 using gbk）
    1. limit
       1. 按照默认排序或order by 排序后，限制最终显示表格的行数
       2. limit n则显示前n行，如：limit 10 显示前10行
       3. limit x,y 则从第x行后显示y行，即取x+1行到x+y行，共y行
          1. limit 10,2会显示第11行和12行，共2行
          2. limit 10,4从10行后去4行，会显示第11行到第14行，共4行
          3. 代码演示
  1. 场景：查询2020年饿了么平台上每个门店GMV最高那天的日期和GMV【窗口函数】
     1. 方法
        1. order by 和 limit**只针对取整个表格内部排名第一**的数值，无法指定分区，取分区内的第一
        2. 可以使用窗口函数对表格分区，对分区内的数据进行相关计算
     2. 语法
        1. select
        2. 字段名
        3. ,窗口函数
           1. 函数名()over (partition by字段名 order by 字段名）
           2. 窗口函数到底是什么？

表面上，窗口函数是基于某个排序规则进行排序并计算

row\_number()over(partition by 门店名称 order by GMV desc）



按门店名称分组，在门店名称这个分组内部，按GMV降序排列，然后在分组内部依据GMV排序给每行分配一个序号



实际上，窗口函数是对准备好的表格，在任意分组内部进行排序和计算（为了区分，我们称之为分区）（也就是指定任意详细级别进行各种自定义的运算）

代码

select

门店名称

,平台

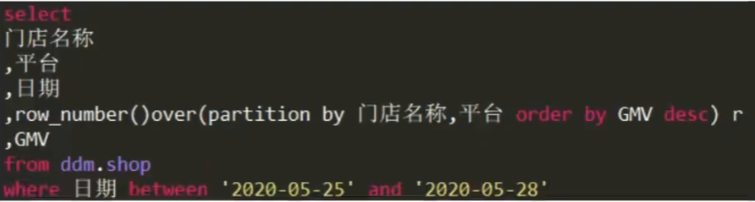
,日期

,row\_number() over(partition by 门店名称, 平台order by 日期) r

,GMV

from [ddm.shop](http://ddm.shop/)

where 日期between '2020-05-25' and '2020-05-28'



再创建一个之前准备好的表格（还未排序和限制）

from [ddm.shop](http://ddm.shop/)

where 日期 between ‘2020-05-25’ and ‘2020-06-28’

根据partition by 后的字段（没有则默认针对整个表）形成的分区（不去重合并）【Excel演示】

over (partition by 门店名称，平台order by 日期)

先对门店名称分区，再对该区中的平台分区

注意：mysql支持over聚合字段

select

门店名称

,平台

,sum(gmv)

,row\_number() over(partition by 门店名称 order by sum(gmv)) r

from [ddm.shop](http://ddm.shop/)

group by 1, 2

在分区内根据order by 后的条件对所有行进行排序 ---- 以上创建一个新的表格，并重新进行分区排序就是“over”的本质

**计算over()前函数的结果【窗口函数合集】**

**各种聚合运算都支持**

sum()over()

over()中分为partition by 子句、order by子句和window子句

只使用partition by 子句，未指定order by，聚合为分组内的聚合

使用partition by 和order by 子句，未使用window子句的情况下，默认从起点到当前行

**window子句**

preceding：往前

following：往后

current row： 当前行

unbounded：起点

unbounded preceding 表示从前面的起点

unbounded following 表示到后面的终点

区别下面几个窗口函数结果差别

,sum(GMV) over(partition by 门店名称, 平台) 分区 和

,sum(GMV) over(partition by 门店名称, 平台 order by GMV desc) 分区第1行到当前行的累计和

,sum(GMV) over(partition by 门店名称, 平台 order by GMV rows 2 preceding) 分区前2行和当前行的累计 和

,sum(GMV) over(partition by 门店名称, 平台 order by GMV desc rows between unbounded preceding and 1 preceding) 分 区第1行到前1行的累计 和

avg()over()

count()over()

max()over()

min()over()

根据排序分配序号

row\_number() : 根据分区内的排序，分配唯一且连续排名序号(1 2 3)

row\_number() over(partition by 门店名称，平台 order by 日期 ) r

dense\_rank(): 根据分区内的排序，分配不唯一且连续排名序号(1 1 2 2 )

,dense\_rank() over(partition by 门店名称， 平台 order by 日期) dense\_r

rank() : 根据分区内的排序，分配不唯一且不连续排名序号(高考排名逻辑 1 1 3)

,rank() over(partition by 门店名称，平台 order by 日期）rank\_r

根据排序取具体字段数值

first\_value(字段): 根据分区内的排序，返回排在第一行的值（降序就可以取最后一行对应的数值）

,first\_value(GMV) over(partition by 门店名称, 平台 order by 日期) first\_gmv

,first\_value(GMV) over(partition by 门店名称, 平台 order by 日期 desc) lastgmv

nth\_value（字段，n）: 返回分区内的排序第n行的字段数值。如果第n行还未运行到，则返回NULL。n必须为正整数，例如 1，2

,nth\_value(GMV,3) over(partition by 门店名称, 平台 order by 日期） nth\_gmv\_3

根据排序取上下几行的数值（计算同环比）

lag(字段，n，默认值)：返回分区内，本行前n 行的字段数值，如果为空，则填充默认值

lag(gmv, 2， 666) over(partition by 门店名称, 平台, order by 日期）

lead（字段，n, 默认值）：返回分区内，本行后n行的字段数值，如果为空则填充默认值

lag(gmv, 2, 666) over(partition by 门店名称, 平台, order by 日期) lag\_2b

根据排序统计分布位置

percent\_rank(): 根据分区内的排序，从0开始统计当前行所在排序中的处于百分之多少的位置（不管指定值与分组序列中某值是否重复，均将此值视为序列一部分）

percent\_rank()也是 0<= percent\_rank() <= 1, 0和1都可以取到

percent\_rank()的计算公式应该是 = （（统计的列的种类））/（统计的列的总数）\*100%，但是第一个总是从0开始的，不管重复几次，要是567有4次，那就是有四个都是0，最后一个是1

,percent\_rank() over (partition by 门店名称, 平台 order by 日期) perpos

cume\_dist(): 根据分区内的排序，从0之后开始统计当前行所在排序中的百分比分布位置（如果指定值与分组序列中某值重复，则将二值视为一个值处理）

cume\_dist()函数统计的范围是 0 < cume\_dist() <= 1，但是取不到0

cume\_dist()的计算公式应该是 = （（统计的列的种类）/（统计的列的总数））\*每个重复种类中的个数；如value值为567的两个值，公式就是（4/5）\*2=0.4。而最大就是1了。

,cume\_dist() over(partition by 门店名称, 平台 order by 日期) pos

在排序内分组

ntile(n): 将排序内的行分为n组，根据分区内的排序，返回每一行是第几组

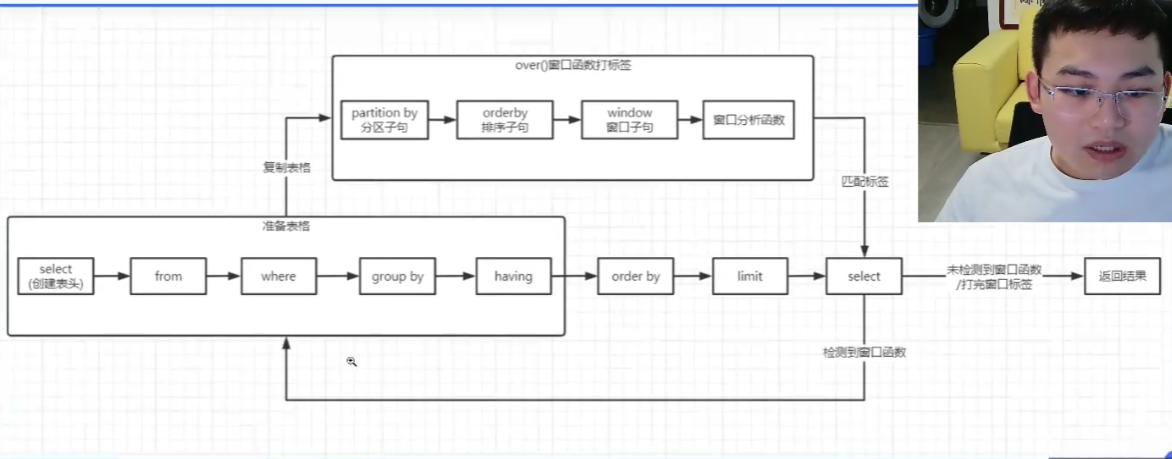
,ntile(4) over (partition by 门店名称, 平台 order by 日期) ntile\_r

窗口函数合集

将结果返回到最终查询出的表格

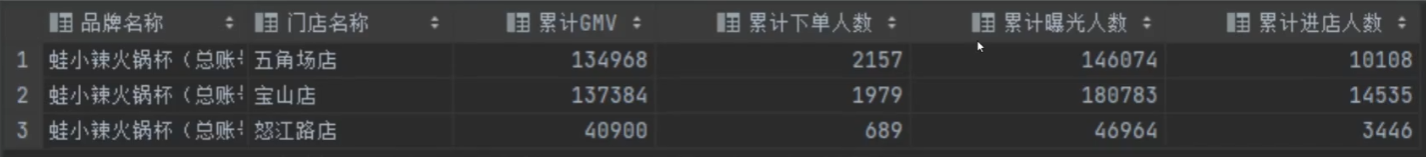
注：窗口函数只能写在select后面

* + - 1. from 表格名
  1. SQL基础语法与运行原理总结
     1. SQL四期-L组-Hee整理



* + 1. 准备表格
       1. 复制源数据进行筛选
          1. select：按照字段原始名称或别名新建表头和计算逻辑，方便后续引用
          2. from：从以下数据库中取出表格，复制一个临时的表格用于后续处理
          3. join：根据字段匹配连接各个表格，生成一个临时的“宽表”用于后续处理
          4. where：将数据按以下条件进行筛选
       2. 如果需要进行聚合运算，则基于连接、筛选好的数据，创建“数据透视表”
          1. group by : 分组（去重合并）指定字段，作为聚合运算的依据
          2. having：对去重合并的字段或聚合运算的结果进行筛选
    2. 查询字段
       1. 对准备好的表格进行排序和提取
          1. order by：根据以下字段的升降序进行排序，可以直击引用字段别名和序号
          2. limit：只提取指定行数或指定位置的数据
       2. 基于提取后的数据，进行计算和填充
          1. select：在已经准备好了的表格中，保留并新增计算字段
    3. 窗口函数
       1. 基于准备好的表格（还未order by 和limit）

1. 分步解题法：看/想到哪写到哪
   1. 第一步：翻译它的数据需求，让它说人话
   2. 第二步：看/想到哪，就写到哪
   3. 第三步：重新组合代码
   4. 第四部：穷尽思考，优化代码
      1. 用if函数省略子查询
      2. 直接用having筛选
      3. 表连接进行数据筛选
      4. 窗口函数实现任意级别聚合运算
2. 10道真题搞定SQL基础语法
   1. 基础语法真题
      1. 1、2020年第一季度下单人数前3名的门店，业绩表现如何 <1>



* + - 1. 分步实现法：看到哪写到哪
         1. 第一步：翻译它的数据需求，让它说人话

2020-01-01到2020-03-31期间累计下单人数排名前3的门店

业绩表现指什么？

累计GMV、累计下单人数、累计曝光人数、累计进店人数

* + - * 1. 第二步：看/想到哪，就写到哪

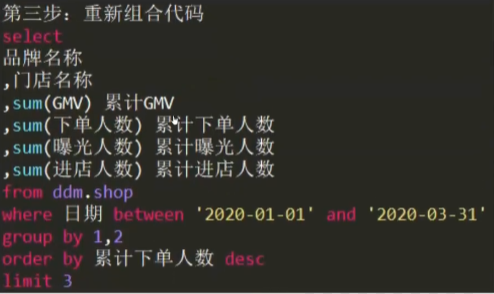
where 日期 between '2020-01-01' and '2020-03-31'

,sum(下单人数)累计下单人数

order by 累计下单人数 desc

limit 3

* + - * 1. 第三步：重新组合代码



* + 1. 2、2020年第一季度每个月，每个门店的累计GMV <2>



* + - 1. 2020年第一季度
         1. where 日期 between '2020-01-01' and '2020-03-31'
      2. 每个月
         1. group by 月份
      3. 每个门店的累计GMV
         1. select
         2. 门店
         3. ,sum(GMV)
      4. 组合代码
         1. select
         2. 门店
         3. ,sum(GMV)
         4. from [ddm.shop](http://ddm.shop/)
         5. where 日期 between '2020-01-01' and '2020-03-31'
         6. group by 月份
      5. 月份怎么算
         1. 取日期的前七位不就行了
         2. 函数substring（字段，从第几位开始，向后取几位）

substring（日期,1,7）

‘2020-01-01’ --- '2020-01'

* + - * 1. substring（日期，1,7）月份
      1. group by要前后对应，品牌名称默认要加上（实际场景就是这样，业务会忘，你不能忘）
         1. select
         2. 品牌名称
         3. ,门店名称
         4. ,substring(日期,1,7) 月份
         5. ,sum(GMV)
         6. from [ddm.shop](http://ddm.shop/)
         7. where 日期 between ‘2020-01-01’ and ‘2020-03-31’
         8. group by 1,2,3
         9. order by 品牌名称, 门店名称, 月份
    1. 3、2020年第一季度曝光人数大于10w的门店，每个月的GMV <3>
       1. 2020年第一季度
          1. where 日期 between '2020-01-01' and '2020-03-31'
       2. 曝光人数大于10w
          1. having sum(曝光人数) > 100000
       3. 的门店
          1. select
          2. 门店名称
       4. 每个月的GMV
          1. sum(GMV)
          2. group by substring(日期,1,7)
       5. 组合代码
          1. select
          2. 品牌名称
          3. ,门店名称
          4. ,substring(日期,1,7) 月份
          5. ,sum(GMV)
          6. ,sum(曝光人数) 累计曝光人数
          7. from [ddm.shop](http://ddm.shop/)
          8. where 日期 between '2020-01-01' and '2020-03-21'
          9. group by 1,2,3
          10. having sum(曝光人数) > 100000
       6. 有什么问题？
          1. 2020年**第一季度曝光人数**大于10w的门店，**每个月的GMV**
          2. 曝光人数要依据季度算，GMV要依据月份算
          3. 但是一次查询只能有一个聚合运算的依据，怎么办？

先把这三个门店查出来，再计算这三家门店不就行了？

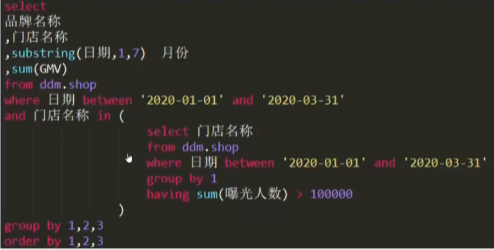
where子查询

筛选条件的聚合计算依据 与需要聚合运算的字段的聚合计算语句不同时使用

1、同一个表格依据不同

2、根本不在同一个表格

* + - 1. 先把三个门店查出来
         1. select
         2. 门店名称
         3. from [ddm.shop](http://ddm.shop/)
         4. where 日期 between '2020-01-01' and '2020-03-31'
         5. group by 1
         6. having sum(曝光人数) > 100000
      2. 再用这三个门店做筛选条件



* + 1. 4、2020年第一季度cpc总费用小于1000的门店（门店ID不同视为不同门店），每个月的GMV <4>
       1. 2020年第一季度
          1. where 日期 between '2020-01-01' and '2020-03-31'
       2. cpc总费用小于1000
          1. having sum(cpc总费用) < 1000
       3. 的门店
          1. select
          2. 门店名称
       4. 每个月
          1. substring(日期,1,7)
       5. 的GMV
          1. sum(GMV)
       6. 代码汇总
          1. select
          2. 品牌名称
          3. ,门店名称
          4. ,substring(日期, 1, 7) 月份
          5. ,sum(GMV)
          6. from [ddm.shop](http://ddm.shop/)
          7. where 日期 between '2020-01-01' and '2020-03-31'
          8. and 门店ID in
          9. (

select 门店ID

from ddm.cpc

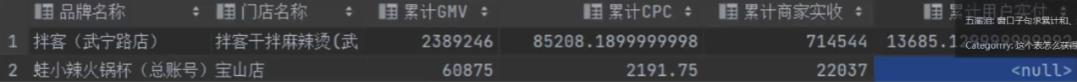
where 日期 between '2020-01-01' and '2020-03-31'

group by 1

having sum(cpc总费用) < 1000

* + - * 1. )
        2. group by 1, 2, 3
        3. order by 1, 2, 3
  1. 表连接真题
     1. 6、所有门店20年7月的累计GMV、累计CPC、累计商家实收、累计用户实付 <6> 笛卡尔积
        1. 能不能把三张表直接连接在一起？
           1. 直接连接会出现笛卡尔积





* + - * 1. 当你发现连接键无法让两边数据的行一一对应时，可能就需要先聚合运算再连接了
    1. 7、拌客在饿了么上20年第二季度每个月的GMV，以及每个月GMV在20年第二季度累计GMV的占比



* + - 1. 占比怎么算？
         1. 先算出月累计GMV再算出季度累计GMV，相除就好了
      2. 怎么调整成百分比？
         1. 字符串拼接(CONCAT)

将两个字符串进行拼接

concat(字段1,'wdnmd',字段2)

字段1wdnmd字段2

* + - * 1. ROUND()

对某个数值字段进行指定小数位数的四舍五入

负数则往前四舍五入

* + - 1. 最终代码
         1. select
         2. a.品牌名称
         3. ,a.月份
         4. ,a.单月GMV
         5. ,b.季度GMV
         6. ,concat(round((a.当月GMV/b.季度GMV)\*100,0),'%') 占比
         7. from
         8. (

select

品牌名称

,substring(日期,1,7) 月份

,sum(GMV) 当月GMV

from [ddm.shop](http://ddm.shop/)

where 日期 between '2020-04-01' and '2020-06-30'

and 品牌名称 like '%拌客%'

and 平台 = 'eleme'

group by 1,2

* + - * 1. ) a
        2. left join
        3. (

select

品牌名称

,sum(GMV) 季度GMV

from [ddm.shop](http://ddm.shop/)

where 日期 between '2020-04-01' and '2020-06-30'

and 品牌名称 like '%拌客%'

and 平台 = 'eleme'

group by 1,2

* + - * 1. ) b
        2. on a.品牌名称 = b.品牌名称
      1. 研究一下窗口函数，更简单哦~
         1. select
         2. 品牌名称
         3. ,substring(日期, 1, 7) 月份
         4. ,sum(GMV) 当月GMV
         5. ,concat(round((sum(GMV)/sum(sum(GMV))over())\*100,0),'%') 占比
         6. from [ddm.shop](http://ddm.shop/)
         7. where 日期 between '2020-04-01' and '2020-06-30'
         8. and 品牌名称 like '%拌客%'
         9. and 平台 = 'eleme'
         10. group by 1, 2
  1. 窗口函数真题
     1. 8、2020年拌客和蛙小辣两个品牌在饿了么和美团两个平台上各自GMV最高的三天以及这三天的GMV <9>
        1. select \*
        2. from
           1. (
           2. select
           3. 品牌名称
           4. ,平台
           5. ,日期
           6. ,GMV
           7. ,row\_number()over(partition by 品牌名称, 平台 order by GMV desc) r
           8. from [ddm.shop](http://ddm.shop/)
           9. where substring(日期,1,4) = '2020'
           10. and (品牌名称 like '%蛙小辣%' or 品牌名称 like '%拌客%')
           11. ) a
           12. where r in (1,2,3)
     2. 9、2020年拌客和蛙小辣两个品牌各自门店在饿了么GMV排名前5%（GMV为0不计入排名）的日期、GMV、以及具体排名和排名百分比（取一位小数） <11>
        1. 用percent\_rank() 还是cume\_dist()
        2. 代码
           1. select
           2. 品牌名称
           3. ,门店名称
           4. ,日期
           5. ,GMV
           6. ,r 排名
           7. ,concat(round(r\_per\*100,1),'%') 百分比排名
           8. from
           9. (

select

品牌名称

,门店名称

,日期

,GMV

,row\_number()over(partition by 品牌名称,门店名称 order by GMV desc) r

,cume\_dist()over(partition by 品牌名称,门店名称 order by GMV desc) r\_per

from [ddm.shop](http://ddm.shop/)

where substring(日期,1,4) = '2020'

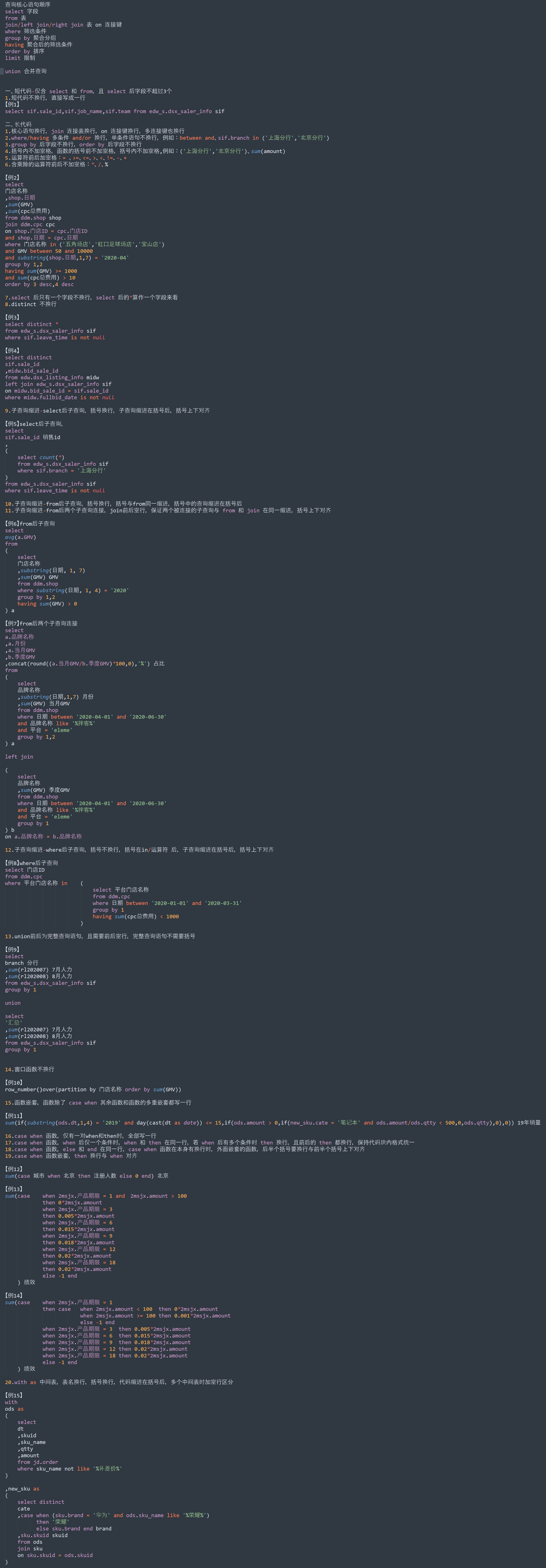
and (品牌名称 like '%蛙小辣%' or 品牌名称 like '%拌客%' )

and 平台 = 'eleme'

and GMV > 0

* + - * 1. ) a
        2. where r\_per <= 0.05
    1. 10、2020年拌客在双平台每日GMV及前一日GMV，及GMV周同比增长与百分比 <12>
       1. 前一日GMV怎么算？
          1. lag(gmv,1,0)
       2. 一周前GMV怎么算？
          1. lag(GMV,7,0)
          2. 也可以lead(gmv,-7,0)，但是我们课程的mysql的lead函数不支持负数查询
       3. 周同比增长百分比怎么算？
          1. （当日 - 一周前）/ 一周前
          2. 当日/一周前 - 1
       4. 代码
          1. select
          2. 品牌名称
          3. ,平台
          4. ,日期
          5. ,GMV 当日GMV
          6. ,lag(gmv,1,0)over(partition by 门店名称,平台 order by 日期) 前日GMV
          7. ,lag(gmv,7,0)over(partition by 门店名称,平台 order by 日期) 前一周GMV
          8. ,GMV-,lag(gmv,7,0)over(partition by 门店名称,平台 order by 日期) GMV 周同比增长绝对值
          9. ,(GMV - lag(gmv,7,0)over(partition by 门店名称,平台 order by 日期)) / lag(gmv,7,0)over(partition by 门店名称,平台 order by 日期) GMV周同比增长百分比
          10. from [ddm.shop](http://ddm.shop/)
          11. where substring(日期,1,4) = '2020'
          12. and 品牌名称 like '%拌客%'
       5. 通过子查询提升代码美观度
       6. select \*
       7. ,当日GMV - 前一周GMV GMV周同比增长绝对值
       8. ,concat(round(((当日GMV - 前一周GMV)/前一周GMV)\*100,0),'%') GMV周同比
       9. from
       10. (
           1. select
           2. 品牌名称
           3. ,平台
           4. ,日期
           5. ,GMV 当日GMV
           6. ,lag(gmv,1,0)over(partition by 品牌名称,平台 order by 日期) 前日GMV
           7. ,lag(gmv,7,0)over(partition by 门店名称,平台 order by 日期) 前一周GMV
           8. from [ddm.shop](http://ddm.shop/)
           9. where substring(日期, 1, 4) = '2020'
           10. and 品牌名称 like '%拌客%'
       11. ) a

1. 解题思路总结
   1. 1、尽可能使用分步解题法，顺着题目一步步写
   2. 2、写SQL代码时应该尽量脱离数据库，在脑海里完成代码的组合和调试
   3. 3、遇到特别复杂的逻辑，一次查询无法解决，可以直接上子查询，然后再慢慢优化
   4. 4、遇到暂时想不清楚的地方，可以先搁置，最后基于其他步骤已成型的代码进行推导
   5. 5、实在不行就画图，或者用excel表格先处理出来，再一步步用SQL实现
   6. 6、完成代码后一定要尽量优化自己的代码，这样逻辑才能越来越好
   7. 7、不断反复练习，最终能在读题时就在脑海里理清代码逻辑
2. 代码规范
   1. 代码千万条，注释第一条。代码不规范，亲人两行泪。



* 1. 可以帮你提升加薪的代码规范
     1. 0、只有select和from不换行，直接写成一行
     2. 1、核心语句不换行，join连接表换行，on连接键换行，多连接键也换行
     3. 2、where/having多条件 and/or 换行，单条件语句不换行，例如 ：between and 、 sif.branch in ('上海分行'，'北京分行')
     4. 3、group by 后字段不换行， order by 后字段不换行
     5. 4、括号内不加空格，函数的括号前不加空格，括号内不加空格，例如('上海分行','北京分行')、sum(amount)
     6. 5、运算符前加空格： =、 >=、 <=、 >、 <、 !=、 -、 +
     7. 6、含乘除的运算符前后不加空格：\*、/、%
     8. 7、select后只有一个字段不换行，select后的\*算作一个字段来看
     9. 8、distinct不换行，后面的字段大于一个字段换行
     10. 9、子查询缩进 -select 后子查询，括号换行，子查询缩进在括号后，括号上下对齐
     11. 10、子查询缩进-from后子查询，括号换行，括号与from同一缩进，括号中的查询缩进在括号后
     12. 11、子查询缩进-from后两个子查询连接，join前后空行，保证两个被连接的子查询与from和join在同一缩进，括号上下对齐
     13. 12、子查询缩进-where后子查询，括号不换行，括号在in/运算符后，子查询缩进在括号后，括号上下对齐
     14. 13、union前后为完整查询语句，且需要前后空行，完整查询语句不需要括号
     15. 14、窗口函数不换行
     16. 15、函数嵌套，函数除了case when 其余函数和函数的多重嵌套都写一行
     17. 16、case when 函数，仅有一对when 和then时，全部写一行
     18. 17、case when 函数，when后仅一个条件时，when和then在同一行，若when后有多个条件时then换行，且前后的then都换行，保持代码块内格式统一
     19. 18、case when函数，else和end在同一行，case when 函数在本身有换行时，外面嵌套的函数，后半个括号要换行与前半个括号上下对齐
     20. 19、case when函数嵌套，then 换行与 when对齐
     21. 20、with as 中间表，表名换行，括号换行，代码缩进在括号后，多个中间表时甲空行区分
     22. 21、别名尽量统一缩进
     23. 22、代码结束处加；
     24. 23、多写注释

1. **刷题提升**
   1. 牛客网数据库SQL实战一定要刷完！
      1. <https://www.nowcoder.com/ta/sql> (93题)
   2. 历年大厂真题里的SQL有时间就要记得去做一做
      1. <https://www.nowcoder.com/contestRoom?mutiTagIds=894>