现在正式进去MYSQL命令的学习

打开HeidiSQL软件，进入MySQL窗口，准备敲代码吧！

当然，我们要用官方文档的标准例子练习

就像上节一样先导入create.txt创建好数据表

再导入populate.txt的内容更新数据表内的内容

再罗嗦一点吧，记得刷新

然后就可以对照《MySQL必知必会》进行练习

或者跟着老师的视频一一敲出来

进入products数据表，点击查询

SELECT vend_id FROM products;

就能显示出来products数据表内的vend_id列

以下笔记就要粗糙很多，需要专心学习数据库命令了

就不再多言

来也匆匆，去也匆匆，恨不能相逢！

第一次学习

SELECT ..... FROM ..... ： 从xx数据库导入xx数据表

WHERE ： 找出符合条件的数据

ORDER BY ： 可以根据SELECT的顺序取编号

DESC ： 倒序排序，如果全部都采取倒序，必须单独对每一列加上DESC

IN ： 后面接元组形式的数据

AND ： 交集

OR : 并集

BETWEEN 3 AND 8 ： 找出符合3到8之间的数据

NOT ： 否定。注意：MariaDB支持NOT否定IN、BETWEEN和EXISTS等

第二次学习

WHERE ..... LIKE 'R%y' ： LIKE是普通查找，查找以R开头的数据

WHERE ..... LIKE '%bear' : 查找以数字开头的数据

WHERE ..... LIKE '_ inch%bear' : 一个下划线接一个空格，查找以1位数字开头的

WHERE ..... LIKE '__ inch%bear' : 两个下划线接一个空格，查找以2位数字开头的

WHERE ..... REGEXP '[jm]' : REGEXP是正则查找，不区分大小写，查找包含j或m数据

WHERE ..... REGEXP '^[JM]' : 不区分大小写，查找以j或m开头的数据

WHERE ..... REGEXP '^j' : 不区分大小写，查找以j开头的数据

WHERE ..... REGEXP 'smith$' : 不区分大小写，查找以smith结尾的数据

SELECT Concat (vend_name,'(',vend_country,')') : 以Bear Emporium(USA)的方式来寻找数据并组合

SELECT Concat (vend_name,'_',vend_country)： 以Bear Emporium_USA的方式来寻找数据并组合

AS .....： 将找到的数据起个别名，列出数据

SELECT Concat (RTRIM(vend_name),'_',RTRIM(vend_country))： 另一种找到数据并组合的方法

SELECT LTRIM(' MASTER ') : 给找到的数据插入字符串

SELECT quantity*item_price AS total_price： 抓取到的两组数据取乘积，将结果得到别名产生新列

第三次学习

WHERE YEAR(order_date) = 2012 : 自动对日期数据提取年的数据

SELECT AVG (prod_price) AS avg_price： 将prod_price取所有数的平均数得到的结果放在新列

SELECT COUNT(cust_email) AS num_cust : 数有多少个非空数据

SELECT MIN(prod_price) AS min_price： 获取最小值的数据

SELECT MAX(prod_price) AS max_price： 获取最大值的数据

SELECT vend_id,COUNT(*) AS num_prods FROM products

WHERE prod_price >= 3.99

GROUP BY vend_id

HAVING COUNT(*) >= 3

```
BRS01 3
```

首先需要声明的是，HAVING 强制放最后，GROUP BY 倒数第二

现在一个个分析数据的产生过程

第一行是抓取所有vend_id和所有行的数,但是没有分组，最终结果只显示一行

```
BRS01 9
```

然后加上GROUP BY vend_id，就有了正确的分组

```
BRS01 3
DLL01 4
FNG01 2
```

然后我们最后一行加一个条件 HAVING COUNT(*) >= 3，对coutn数进行了筛选：

```
BRS01 3
DLL01 4
```

最后再加一个价格筛选，WHERE prod_price >= 3.99，最终结果如下

```
BRS01 3
```

有些人可能就有点疑惑了，DLL01组不是有1个数据是4.99吗?符合条件也被筛选

请注意，下面有个HAVING COUNT(*) >= 3一直在限制，必须要总数超过3才能列出来噢

所以得出一个非常重要的结论

COUNT(*) 的用法必然伴随GROUP BY，不然只能产生一行数据

真正放在最后的其实是ORDER BY，就是说将最终的数据列出来的时候，还能再进行排序

于是有下面的标准执行语句顺序：

```
SELECT      要返回的列表或表达式
FROM      从中检索数据的表
WHERE      行级过滤
GROUP BY      分组说明
HAVING      组级过滤
ORDER BT     输出排序顺序
```

第四次学习

使用子查询,无非就是外键数据的查询，使用IN，嵌套SELECT

第一个例子

SELECT cust_id FROM orders

WHERE order_num IN

(

SELECT order_num FROM orderitems

WHERE prod_id = 'RGAN01'

);

以上的例子就是一个很经典的子查询，利用外键关联的关系筛选到真正想要的数据

这个就是想要查询产品ID为'RGAN01'的订单在顾客数据表中所对应的顾客id

第二个例子

SELECT cust_name , cust_state,

(

```
SELECT COUNT(*) FROM orders

WHERE orders.cust_id = customers.cust_id
```

) AS orders

FROM customers

ORDER BY orders

这个例子虽然很不好理解，那我就慢慢讲，首先我们要处理两个表，一个是顾客表一个是订单表

那么我们在订单表中，有5个订单，当然是不同的顾客买的，有重复买的，有没买的

当然也要加上州信息，因为我们发现有顾客重名了

我们所要做的就是将所有顾客的信息列出来，然后后面做一个新列来展示顾客购买的订单量，OK，let's GO！！！

首先肯定是SELECT cust_name , cust_state, () FROM customers啊，先把模板做好

这是一个良好的习惯，要先确定好自己的目标，打好模板，

然后这个() 就用来AS orders作为别名，展现出顾客购买的订单数量

接下来就是对这个() 处理了，可以使用计算的方法来获取订单数量

首先SELECT订单表，获取到里面的订单数据，然后将两个表的顾客id一起对应起来

最后排个序，不就得到了这样的结果吗?

```
Kids Place      OH 0

Fun4All         IN 1

Fun4All         AZ 1

The Toy Store   IL 1

Village Toys    MI 2
```

第三个例子

SELECT vend_name,prod_name,prod_price

FROM vendors AS V,products AS P

WHERE V.vend_id = P.vend_id;

这个例子放出来只是想告诉自己，要善用别名，很重要

第四个例子

SELECT c1.cust_id,c1.cust_name,c1.cust_contact

FROM customers AS c1,customers AS c2

WHERE c1.cust_name = c2.cust_name

AND c2.cust_contact = 'Jim Jones'

这个是经过优化的代码，为什么要用这个方式呢？原代码是使用同一个数据表内反复检索，定位到精准的数据，但是在子查询中再次使用SELECT检索同样的数据库，是不利于性能的，所以就采用开头就检索两遍同一数据库，分别取别名c1，c2，然后以c2的某一内容找到顾客的名字，在c1筛选出此顾客的id和顾客名字和顾客发表的内容

内联结和外联结

SELECT customers.cust_id , orders.order_num

FROM customers INNER JOIN orders

此方法叫做内联结，分别获取两个表的两个列，然后联结数据到一起

ON customers.cust_id = orders.cust_id

在使用内联结之后，当然还要整合一下重复的数据，就使用ON来整合cust_id

当然，外联结的方法本质也没有什么区别，但是要弄对方法

将文中的INNER 改成 RIGHT JOIN，没什么区别，因为确实是把orders联结到右边

但是如果改成LEFT RIGHT JOIN左联结的话,数据会发生变化

会把原本没有对应的cust_id给显示出来，右边的订单号会显示null

第五次学习

怎么样使用组合查询

SELECT cust_name , cust_contact,cust_email

FROM customers

WHERE cust_state IN ('IL','IN','MI');

SELECT cust_name,cust_contact,cust_email

FROM customers

WHERE cust_name = 'Fun4All';

运行这些代码，自然就产生了两个查询结果窗口，问题是，该怎么样使用组合查询弄到一个窗口呢？

其实去掉第一个分号，然后在中间插入个UNION，就能得到组合查询的结果了

更新一个数据

SELECT cust_name , cust_contact,cust_email

FROM customers

WHERE cust_contact = 'Michelle Green'

执行上面的语句我们会发现，cust_email的值为null，这个时候我们需要进行添加

UPDATE customers

SET cust_email = ' green@red.com'

WHERE cust_contact = 'Michelle Green'

使用UPDATE-SET-WHERE三步走：

先UPDATE选定更新的数据表，再SET选定新的列，最后WHERE定位具体位置

然后去数据表中查看更新情况，发现邮箱地址确实被加进去了

要注意,严格遵守三步走，少了一个WHERE的话，列的所有数据全会被更新，有安全隐患

创建一个视图

有些数据调用过来组合之后，如果想要对这个组合数据反复操作，应该怎么办呢?

CREATE VIEW vend_table AS

SELECT CONCAT (vend_name,'_',vend_country) AS name_country

FROM vendors

先看后面两行，能够得出最终的列，那么，该怎么保存这个列以便后面的使用呢?

当然是看看第一行的CREATE VIEW 创建视图了 后接命名 ，最后一个AS 是得出的列

执行成功之后，我们调试一下新产生的视图：

SELECT * FROM vend_table;

确实成功设计出来了一个视图，方便反复调用

数据的索引

- 什么是索引
  - 索引是允许更快地检索记录的一种优化方法。
  - 使用B+ tree或者hash算法，建立了数据与地址的对应关系

- 索引的作用
  - 索引为出现在索引列中的每个值创建一个记录。
  - 索引包含了排序的数据和一个指向原始数据的链接。
  - 在查询数据时，避免了一行行的进行查找而加快速度。

- 索引的类型
  - primary：主键
  - Unique：数据唯一，但可以为NULL
  - Multiple-column Indexes：多列合作当主键
  - KEY = INDEX

- 索引的操作
  - 使用primary key
  - 使用unique key