The first line is test.

How to find a string within one, return the position.

Original table

|  |
| --- |
| Name |
| This is a test. |

Desired table

|  |
| --- |
| Start position of test |
| 11 |

Using position

# Find one string within one another. The value 0 indicates not found. 11 indicates find the test is start at

How to find a key word in string or not. Return 0 or 1

Original table

|  |
| --- |
| Name |
| This is a test. |

Desired table

|  |
| --- |
| If contain test in the string (1 stands for yes) (0 stands for no) |
| 1 |

Using decode and position, the number of position return is two categories, 0 ->not found, (1~1+) found.

decode is return the exactly match value.

decode (position(‘test’ in name),0,0,1) as flag\_test.

select case when position(‘test’ in name) = 0 then ‘0’

else ‘1’

end

Case when using decode too

decode, the very first beginning of the element is .. the element inside could be odd number or even number.

How to change null to 0, others still others -> iferror

If the column is not null, then some value, otherwise return 0

Original table

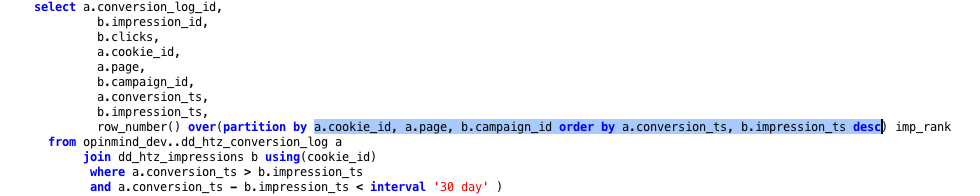
|  |
| --- |
| Click count |
| Null |
| 1 |
| Null |

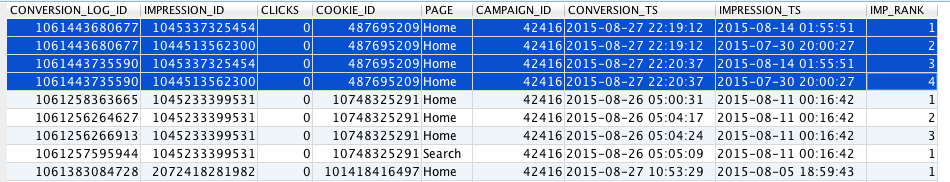
Desired table

|  |
| --- |
| Click count |
| 0 |
| 1 |
| 0 |

nvl2(click\_count, 1, 0)

nvl(click\_count,0)





Countif in excel

Original table

|  |
| --- |
| Activity |
| Home |
| Home |
| Search |
| Book |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| Home | Search | Book |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

original

|  |  |
| --- | --- |
| Activity | num |
| Home | 2 |
| Search | 1 |
| Book | 1 |

desired table

|  |  |  |
| --- | --- | --- |
| Home | Search | Book |
| 2 | 1 | 1 |

Sum(Decode(Activity,’home’,num,0)as Home)as

Decode(Activity,’Search’,num,0)as Home

Decode(Activity,’book,num,0)as Home

sum( decode(Activity, ‘Home’,1,0) as Home,

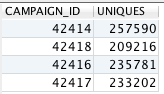
sum( decode(Activity, ‘Search’,1,0) as Search,

sum( decode(Activity, ‘Book’,1,0) as Book,

the operation order of Aggregate function in MySQL is after where and group, like row\_number and sum, count

Join two table without key words, and return all the combination

Original table 1



Original table 2

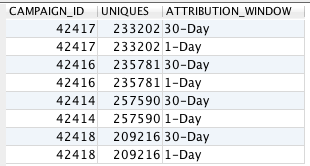


select ‘30-Day’ as attribution\_window

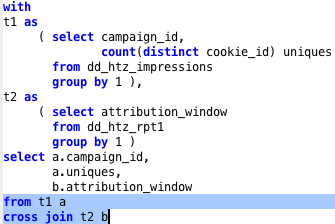
union all

select ‘1-Day’ as attribution\_window

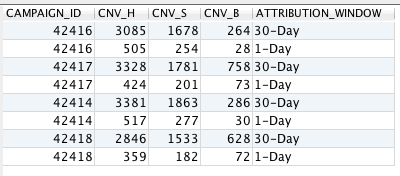
Decided table



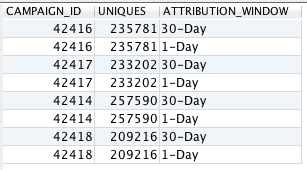
Code



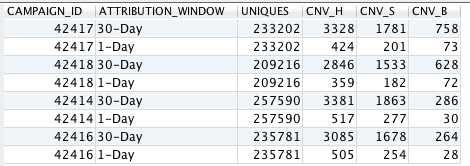
Original table 1



Original table 2



Designed table



code

using/on

SELECT \* FROM t1 **JOIN** t2 **USING** (id, date)

JOIN t2 ON (t2.id = t1.id AND t2.date = t1.date)

Look the brand, if shrik might be where clause, or join,

If different the column, might be use case when

How to use nvl, how to use nvl2

How to avoid the join or left join, within the result of null value

How to use extract

* Extract could extract time between two timestamp but use the epoch from syntax

extract ( epoch from book\_ts – imp\_ts) as slot\_time

NOTE!!! Don’t add () in **(book\_ts – imp\_ts )**

* Extract could condition get the time different using lag

Extract (epoch from imp\_ts – lag (imp\_ts,1) over (partition by cookie\_id order by imp\_ts asc) )as pacing

Note epoch from no () after the calculation, and over partition by , partition by must have () to. so over must followed by ().

Get the list of those customers shopping every consecutive three month

Select distinct (a.customer)

From order a

join order b

on (a. customer = b.customer)

and month (a.order\_ts + 1 Month) = Month(b.order\_ts)

and Year (a.order\_ts + 1 Month) = Year(b.order\_ts)

join order c

on (b. customer = c.customer)

and month (b.order\_ts + 1 Month) = Month(c.order\_ts)

and Year (b.order\_ts + 1 Month) = Year(c.order\_ts)

Use the decode or case when

Nvl2( type, a,b) if type column is not null in that row, return “ a” , else return ‘ b

Nvl( type, b) if type column is not null, then return type column ,else return b

Case when type =”new” then inventory

when type = “used” then inventory

else “0”

end

as new\_cloumn name

decode ( type, ‘new’, inventory,”use”, inventory ,0) as new column name

\* there is no end in the decode

1. Given the following table, how do I pivot using SQL to get the result pivoted by brand, new inventory, and used inventory?

select brand,

sum(decode(type,’new’,inventory,0) )as new,

sum(decode(type,’old’, inventory, 0) )as old

from inventory

group by 1

order by 1

inventory

|  |  |  |  |
| --- | --- | --- | --- |
| **brand** | **locationid** | **type** | **inventory** |
| Toyota | 1113 | new | 100 |
| Chevrolet | 2505 | new | 125 |
| BMW | 9856 | new | 63 |
| Toyota | 3136 | used | 45 |
| Toyota | 48753 | new | 120 |
| BMW | 79285 | used | 31 |
| Honda | 4663 | new | 133 |
| Jeep | 7931 | used | 22 |

Desired Result:

|  |  |  |
| --- | --- | --- |
| **Brand** | **new\_inventory** | **used\_inventory** |
| BMW | 63 | 31 |
| Chevrolet | 125 | 0 |
| Honda | 133 | 0 |
| Jeep | 0 | 22 |
| Toyota | 220 | 45 |

Answer

Select brand,

sum(decode (type = “new” ,inventory,0 ) as new\_inventory

sum(decode (type = “used” ,inventory,0) as old\_inventory

from T1

group by 1

order by 1

call opinmind\_dev..utl\_drop\_table\_if\_exists('wy\_inventory1');

create table opinmind\_dev..wy\_inventory1 as

select 'Toyota' as brand, '1113' as locationid, 'new' as type, 100 as inventory

union all

select 'Chevrolet' as brand, '2505' as locationid, 'new' as type, 125 as inventory

union all

select 'BMW' as brand, '9856' as locationid, 'new' as type, 63 as inventory

union all

select 'Toyota' as brand, '3136' as locationid, 'used' as type, 45 as inventory

union all

select 'Toyota' as brand, '48753' as locationid, 'new' as type, 120 as inventory

union all

select 'BMW' as brand, '79285' as locationid, 'used' as type, 31 as inventory

union all

select 'Honda' as brand, '4663' as locationid, 'new' as type, 133 as inventory

union all

select 'Jeep' as brand, '7931' as locationid, 'used' as type, 22 as inventory

---------

watch the original table first, we could find using join. But you find Toyota has two record as new inventory always remember if you want to de the join ,must keep sure you have the unique join key\_id, otherwise would be wrong!!!, even yourself don’t know where is wrong.

For the first new inventory part, you would find the Toyota has tow rows of the same new inventory. So the first step is the merge or sum by the same brand ,then using Full join.

--------

select \*

from wy\_inventory1

with T1 as

(

select brand, sum(inventory) as new\_inventory

from wy\_inventory1

where type = 'new'

group by 1

),

T2 as

(

select brand, inventory as used\_inventory

from wy\_inventory1

where type = 'used'

)

select brand,nvl(new\_inventory,0) as new\_inventory ,nvl(used\_inventory,0) as new\_inventory

from T1

full join T2

using (brand)

order by 1

Select brand,

Sum(case when type = ‘new’ then inventory

else 0

end) as new\_inventory,

as used\_inventory

sum(case when type = ‘old’ then inventory

else 0

end) as old\_inventory

from inventory

group by 1

select brand,

decode(type,’new’, inventory,0) as

Select brand,

Case when type = ‘new’ then inventory

Else 0

End as new\_inventory,

Case when type = ‘used’ then inventory

Else 0

End as usd\_used\_inventory

From inventory

|  |  |  |
| --- | --- | --- |
| **brand** | **inventory** | **used\_inventory** |
| Toyota | 100 | 0 |
| Chevrolet | 125 | 0 |
| BMW | 63 | 0 |
| Toyota | 0 | 45 |
| Toyota | 120 | 0 |
| BMW | 0 | 31 |
| Honda | 133 | 0 |
| Jeep | 0 | 22 |

Select brand, sum(**inventory),sum(used\_inventory)**

**From inventory**

**Group by 1**

**Order by 1**

Select brand,

Sum( Case

when type = ‘new’ then inventory

else 0

End ) as new\_inventory,

Sum(Case when type = ‘used’ then inventory

Else 0

End )as usd\_used\_inventory

From inventory

Group by 1

Order by 1

Also we can do join

Select brand, sum(inventory) as inventory

From inventory

Where type = ‘new’

Group by 1

Select brand, type as used inventory

From inventory

Where type = ‘used’

|  |  |
| --- | --- |
| **brand** | **inventory** |
| Toyota | 220 |
| Chevrolet | 125 |
| BMW | 63 |
| Honda | 133 |

|  |  |
| --- | --- |
| **brand** | **used\_inventory** |
| Toyota | 45 |
| BMW | 31 |
| Jeep | 22 |

With T1 as

(

Select brand, sum(inventory) as inventory

From inventory

Where type = ‘new’

Group by 1

)

T2 as

(

Select brand, type as used inventory

From inventory

Where type = ‘used’

)

select T1. Brand, nvl(T1.inventory,0), nvl(T2.used\_inventory,0)

from T1

full join T2

on( T1.brand = T2.brand)

as used\_inverntory

SQL:

with T1 as

(

select brand, **inventory as** new\_inventory

from A

where type = ‘new’

)

select b.brand, T1.new\_inverntory,b.inventory as used\_inventory,

from A as b

full join T1

on T1.brand = b.brand

where type = ‘used’

Clickstream

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date\_time | Event\_id | Visitor | Session | Page | Exit Flag |
| Jan 1st 2pm | 123rtfyd | V1 | S1 | Page A | 0 |
| Jan 1st 2:10 pm | 3434jdjsd | V1 | S1 | Page B | 1 |
| Jan 1st 3:00pm | 234343rd | V2 | S2 | Page A | 0 |
| Jan 1st 3:10pm | rereg6677 | V2 | S2 | Page B | 0 |
| Jan 1st 3:20pm | 34fgg | V2 | S2 | Page C | 1 |
| Jan 1st 3:00pm | 234343rd | V3 | S3 | Page A | 1 |

select date\_time::date ,Page, count(distinct Visitor)

Smaple answer:

with T1 as

(

select

date\_time::date,

Visitor,

Page,

row\_number() over ( partition by visitor,page order by date\_time desc) as exit\_flag

from wy\_click1

)

select date\_time::date,

page,

sum(decode(exit\_flag,1,1,0)) as exit\_number

from T1

group by 1,2

\*\* exit\_flag,1,1,0 -> can’t add ‘’ quote, because add quote means varchar, we need do sum so, should leave the ‘’

Select Date\_time::date as date, page, count(distinct Visitor) as num\_visitor

From Clickstream

Group by 1,2

Order by 1,2

Distribute on(date);

|  |  |  |
| --- | --- | --- |
| Date | Page | Num\_visitor |
| Jan 1st | Page A | 3 |
| Jan 1st | Page B | 2 |
| Jan 1st | Page C | 1 |

Select Date\_time::date as date, visitor, page,

Row\_number() over ( partition by visitor, date\_time::date, order by date desc)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date\_time | Visitor | Page |  | Exit Flag |
| ] | V1 | Page A |  | 0 |
| Jan 1st 2:10 pm | V1 | Page B |  | 1 |
| Jan 1st 3:00pm | V2 | Page A |  | 0 |
| Jan 1st 3:10pm | V2 | Page B |  | 0 |
| Jan 1st 3:20pm | V2 | Page C |  | 1 |
| Jan 1st 3:00pm | V3 | Page A |  | 1 |

Total unique Visitors for each page by day:

select Date\_time ::date

,Page,count ( distinct Visitors)

from Clickstream

group by 1,2

create temp table A as (

select visitor, page,

row\_number() over( partition by Visitor asc, order by date\_time desc) as exit Flag

from Clickstream

where row\_number = 1 )

Exit Rate = # of people who exits the site on a page / Total # of people who visited the page

Page B: 1 / 2 = 0.5

Page A: 1 / 3 = 0.3333

select page, count( page), count(visitor)

from Clickstream

group by 1

page count(page) count(vistor)

A 3 3

B 2 2

C 1 1

(count (vistor - count(page) / count(vistor) → exit rate

create the table

call opinmind\_dev..utl\_drop\_table\_if\_exists('wy\_click1');

create table opinmind\_dev..wy\_click1 as

select '2015-10-13 14:00:00' as date\_time, 'V1' as Visitor, 'Page A' as page

union all

select '2015-10-13 14:00:00' as date\_time, 'V4' as Visitor, 'Page A' as page

union all

select '2015-10-13 14:10:00' as date\_time, 'V1' as Visitor, 'Page B' as page

union all

select '2015-10-13 15:00:00' as date\_time, 'V2' as Visitor, 'Page A' as page

union all

select '2015-10-13 15:10:00' as date\_time, 'V2' as Visitor, 'Page B' as page

union all

select '2015-10-13 15:20:00' as date\_time, 'V2' as Visitor, 'Page C' as page

union all

select '2015-10-13 15:00:00' as date\_time, 'V3' as Visitor, 'Page A' as page

union all

select '2015-10-15 14:00:00' as date\_time, 'V1' as Visitor, 'Page A' as page

union all

select '2015-10-15 14:10:00' as date\_time, 'V1' as Visitor, 'Page B' as page

union all

select '2015-10-15 15:00:00' as date\_time, 'V2' as Visitor, 'Page A' as page

union all

select '2015-10-15 15:10:00' as date\_time, 'V2' as Visitor, 'Page B' as page

union all

select '2015-10-15 15:20:00' as date\_time, 'V2' as Visitor, 'Page C' as page

union all

select '2015-10-15 15:00:00' as date\_time, 'V3' as Visitor, 'Page A' as page

order by 1,2

output :

|  |  |  |
| --- | --- | --- |
| DATE\_TIME | VISITOR | PAGE |
| 2015-10-13 14:00:00 | V1 | Page A |
| 2015-10-13 14:00:00 | V4 | Page A |
| 2015-10-13 14:10:00 | V1 | Page B |
| 2015-10-13 15:00:00 | V2 | Page A |
| 2015-10-13 15:00:00 | V3 | Page A |
| 2015-10-13 15:10:00 | V2 | Page B |
| 2015-10-13 15:20:00 | V2 | Page C |
| 2015-10-15 14:00:00 | V1 | Page A |
| 2015-10-15 14:10:00 | V1 | Page B |
| 2015-10-15 15:00:00 | V2 | Page A |
| 2015-10-15 15:00:00 | V3 | Page A |
| 2015-10-15 15:10:00 | V2 | Page B |
| 2015-10-15 15:20:00 | V2 | Page C |

Solution:

Thinking: generate tow tables, first table is how many distinct people to visit each page by date.

The other table is find the last page for each visitor they visit by date.

Sine the row number can be choose where = 1, then using the third table to bridge it.

When generate the third table, sum of the exit\_flag is 1 table, the got the total number of exit number.

Last step is choose the first table join the exit table, calculate the exit\_rate

But must to care about the last page for each visitor visit by day,

First think about how to the row number, last time visit which page, so last time order by desc time. Then partition by is every date and every visior, so it could be every date , we much convert date\_time to date -> partition by date\_time::date, visitor order by date\_time desc) be careful the ::date or convert date it would matter the result.

----------------------------------------------------------------

--- how many distinct unique visitor touch each page everyday

--- Among this page when page is their last activities everyday

----------------------------------------------------------------

with T1 as

( select date\_time::date as date, page, count(distinct visitor) as num\_visitor

from wy\_click1

group by 1,2 ),

T2 as

( select date\_time::date as date, visitor, page,

row\_number() over ( partition by date\_time::date,visitor order by date\_time desc) as exit\_page

from wy\_click1

),

T3 as

(

select date,page, sum(exit\_page) as num\_exit

from T2

where exit\_page = 1

group by 1,2

)

select T1.date, T1.page,T1.num\_visitor,T3.num\_exit, T3.num\_exit/T1.num\_visitor as exit\_rate

from T1

join T3

on (T1.date = T3.date) and (T1.page = T3.page)

Exit rate;

Logic, we must to get the distinct cookie in different page within a day.

Why distinct is some people would reload the page many times. So count as distinct.

If that people go the page B every hour, we still count as one didn’t make sense. We don’t allow the visit within 5 mins

How to write the query?

**Get the epoch with the same cookie\_id**

select cookie\_id, time\_ts,

extract **(**epoch from time\_ts - lag(time\_ts,1)

over (partition by cookie\_id order by time\_ts asc)**) as time\_lag\_seconds**

from opinmind\_dev..wy\_dedupe1

where time\_lag\_seconds >60

|  |  |  |  |
| --- | --- | --- | --- |
| COOKIE\_ID | TIME\_TS | \*\*\* \*\*\* extract content\* | TIME\_LAG\_SECONDS |
| 123 | 2015-10-13 14:01:00 |  | (null) |
| 123 | 2015-10-13 14:02:00 | 2015-10-13 14:01:00 | 60 |
| 123 | 2015-10-13 14:03:00 | 2015-10-13 14:02:00 | 60 |
| 123 | 2015-10-13 14:04:00 | 2015-10-13 14:03:00 | 60 |
| 123 | 2015-10-13 14:05:00 | 2015-10-13 14:04:00 | 60 |
| 123 | 2015-10-13 14:06:00 | 2015-10-13 14:05:00 | 60 |
| 123 | 2015-10-13 14:08:00 | 2015-10-13 14:06:00 | 120 |
| 456 | 2015-10-13 14:01:00 |  | (null) |
| 456 | 2015-10-13 14:07:00 | 2015-10-13 14:01:00 | 360 |
| 456 | 2015-10-13 14:17:00 | 2015-10-13 14:07:00 | 600 |

\*extract follow by () , inside with the calculation time - time, for the seconde time, we write lag(time\_ts,1) , that means move the time *ts down one position as anyother column( fake column) time A column minus time B column. How to move is if the original row cookie*id is the same with next row cookie id then move other wise list null.

\*Move the time\_ts to down 1 postion, if the donw position is same as the original, otherwise is null

**Get the epoch just by time**

select cookie\_id, time\_ts,

extract (epoch from time\_ts - lag(time\_ts,1) over (partition by null order by time\_ts asc)) as time\_lag\_seconds

from opinmind\_dev..wy\_dedupe1

|  |  |  |  |
| --- | --- | --- | --- |
| COOKIE\_ID | TIME\_TS | \*\*\* extract content\* | TIME\_LAG\_SECONDS |
| 456 | 2015-10-13 14:01:00 |  | (null) |
| 123 | 2015-10-13 14:01:00 | 2015-10-13 14:01:00 | 0 |
| 123 | 2015-10-13 14:02:00 | 2015-10-13 14:01:00 | 60 |
| 123 | 2015-10-13 14:03:00 | 2015-10-13 14:02:00 | 60 |
| 123 | 2015-10-13 14:04:00 | 2015-10-13 14:03:00 | 60 |
| 123 | 2015-10-13 14:05:00 | 2015-10-13 14:04:00 | 60 |
| 123 | 2015-10-13 14:06:00 | 2015-10-13 14:05:00 | 60 |
| 456 | 2015-10-13 14:07:00 | 2015-10-13 14:06:00 | 60 |
| 123 | 2015-10-13 14:08:00 | 2015-10-13 14:07:00 | 60 |
| 456 | 2015-10-13 14:17:00 | 2015-10-13 14:08:00 | 540 |

Other 5 mins 1,2,3,4,5,6 get 1 and 6 not each conversion 5 mins

select a.cookie\_id, a.time\_ts, b.time\_ts---,row\_number() over(partition by a.cookie\_id order by b.time\_ts desc) as flg

from opinmind\_dev..wy\_dedupe1 a

join opinmind\_dev..wy\_dedupe1 b

using (cookie\_id)

where b.time\_ts > a.time\_ts

and extract(epoch from b.time\_ts - a.time\_ts) >= 300

work dedupe impression

----- base on the impression

----- find the first conversion

impression left join conversion

where c.conversion\_ts > i.impression\_ts

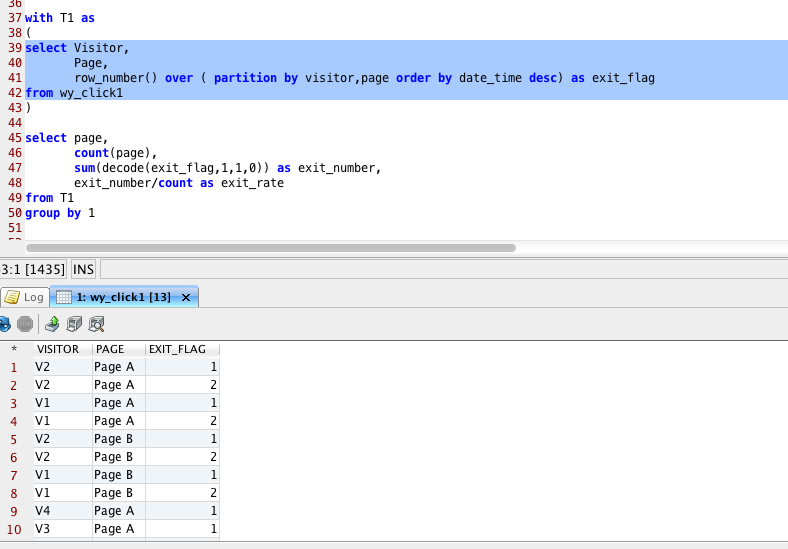
and extract(epoch from conversion\_ts - impression\_ts) <= 86400

row\_number() over(partition by T1.impression\_id order by **T1.conversion\_ts asc**) imp\_rank

----- base on the impression

----- find the cloesest impresssion

a.row\_number() over(partition by a.conversion\_log\_id order by **a.impression\_ts desc**) imp\_rank



\* notice the parentheses

extract ( epoch from time\_ts – lag(time\_ts,1)

over (partition by cookie\_id order by date asc) ) as time\_slot

row\_number()

over( partition by Visitor asc, order by date\_time desc) as exit Flag

how to check the column type

SELECT \*

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE

TABLE\_NAME = 'WY\_DEDUPE1' ;

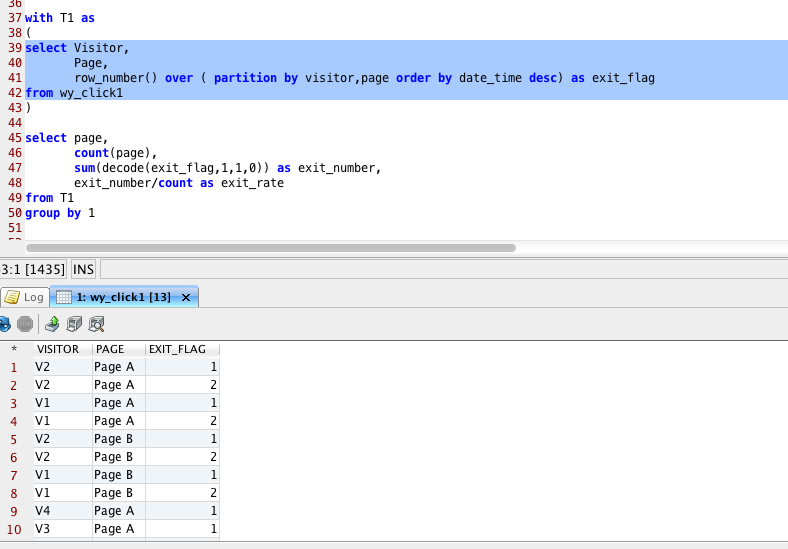
Note: use the upper case the name WY\_DEDUPE1

How to extract to calculate

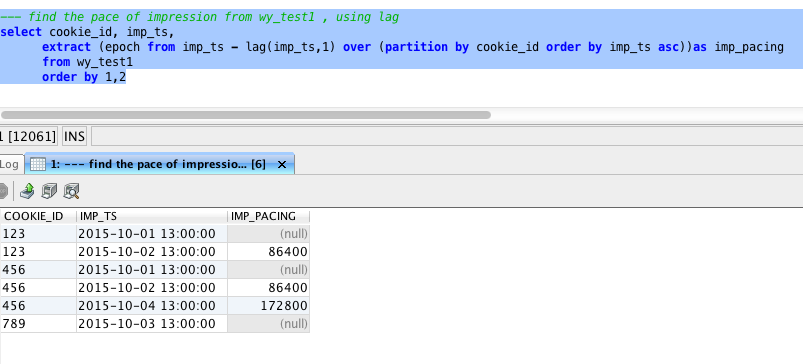
Row\_number

Lag

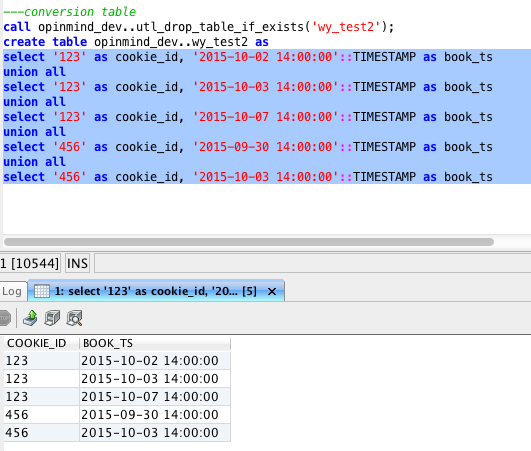
Impression Table



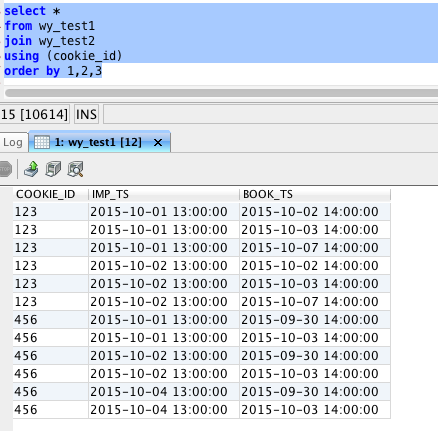
find the time of impression pacing



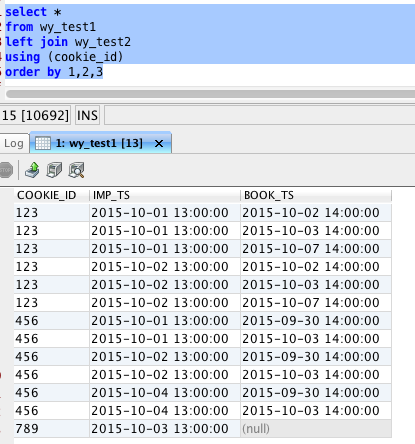
Conversion Table



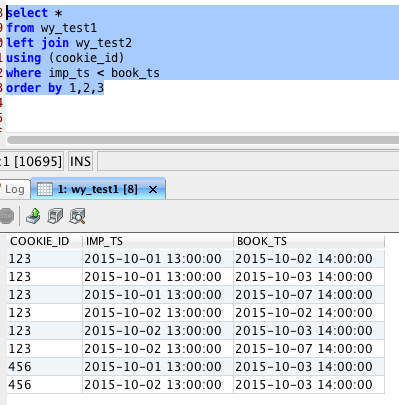
Join table



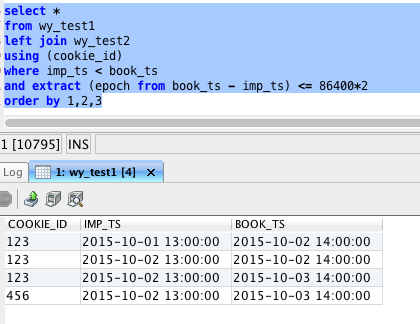
left join



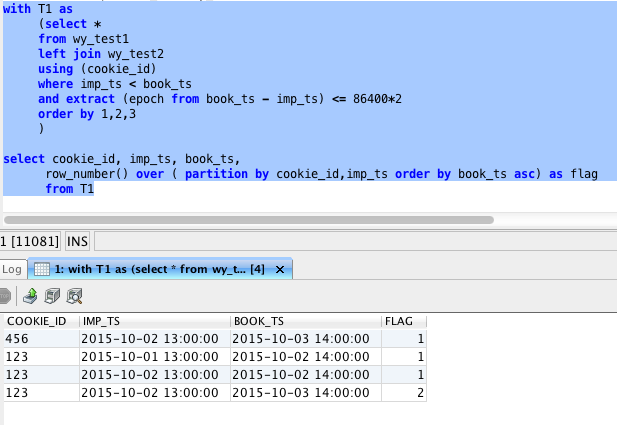
Left join with condition where



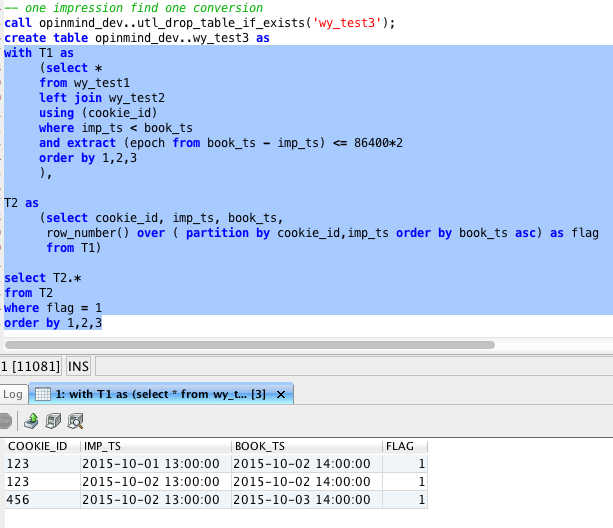
left join with TWO condition



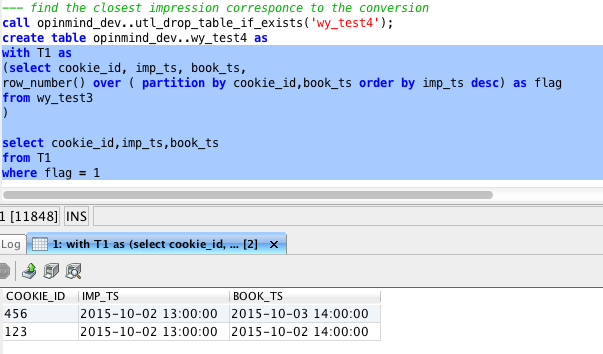
find the first conversion to each impression

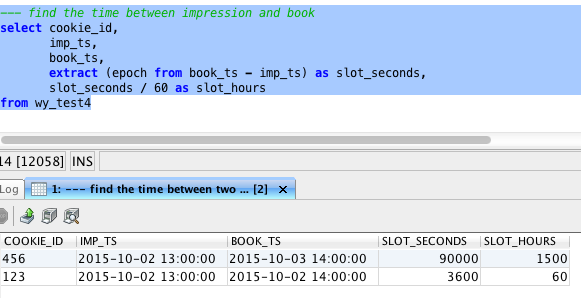


got the flag = 1

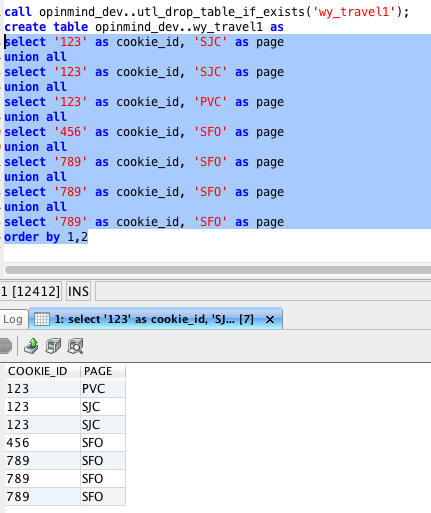


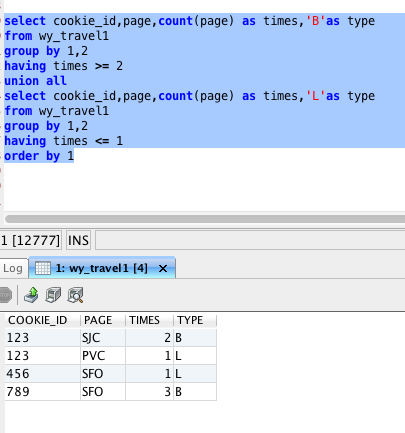
find the closest the impression corresponded to the first conversion





how to count times





call opinmind\_dev..utl\_drop\_table\_if\_exists('wy\_inventory1');

create table opinmind\_dev..wy\_inventory1 as

select 'Toyota' as brand, '1113' as locationid, 'new' as type, 100 as inventory

union all

select 'Chevrolet' as brand, '2505' as locationid, 'new' as type, 125 as inventory

union all

select 'BMW' as brand, '9856' as locationid, 'new' as type, 63 as inventory

union all

select 'Toyota' as brand, '3136' as locationid, 'used' as type, 45 as inventory

union all

select 'Toyota' as brand, '48753' as locationid, 'new' as type, 120 as inventory

union all

select 'BMW' as brand, '79285' as locationid, 'used' as type, 31 as inventory

union all

select 'Honda' as brand, '4663' as locationid, 'new' as type, 133 as inventory

union all

select 'Jeep' as brand, '7931' as locationid, 'used' as type, 22 as inventory

select \*

from wy\_inventory1

------------------------------------------------------------------------

---using decode

---select colume as new name

---if no new value in the cell, it would return 'null' value

---sum to merget the same brand

------------------------------------------------------------------------

select brand, sum(decode(type,'new',inventory,0)) as new\_inventory,sum(decode(type,'used',inventory,0)) as used\_inventory

from wy\_inventory1

group by 1

order by 1

--------------------------------------------------------------------

--using case when

--syntax of case when end by the new name

---if you want to do other calculate no need end follow by others

--------------------------------------------------------------------

--case when type = ' ' then...

-- when type = ' ' then...

-- else 0

-- end as inventory

------

select brand,

sum(case when type = 'new' then inventory

else 0

end)as new\_inventory,

sum(case when type = 'used' then inventory

else 0

end) as used\_inventory

from wy\_inventory1

group by 1

-----------------------------------------------------------------------------

--using join

--when join , make sure key\_id is unique

--full join, left join,join

--is table has '0' using nvl or decode or case when to tackle the null value

-----------------------------------------------------------------------------

with T1 as

(

select brand, sum(inventory) as new\_inventory

from wy\_inventory1

where type = 'new'

group by 1

),

T2 as

(

select brand, inventory as used\_inventory

from wy\_inventory1

where type = 'used'

)

select brand,nvl(new\_inventory,0) as new\_inventory ,nvl(used\_inventory,0) as new\_inventory

from T1

full join T2

using (brand)

order by 1

Given the following table about user activities:

content\_id   user\_id   time   activity   container\_id  content

123                xxx      xxx   post       NULL         xxxxx

234                xxx      xxx     share   NULL         xxxxxx

256               xxx      xxx     picture   NULL         xxxxxx

354               xxx      xxx     comment   123     xxxxxx

574               xxx      xxx     comment   234     xxxxxx

join ? left join? On?

Where

Null!!!

Answer 1

select t1.content\_id, t1.activity, nvl(count(t2.container\_id),0) as num\_comment

from activities T1

left join activities T2

on t1.content\_id = t2.container\_id

where t1.container\_id is NULL ## where t1.activity != ‘comment’

group by 1,2

There are four possible types of activities. Three of those are actually creating a new entry: post (writing a post), share (share an article), and picture (post a picture).

The fourth one is comment, which can be for each of the first three activities (comment for a post, a share, or a picture).

a. How do you return the number of comments for each piece of content?

b. Based on a, how do you calculate the distribution of number of comments for each activity type:

e.g. the result should look like

#comments           num\_activity     type

     1                          32245         post

     1                          24522         share

     1                         28763         picture

    2                         2134            post

    2                         1231            share

    2                          341             picture

Answer2:

select #comment, type, count(content\_id)

from t3

group by 1,2

red book

with T1 as

(

select cookie\_id, impression\_ts, conversion\_ts

from imp

left join con

using (cookie\_id)

where con\_ts > imp\_ts

and extract (epoch con\_ts – imp\_ts) >= 86400\*2

),

T2 as

(

select cookie\_id, impression\_ts, conversion\_ts,

row\_number() over (partition by cookie\_id, conversion\_ts order by impression\_ts desc ) as flag

from T1

)

T3 as

(

select T2.\*

from T2

where flag = 1

) ,

T4 as

(

select cookie\_id, impression\_ts, conversion\_ts,

row\_number() over (partition by cookie\_id, impression\_ts order by conversion\_ts asc) as flag

from T3

),

T5 as

(

select T4.\*

from T4

where flag = 1

)

Self join using rank

|  |  |
| --- | --- |
| Name | Score |
| Wenyi | 30 |
| Zhang | 40 |
| Li | 40 |
| Xing | 50 |

Desired table

|  |  |
| --- | --- |
| Name | Rank |
| Xing | 1 |
| Zhang | 2 |
| Li | 2 |
| Wenyi | 4 |

FROM T1 JOIN T2

|  |  |  |  |
| --- | --- | --- | --- |
| T1.Name | T1.Score | T2.Name | T2.Score |
| Wenyi | 30 | Wenyi | 30 |
| Wenyi | 30 | Zhang | 40 |
| Wenyi | 30 | Li | 40 |
| Wenyi | 30 | Xi | 50 |
| Zhang | 40 | Wenyi | 30 |
| Zhang | 40 | Zhang | 40 |
| Zhang | 40 | Li | 40 |
| Zhang | 40 | Xi | 50 |
| Li | 40 | Wenyi | 30 |
| Li | 40 | Zhang | 40 |
| Li | 40 | Li | 40 |
| Li | 40 | Xi | 50 |
| Xi | 50 | Wenyi | 30 |
| Xi | 50 | Zhang | 40 |
| Xi | 50 | Li | 40 |
| Xi | 50 | Xi | 50 |

On

(T2.Socre > T1.Socre)

or (t1.Name = t2.name and t1.score = t2.score)

|  |  |  |  |
| --- | --- | --- | --- |
| T1.Name | T1.Score | T2.Name | T2.Score |
| Wenyi | 30 | Wenyi | 30 |
| Wenyi | 30 | Zhang | 40 |
| Wenyi | 30 | Li | 40 |
| Wenyi | 30 | Xi | 50 |
| Zhang | 40 | Wenyi | 30 |
| Zhang | 40 | Zhang | 40 |
| Zhang | 40 | Li | 40 |
| Zhang | 40 | Xi | 50 |
| Li | 40 | Wenyi | 30 |
| Li | 40 | Zhang | 40 |
| Li | 40 | Li | 40 |
| Li | 40 | Xi | 50 |
| Xi | 50 | Wenyi | 30 |
| Xi | 50 | Zhang | 40 |
| Xi | 50 | Li | 40 |
| Xi | 50 | Xi | 50 |

Select T1.name, T1.score, count(\*) as rank

From T1

Join T2

On

(T2.Socre > T1.Socre)

or (t1.Name = t2.name and t1.score = t2.score)

group by 1,2

order by rand desc