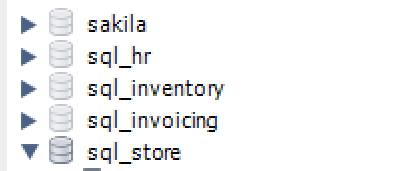
# Lecture2

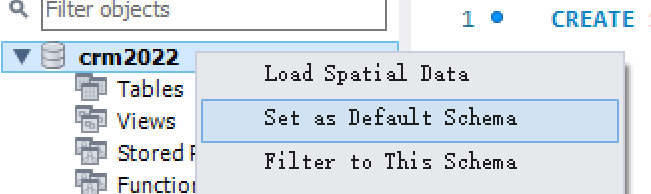
**Create the databases**

这里的每个叫做一个database

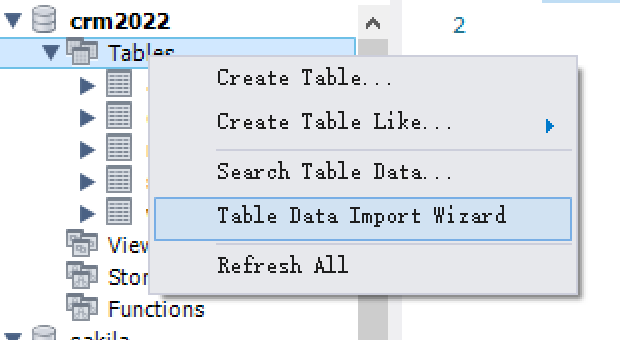
****

创建新的database

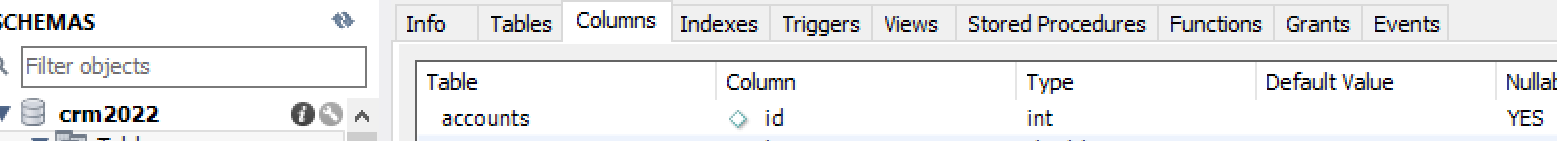
CREATE SCHEMA CRM2022;

****

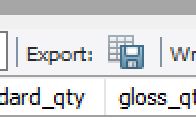
在database中导入table

****

可以看到信息

****

可以导出表格

****

One table has one **Primary key**

A primary key is a unique **column** in a particular table. This is the first column in each of our tables. Here, those columns are all called id, but that doesn't necessarily have to be the name.It is common that the primary key is the first column in our tables in most databases.

But can have several **Foreign key**

A foreign key is a column in one table that is a primary key in a different table.

### 关于截取，改变，连接，查找，替换，删除等等

而DENSE\_RANK 不会skip

例题：求每个account每一个月qty的最大最小总和排名

SELECT id, account\_id, standard\_qty, MONTH(occurred\_at) AS month,

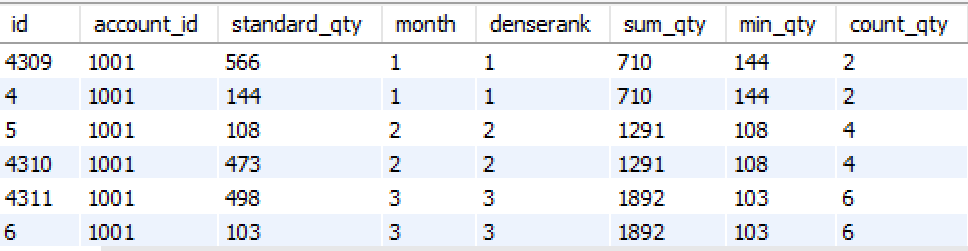
DENSE\_RANK() OVER (PARTITION BY account\_id ORDER BY MONTH(occurred\_at)) AS denserank,

SUM(standard\_qty) OVER (PARTITION BY account\_id ORDER BY MONTH(occurred\_at)) AS sum\_qty,

MIN(standard\_qty) OVER (PARTITION BY account\_id ORDER BY MONTH(occurred\_at)) AS min\_qty,

COUNT(standard\_qty) OVER (PARTITION BY account\_id ORDER BY MONTH(occurred\_at)) AS count\_qty

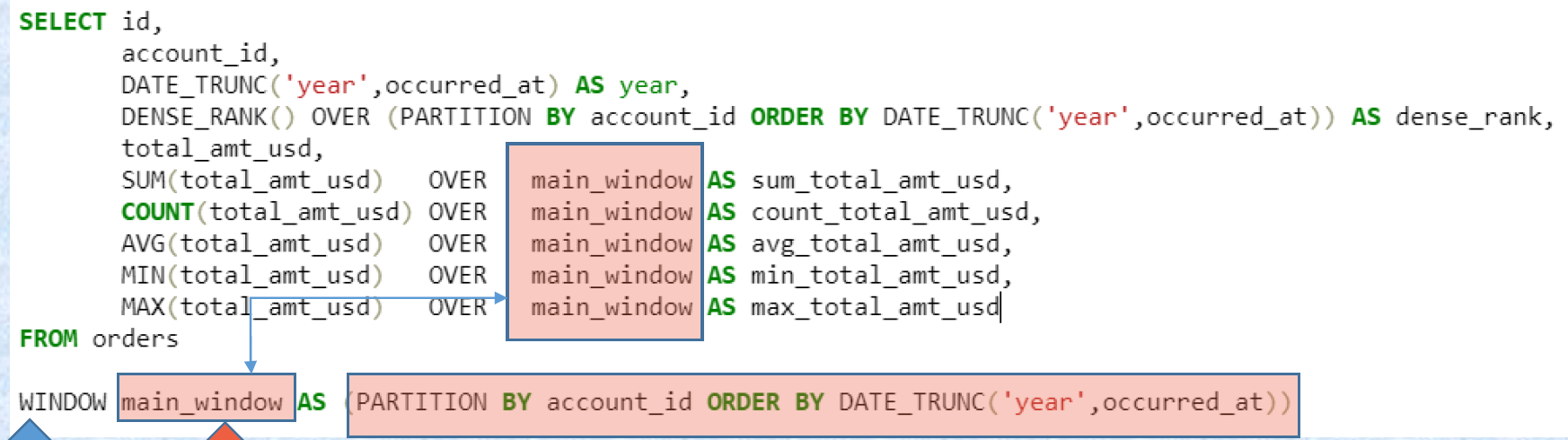
FROM orders;



使用alias去替代重复的部分

在上一题中，反复出现了PARTITION BY account\_id ORDER BY MONTH(occurred\_at)

可以在最后使用一个alias去全部代替，记住alias必须put **below** the main syntax句法

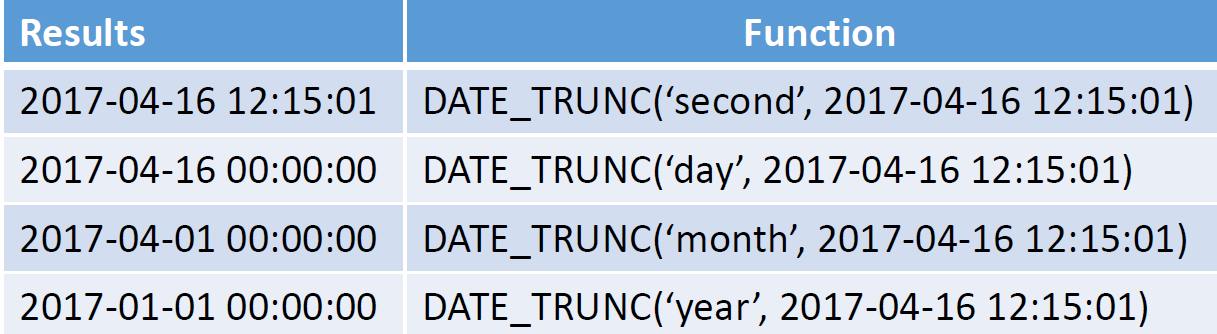


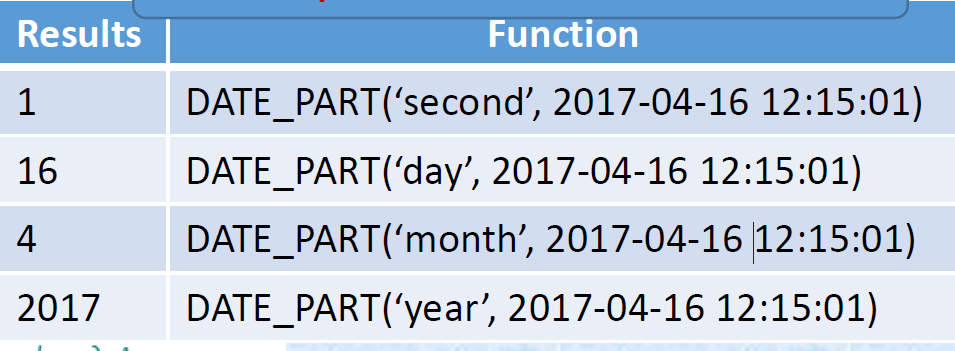
提取一部分时间

Date\_trunc和date\_part

DATE\_PART ‘

1代表周日





## WHERE语句

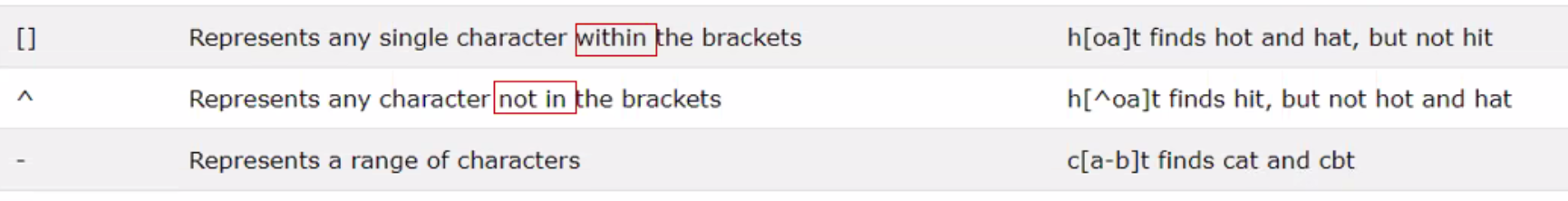
控制character数，并且包含某个东西

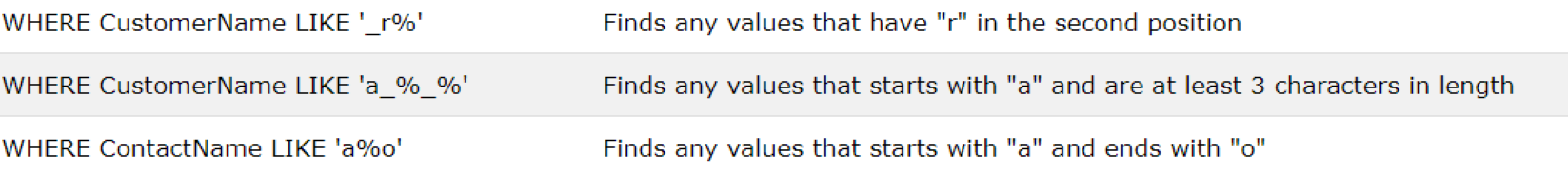
WHERE last\_name LIKE ‘\_y’ : 几个\_就代表有几个字符

[]代表bracket里面的任意character

^代表不在

-代表从..到..





Efficient coding

WHERE (A=0) OR (B=0) OR (C=0) 相当于 WHERE A\*B\*C=0

有/开头/结尾 %表示任意的东西

WHERE last\_name **LIKE** ‘b%’：以b开头 ‘%b%’：有b就行 ‘%b’：以b结尾

WHERE phone NOT LIKE '%9' quotation mark, percent sign

也可以用REGEXP正则表达式

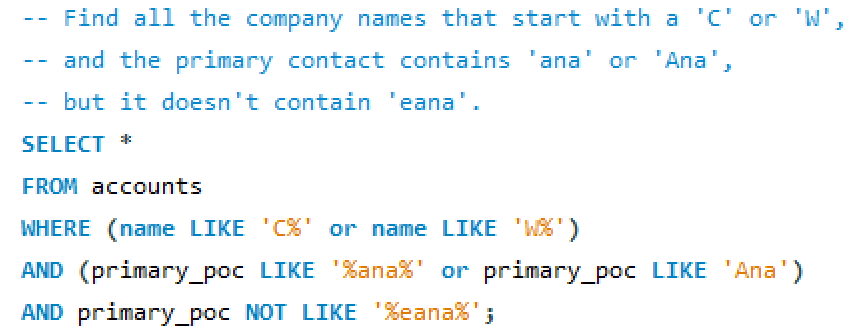
WHERE last\_name REGEXP ‘field’ 此处不需要like，也不需要%

‘**^**filed’：以field开头，

‘filed$’：以filed结尾

当多个条件一起的时候





WHERE address LIKE '%TRAIL%' **OR**

address LIKE '%AVENUE%'

相当于

WHERE address REGEXP ‘TRAIL **|** AVENUE’

也可以一个是开头，另一个包含，另一个结尾，都是随意的

‘**^**TRAIL **|** AVENUE|HH$’

另一个例子

WHERE sales\_rep\_id > 100 AND name NOT IN ('Exxon Mobil','Walmart','Apple')

在x前或者后是y

WHERE last\_name REGXP ‘[gim]e’ 这个表示包含e并且在e前面要么是g要么是i要么是m

其实也相当于

REGEXP ‘ge | ie | me’

‘[a**-**h]e’ 表示从a到h（abcdefgh）

**缺失数据**

WHERE phone IS NULL

WHERE phone IS NOT NULL

## ORDER BY语句

ORDER BY last\_name DESC 从大到小

根据**多个指标**排序

ORDER BY state, last\_name

可以不select，但仍然能根据那列排序

SELECT first\_name, last\_name

FROM customers

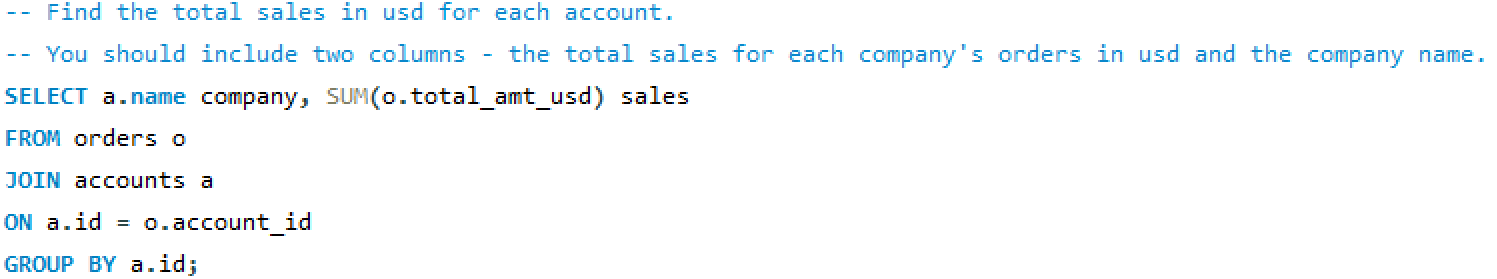
ORDER BY state

**也可以是别名或者math expression**

SELECT 10 AS number

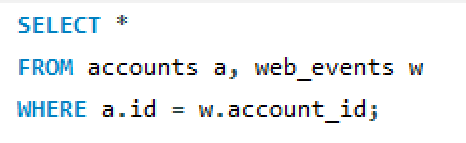
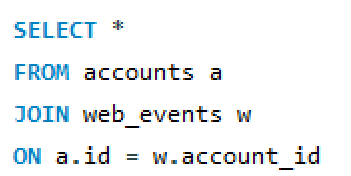
FROM customers

ORDER BY number, quantity\*number

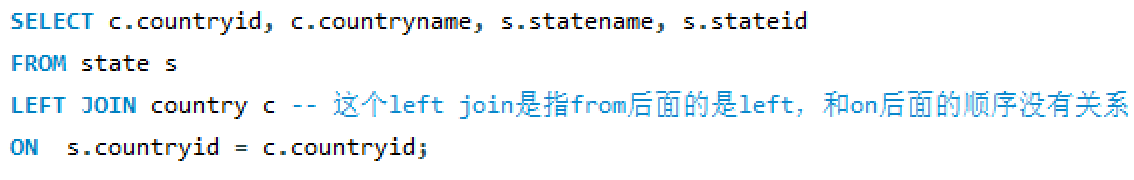


## JOIN 语句

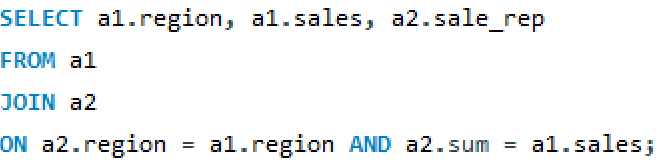
把多个表格里的列合并到一起，并用**Alias**

也可以LEFT JOIN



ON 的有条件可以有**多个**，用AND连接



### Selfjoin 自连接

因为在这个表里，manager他既在employee中，也在report to中。而我们可以看到employee\_id对应的名字，却不能看到report\_to\_id对应的名字。为了找到reprot\_to\_id对应的名字，我们就在employee\_id中搜索对应的id，然后找到名字。

USE sql\_hr;

SELECT e.employee\_id

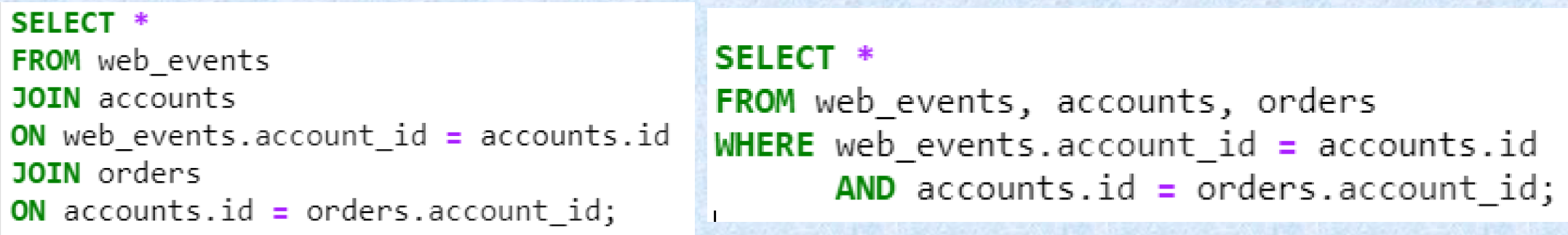
e.employee.name

m.employee.name AS manager\_name 这里改名是因为不改的话就会出现两个name很奇怪

FROM employee e

JOIN employee m

USE e.report\_id = .m.employee\_id



One of the most common use cases for self JOINs is in cases where two events occurred, one after another.

•Find out orders that come within 28 days

SELECT o1.id AS o1\_id,

o1.account\_id AS o1\_account\_id,

o1.occurred\_at AS o1\_occurred\_at,

o2.id AS o2\_id,

o2.account\_id AS o2\_account\_id,

o2.occurred\_at AS o2\_occurred\_at

FROM orders o1

**LEFT JOIN** orders o2

ON o1.account\_id = o2.account\_id

AND o2.occurred\_at > o1.occurred\_at

AND o2.occurred\_at <= o1.occurred\_at + INTERVAL '28 days'

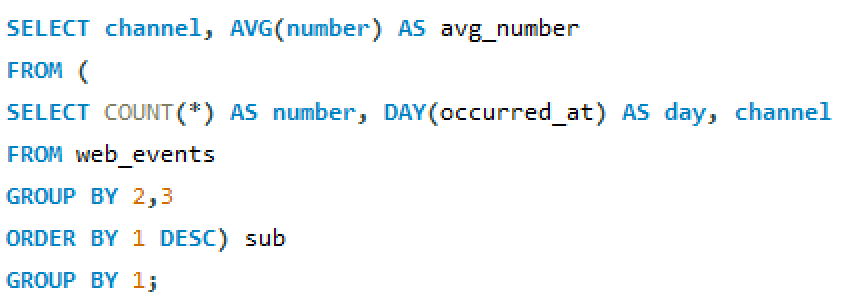
ORDER BY o1.account\_id, o1.occurred\_at

## Subqueries and temporary tables

Find the average number of events for each channel per day.

**每个channel，说明各个channel是分开的。**

**每天平均的event数量，说明要用总的event数量/天数。**



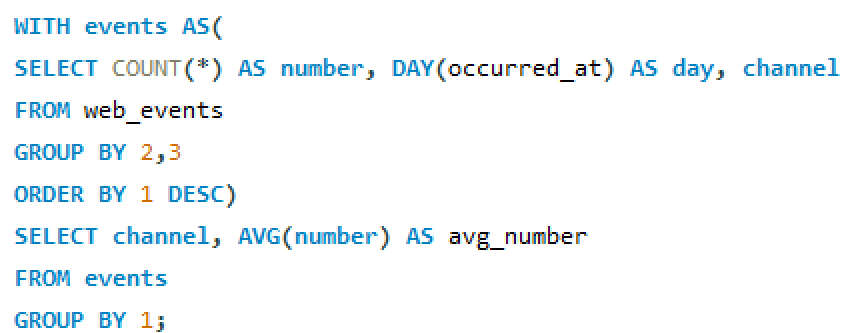
### WITH语句

也叫做common table expression CTE 通用语句表达式

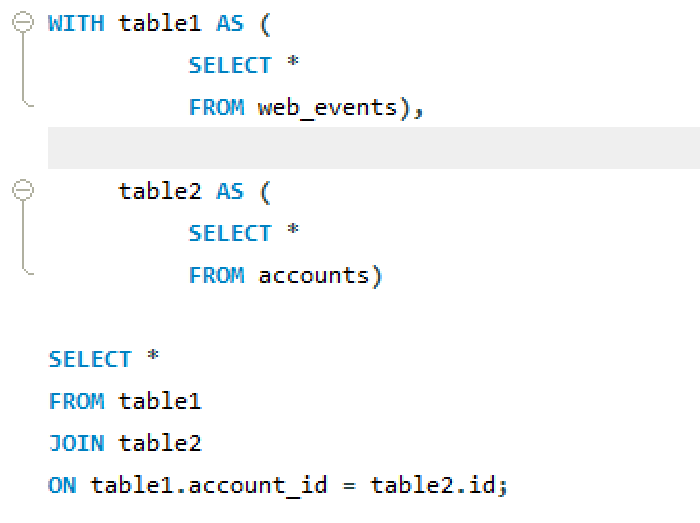
就是先把我们写的一个subquery命名

然后使用命名之后的subquery进行第二次query

其实和上面那个写法是一样的

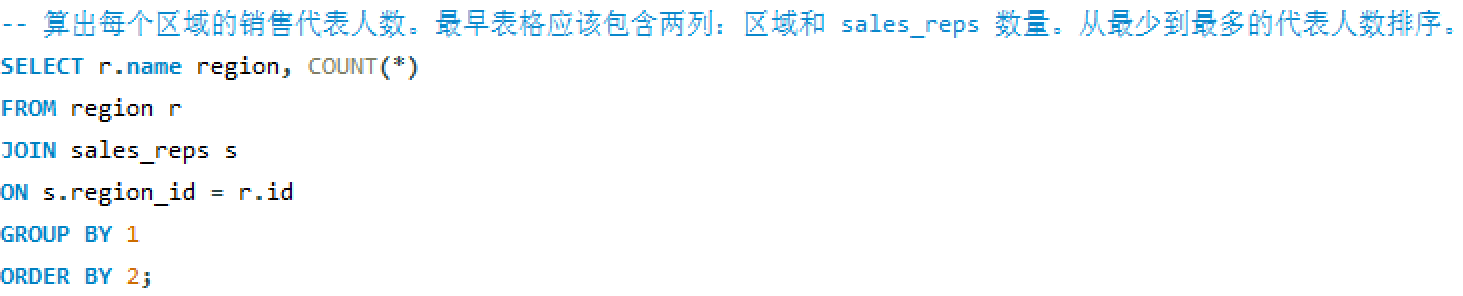


建多个table



错题注意

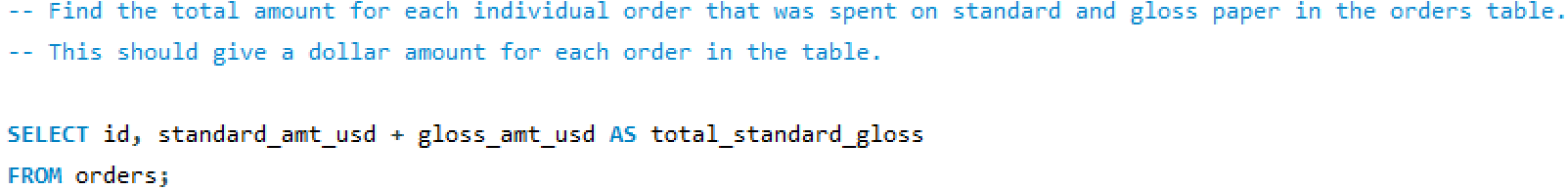
1.由于sale\_reps这个表格里的每一行都代表不同的人，所以只要合并了再数就可以了，不需要写subset



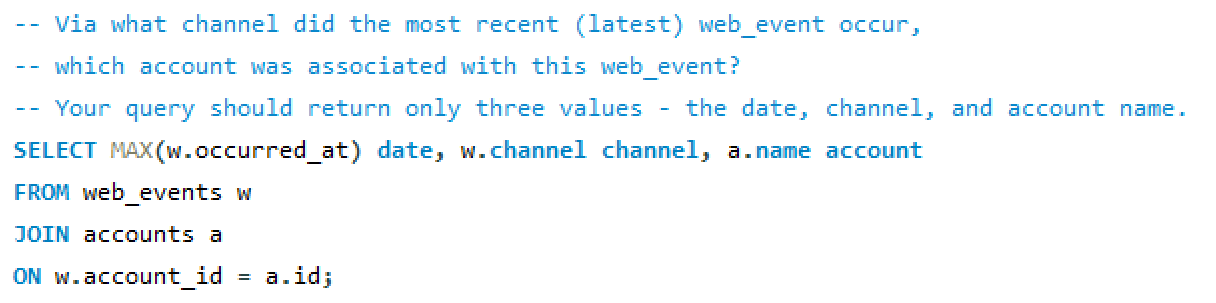
2.sorted from newest to oldest

ORDER BY occurred\_at **DESC**

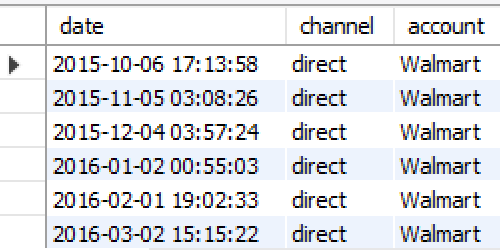
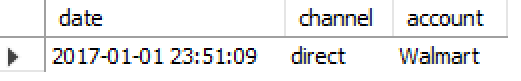
3.Individual order是指每一个order，意思是每一单把standard和gloss的加起来， 而不是用什么SUM()



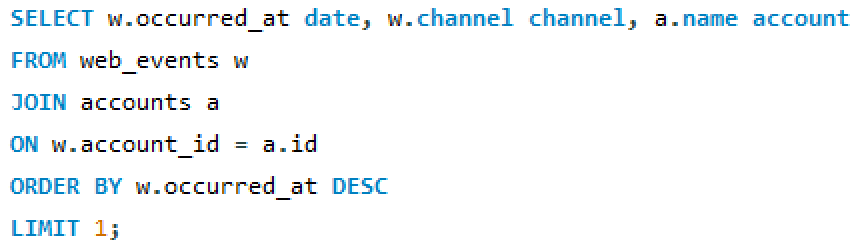
4.求一个变量的最大/最小值，同时要求这个最大/最小变量对应的**其他量**，不可以直接用max这种，因为这样只能选出max的那个变量，而其他变量会是默认的**第一行的值**

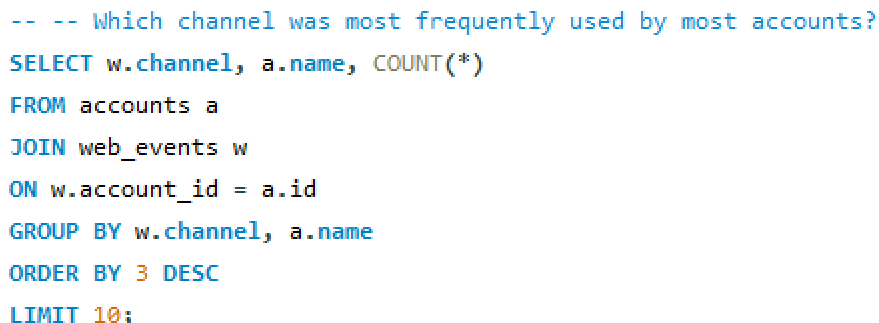


这样求出来是，因为本来这个合并之后的table第一行就是

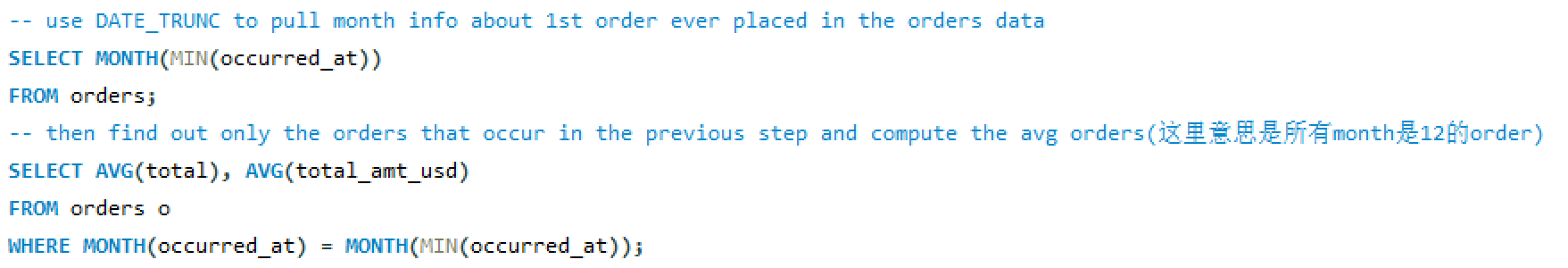


所以只能进行整个的排序

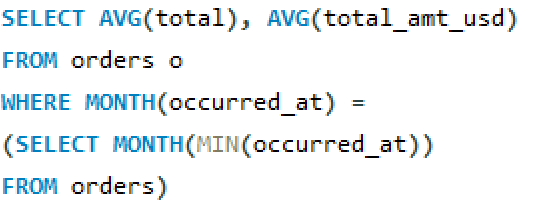




5.MIN,MAX这种aggregation function不能直接放在where里面

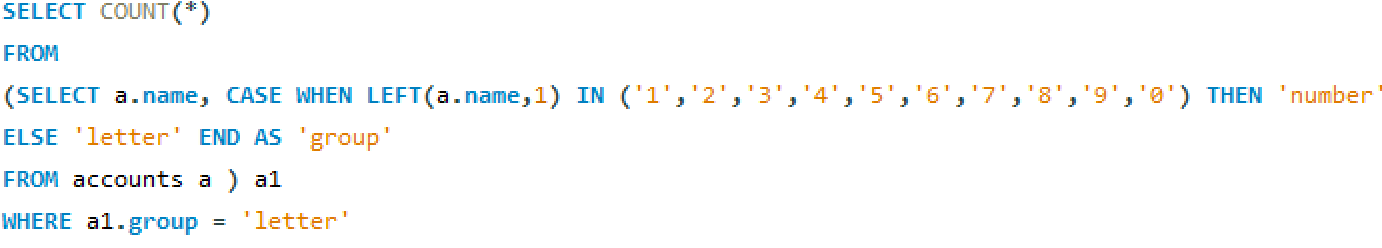


所以应该WHERE中的MONTH(occurred\_at)=query输出的结果 直接用括号把这个输出结果括起来当作一个整体



6. how many total orders是指有多少单，应该count，而不是把total那一列相加

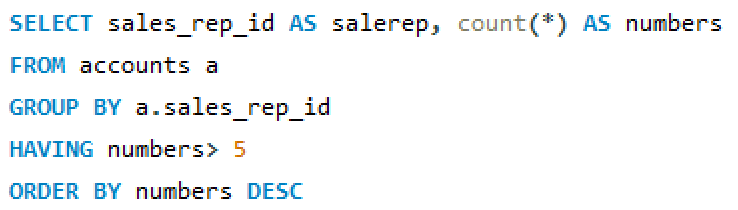
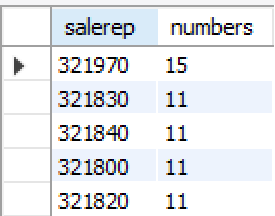
7.使用LEFT,RIGHT的时候注意，即使筛选出的是**数字**，也要加上**双引号**。



8. 查找是否有任何客户与多个区域相关联 数量是否统一/是否是一一对应

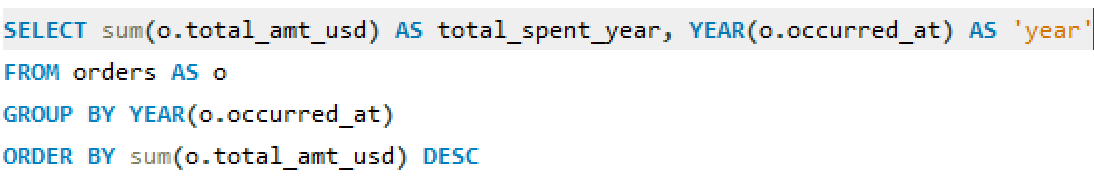
合并之后带有distinct region和company的表 和 distinct

1.How many sale reps have more than 5 accounts they manage?

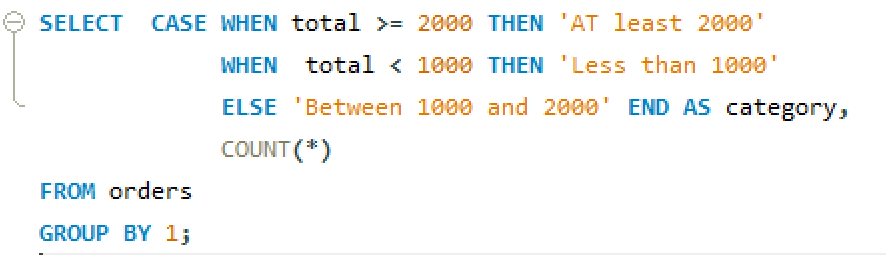
 

2.In which month of which year did Walmart spend the most on glass paper in terms of dollars?

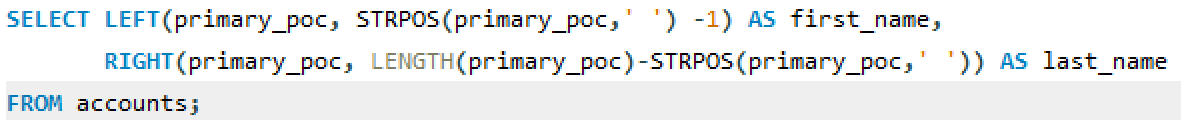
3.Sales in terms of total dollar for all order in each year, ordered from greatest to least.



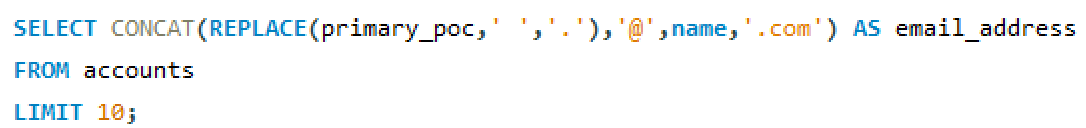
4. Display the **number of orders数有几个order** in each of three categories, base on the **total number of items看每个order里面有多少的东西** in each order. The 3 categories are: ‘At least 2000’, ‘Between 1000 and 2000’, ‘Less than 1000’.



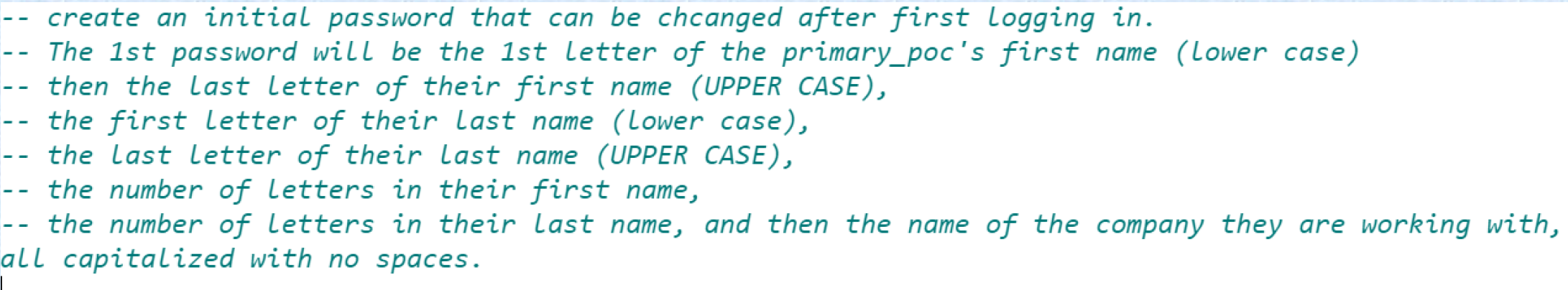
5. Use the account table to create first and last name columns that hold the first and last name for primary\_poc



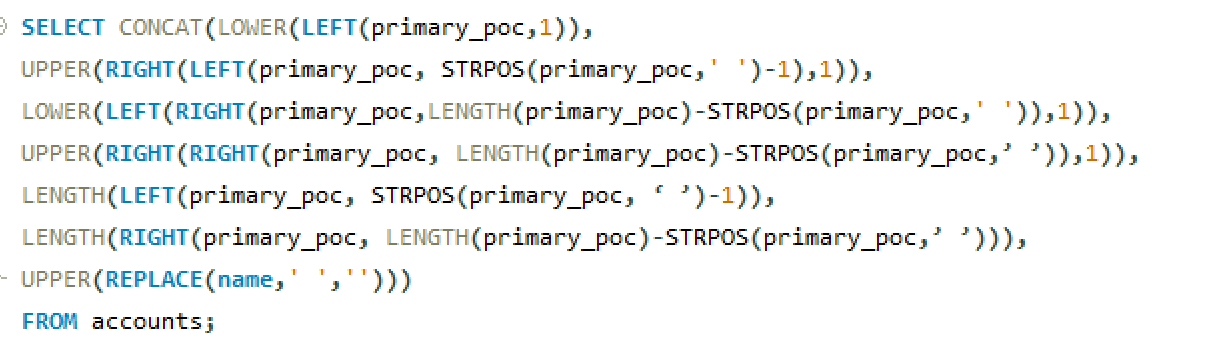
6. Create email address with first name. last [name@**company**.com](mailto:name@company.com) 这里的company是有名字的



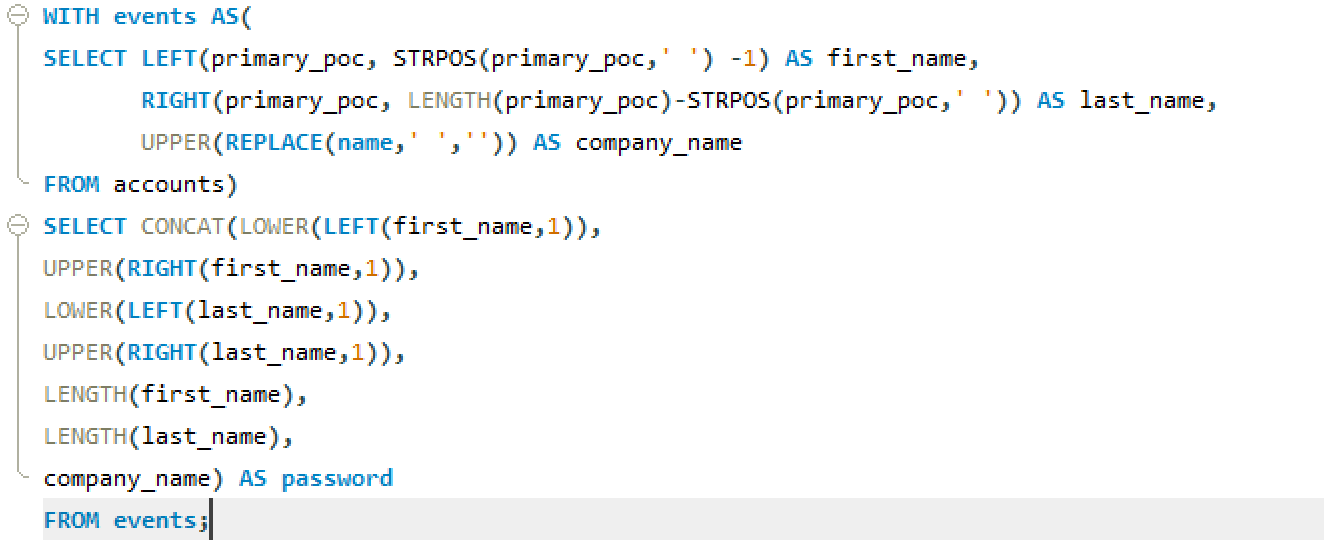
一长串关于名字的复杂的表达式



All capitalized with no space是指要把**所有空格都去掉**，并且要大写



对于这种复杂的，且反复要用到某个求出来的变量的情况，可以用subqueries



一些常用的姓名表达

First name：

LEFT(整个名字, STRPOS(整个名字, ‘ ’)-1)

Last name:

RIGHT(整个名字, LENGTH(整个名字)-STRPOS(整个名字,’ ’))

把first name和last name搞出来

SELECT LEFT(primary\_poc, STRPOS(primary\_poc,' ') -1) AS first\_name,

RIGHT(primary\_poc, LENGTH(primary\_poc)-STRPOS(primary\_poc,' ')) AS last\_name

FROM accounts;

**姓**以xx开头

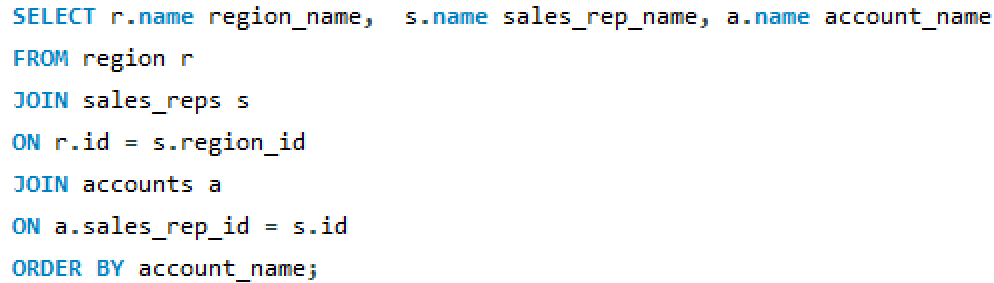
WHERE s.name LIKE '% K%' 注意这里有一个空格！！！表示姓！！

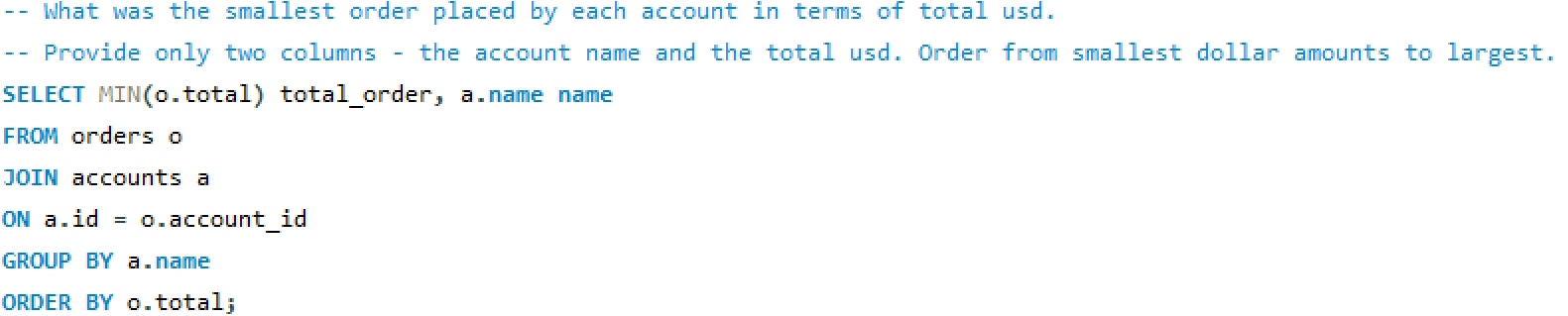
7. -- Provide a table that provides the region for each sales\_rep along with their associated accounts.

-- Your final table should include 3 columns: region name, sales rep name, and account name.

-- Sort the accounts alphabetically (A-Z) according to account name.

像这种题，首先要明确每一列是从哪个table里面来的，比如第一个要求，region for each sales\_rep， 要求region的名字，发现再region里面，所以写r.name





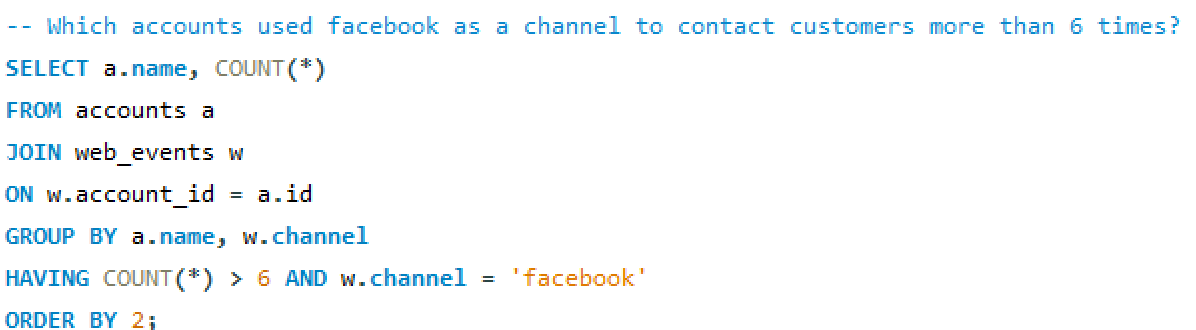
8. determine the average **quantity** of each type of paper they purchased across their orders

determine the average **amount spent** per order on each paper type.

Amount就是金钱，

Quantity才是数量

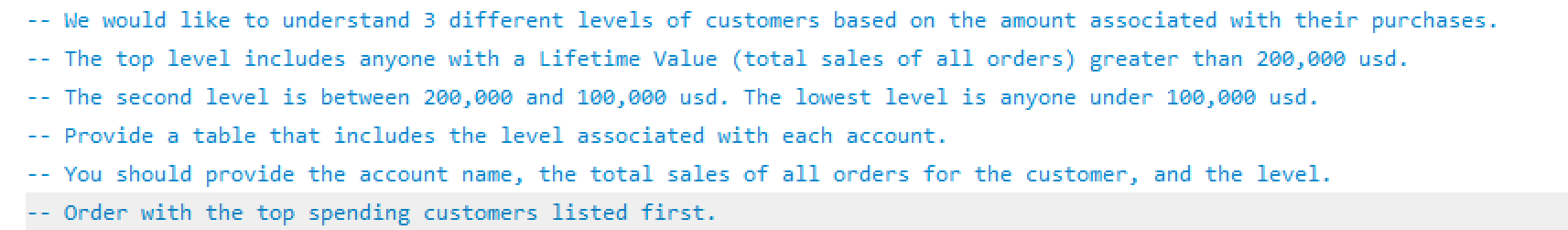
9. 记得有两个筛选条件，用facebook且多于6次，所以一开始要用facebook分组

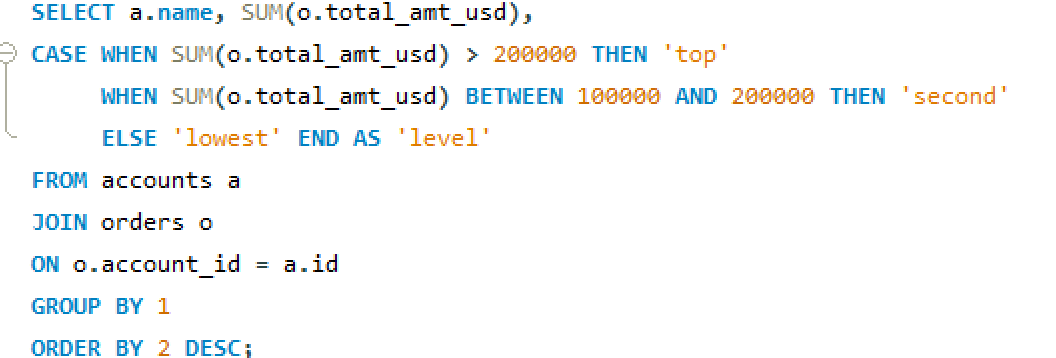


也可以先WHERE筛选再只分组一次

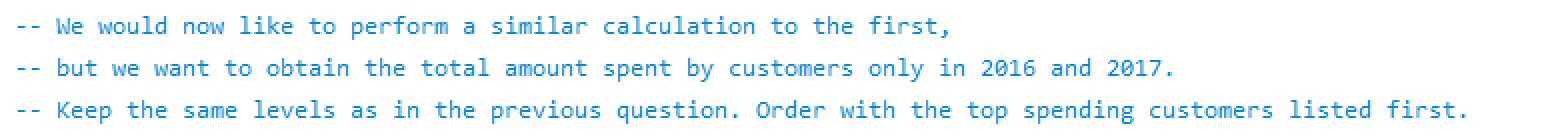


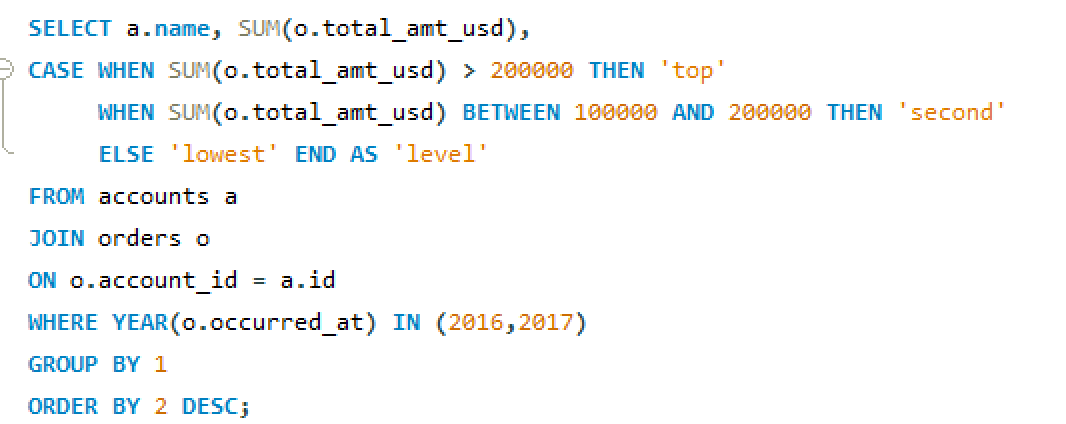
10.



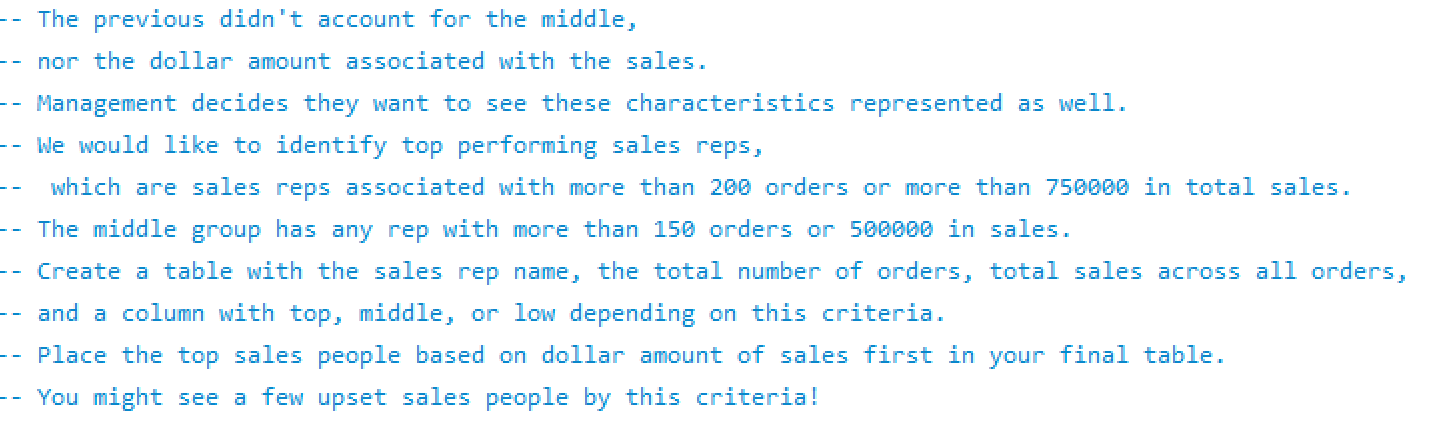


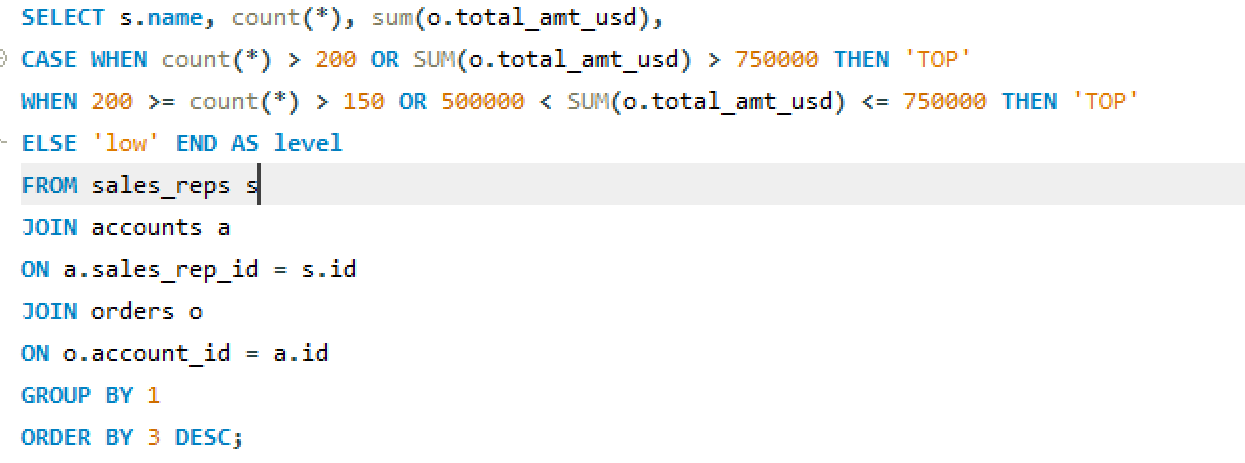
11.





12. 无数个case的条件



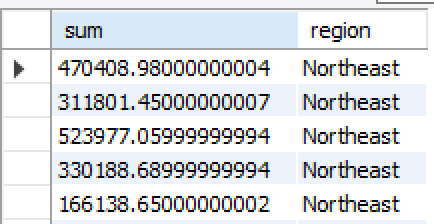
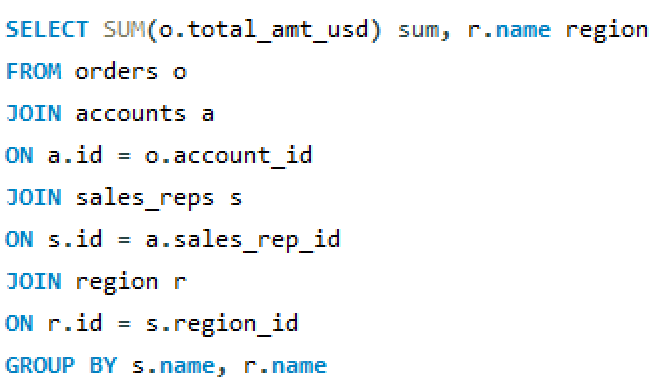


13. 多个subset

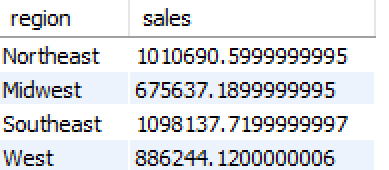
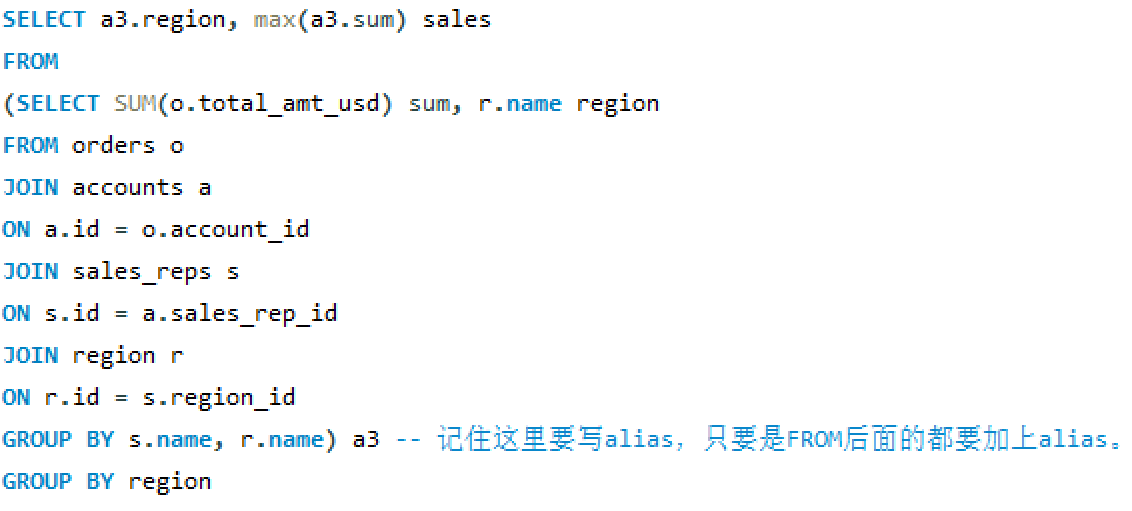
-- Provide the name of the sales\_rep in each region with the largest amount of total\_amt\_usd sales.

第一步，求每个region的最大的total\_amt\_usd是什么（这里没有直接求sales\_rep，因为是分组求最大，所以没办法一次就把name给带进来）

而第一步的第一小步是把所有数据按照region和人name分组。先求每个人在每个区域的total数据，再在这个数据的基础上，筛选出每个区域最大的。

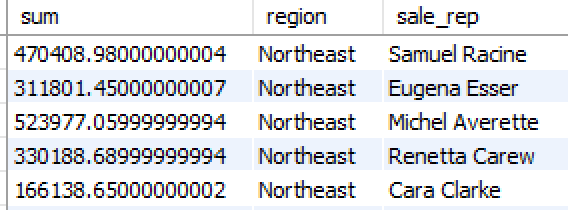
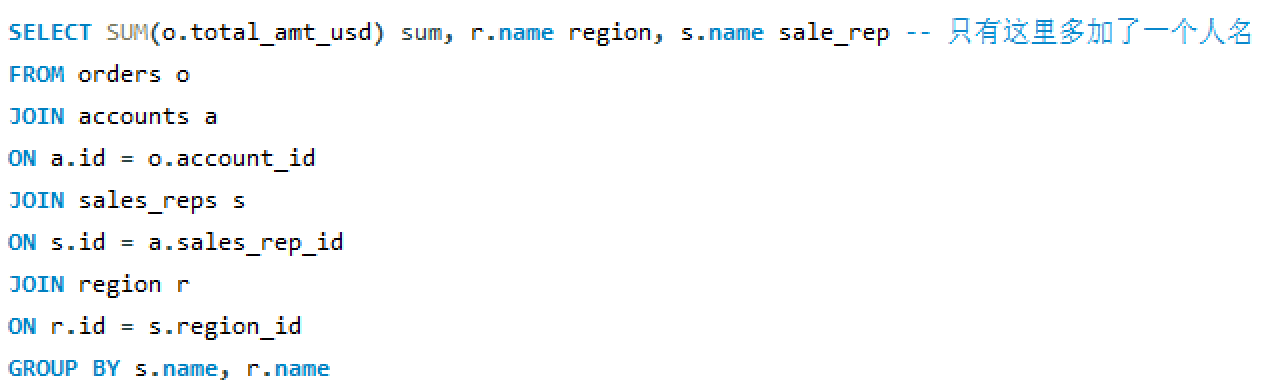


第二小步，筛选每个区域最大的total，不用求出人名

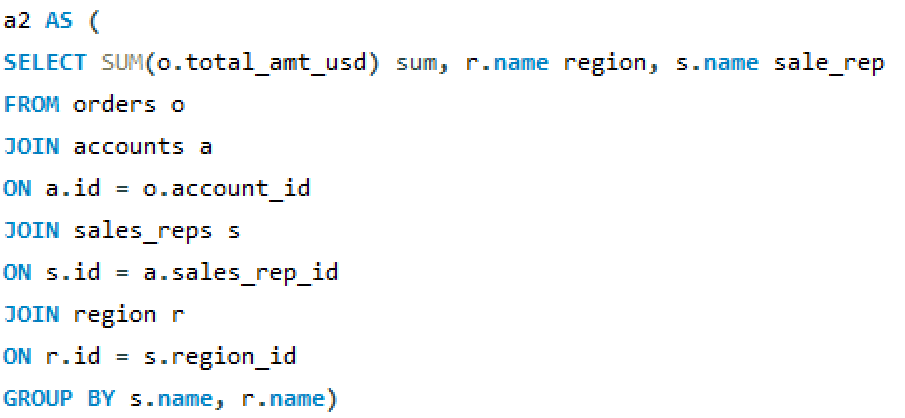
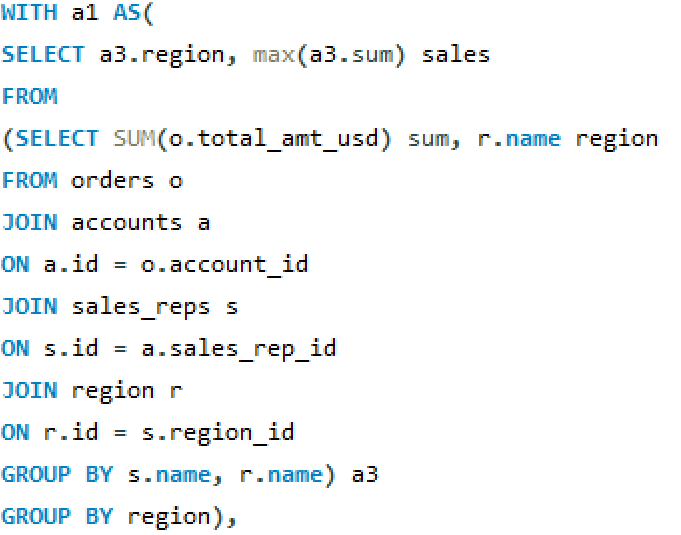


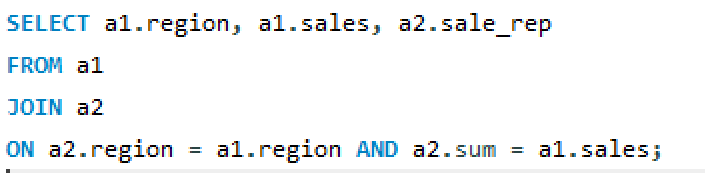
第二步，再建一个table，每个区域每个人的total求出来，把第一个table里面的每一行和第二个table里面的每个值一一对应，从而求出是哪一个人。

每个区域每个人的total其实在第一步里已经写过了，只需要把SELECT的值变一下



第三步，把table1和table2 join起来，提取人名区域和total



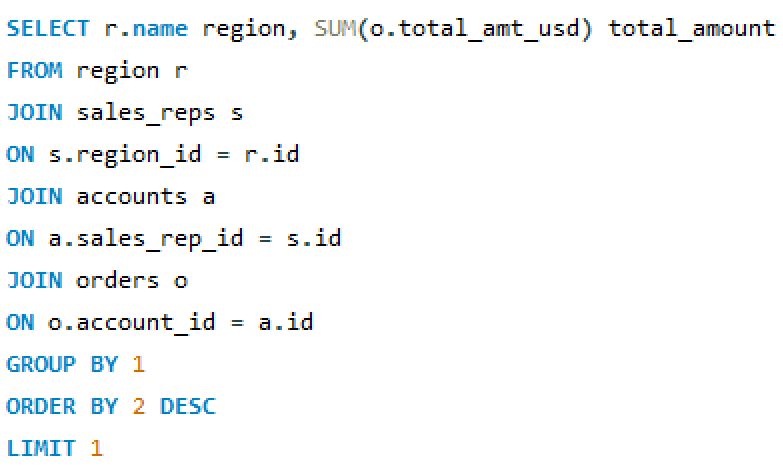
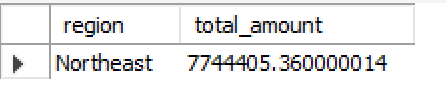


**#多个筛选条件**

注意一下，这里的a3虽然和a1一样，但是必须重新给alias，而不能直接用a1

14. -- For the region with the largest sales total\_amt\_usd, how many total orders were placed?

先求每个region的total\_amt\_usd, 然后得出哪个region最多

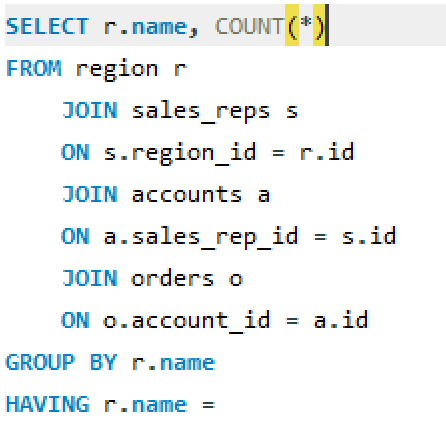
发现是Northeast的最多，最多的有7744405，所以下一步就是求有多少的order

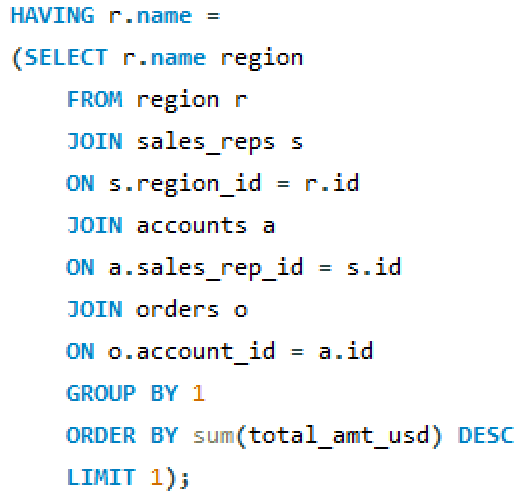
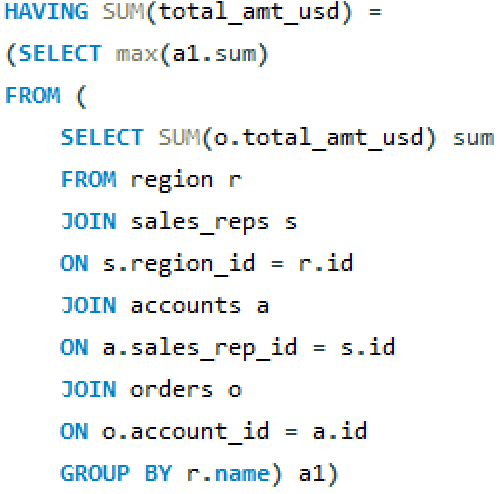
这里也有两种方法。

第一是从northeast出发。 另一种是从sum(total\_amt\_usd)出发

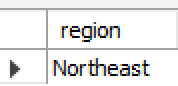
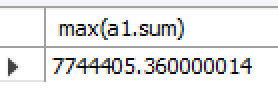
在order这个表里，**筛选**northeast， 在order这个表里算出每个region的sum,

然后求出total order。 看哪个sum和上个表求出的sum相同（麻烦）



下面这个部分返回的是 这个返回的是

15. -- How many accounts had more total purchases than the account name which has bought the most standard\_qty paper

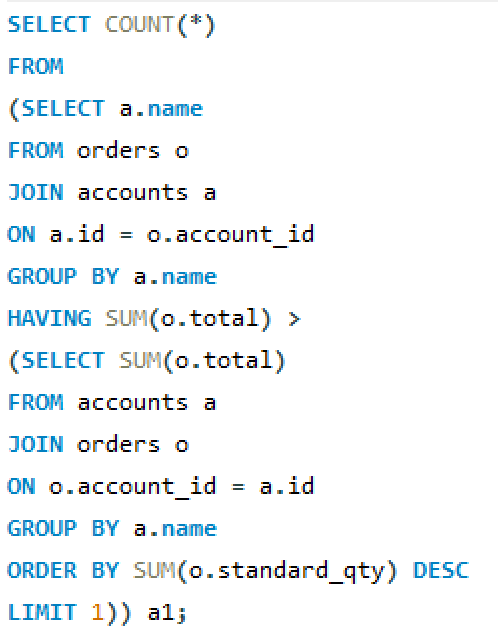
-- throughout their lifetime as a customer?

第一步要求出有最多的standard\_qty paper的account有多少total purchase

之后把这个作为筛选条件，数有多少个满足条件的。

注意1：在第一步中，不需要求出account的名字，也不需要求出最多有多少的standard\_qty，主要求出它的total purchase就可以了

2.第二步中不可以直接count，count会分组count，所以要把第二步求出来的值当成一个新表，再进行count



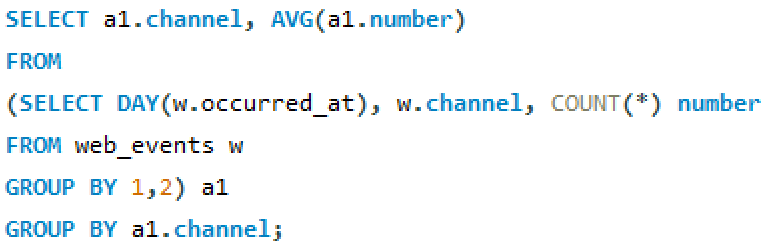
16. -- find the avg for each day and channel

找每个channel每一天平均有多少单

里面的部分是按occurred和channel分组，然后数

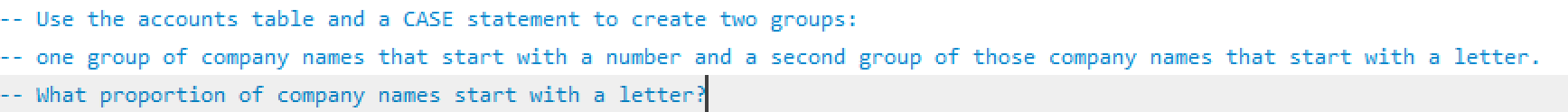
为了求每天有多少个

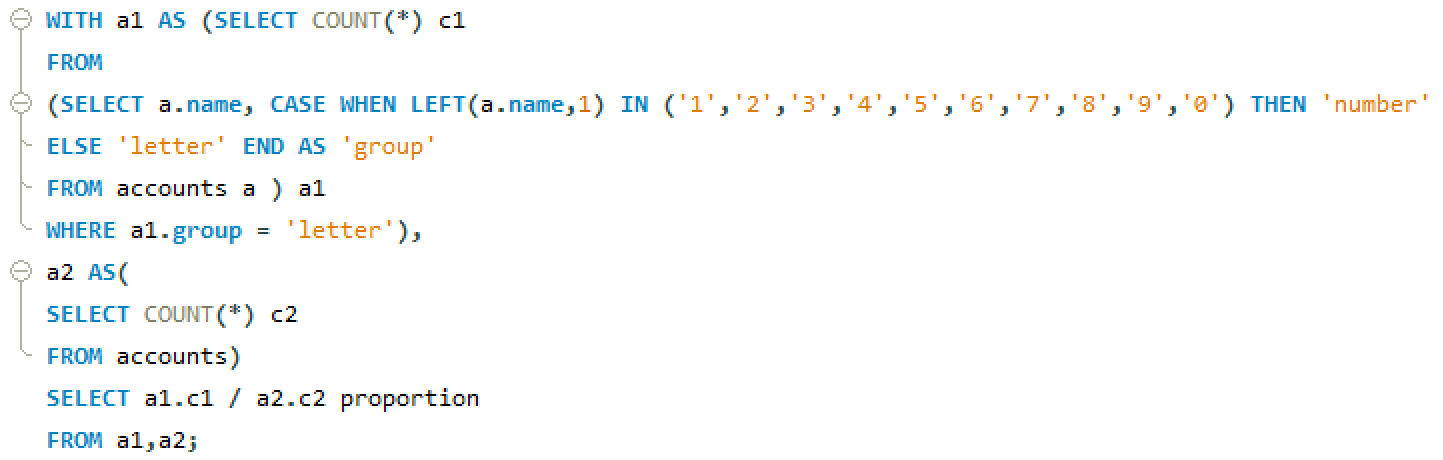
再按channel分组，求平均的



**17.求占比**

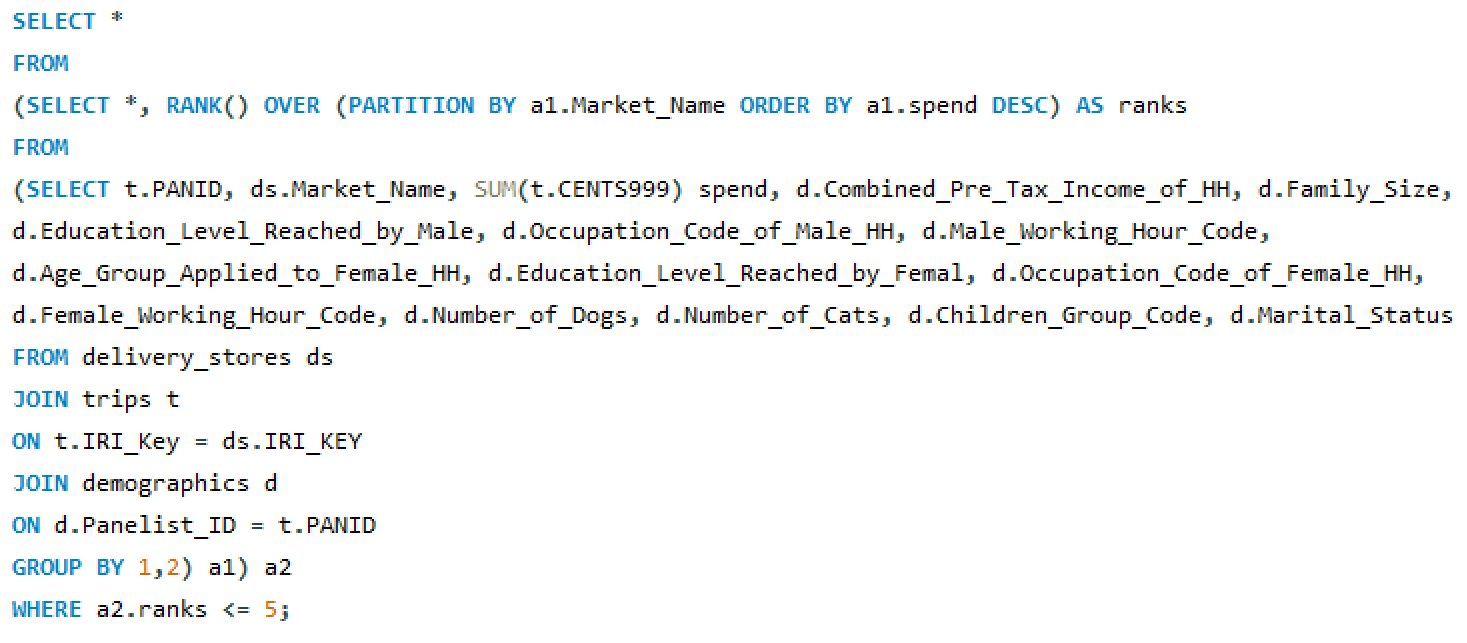
先把分子的数量求出来，再把分母的数量求出来，最后一除就可以

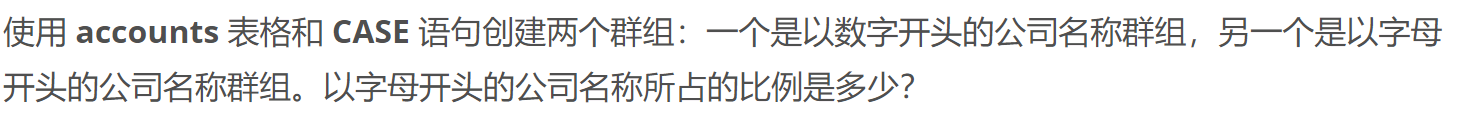




**18.每组取前五个**

**Rank < 5**





**SELECT** **SUM**(**num**) nums, **SUM**(letter) letters

**FROM** (**SELECT** **name**, **CASE** **WHEN** **LEFT**(**UPPER**(**name**), 1) **IN** ('0','1','2','3','4','5','6','7','8','9')

**THEN** 1 **ELSE** 0 **END** **AS** **num**,

**CASE** **WHEN** **LEFT**(**UPPER**(**name**), 1) **IN** ('0','1','2','3','4','5','6','7','8','9')

**THEN** 0 **ELSE** 1 **END** **AS** letter

**FROM** accounts) t1;

分析来在当前买家的总收入于下一时段总收入的差别、

**SELECT** occurred\_at,

total\_amt\_usd,

**LEAD**(total\_amt\_usd) **OVER** (**ORDER** **BY** occurred\_at) **AS** **lead**,

**LEAD**(total\_amt\_usd) **OVER** (**ORDER** **BY** occurred\_at) - total\_amt\_usd **AS** lead\_difference

**FROM** (

**SELECT** occurred\_at,

**SUM**(total\_amt\_usd) **AS** total\_amt\_usd

**FROM** orders

**GROUP** **BY** 1

) sub