SQL exercise

1. Campaigns , spend and exchange rate

Table1: campaigns

Account_id	campaign_id
1	123
2	234
3	235

Table2 :spend

Campaign_id	date	spend amount	currency
123	'2017-08-01'	200	USD

Table 3:

Exchange rate

Currency	rate (to USD)
CAD	0.79
USD	1.00

```
id | cid
----+-----
  1 | 123
  1 | 234
  2 | 130
  2 | 140
```

```
dbname=> select * from spend;
 id  |        date         | amount | currency
-----+---------------------+--------+----------
 123 | 2017-08-02 00:00:00 |    200 | USD
 234 | 2017-08-02 00:00:00 |    300 | USD
 234 | 2017-08-03 00:00:00 |    100 | USD
 234 | 2017-08-03 00:00:00 |    100 | CAD
 123 | 2017-08-02 00:00:00 |    200 | CAD
 123 | 2017-08-02 00:00:00 |    200 | JAP
 130 | 2017-08-01 00:00:00 |    200 | USD
 140 | 2017-08-01 00:00:00 |    100 | USD

dbname=> select * from rate;
 currency | rate
----------+------
 USD      |    1
 CAD      | 0.79
```

Q1:
campaign id |Total spend in USD

select
s.id, sum(s.amount* c.rate) as total_spend

From spend s left join currency c on s.currency=c.currency
group by 1

result:

```
id  |   total_spend
-----+-----------------
 123 | 358.000004291534
 140 |              100
 234 | 479.000002145767
 130 |              200
```

follow up: total spent on each user daily

```
dbname=> select c.id, sum(s.amount*r.rate) as total
from campaigns c, spend s, rate r where c.cid=s.id and
s.currency=r.currency

group by 1;
```

```
 id |      total
----+-----------------
  2 |              300
  1 | 837.000006437302
```

Q3: given a daily login table showing when users logged in each day, figure out the number of customers that logged in two days in a row

login_table
date | user_id

```
select
        count(distinct log1.user_id)

from login l1 ,login l2  where l1.user_id=l2.user_id and l2.date=l1.date+1
```

2.
table name
article_views

date | viewer_id| article_id| author_id

Q1: how many articles authors have never viewed their own article?

```
select
  count(author_id)
from article_views
where author_id not in (select distinct(v2.author_id) from article_views v1, article_views v2
where v1. viewer_id=v2.author_id)
```

Q2: how many members viewed more than one article on 2017-08-01?

```
select count(viewer_id)

from
(select viewer_id, count(article_id)
from …
where date=’ 2017-08-01’
group by 1
having count(article_id)>1) as t
```

3. world
table world

continent |country | population
Q1: find the country with largest population in each continent

continent | country | population sort population in descending order. consider corner case that two country has the same largest population in the same continent

```
with temp as (select *, rank() over (partition by continent order by population desc) as rank)
select continent, country, population
```

from temp
where rank=1

or without using window function

with maxp as (select continent, max(population) from world group by1)
select w.continent, w.country, w.population
from world w join maxp m on w.continent=m.continent and w.population=m.population

Q2: for each continent, find the country with largest % population in a given continent
result should look like

continent| country | % of population in that continent

with temp as (select continent, sum(population), max(population) as max from world)

select w.continent, w.country, t.max/t.sum as percentage

from world w , temp t where w.continent=t.continent and w.population=t.max

4. from Microsoft to google

table
member_id| company_name | year_start

```
member_id | company_name | year_start
----------+--------------+------------
        1 | Google       |       1990
        1 | Microsoft    |       2000
        2 | Microsoft    |       2000
        2 | Google       |       2001
        3 | Microsoft    |       1997
        3 | Google       |       1998
        4 | Microsoft    |       1997
        4 | LinkedIn     |       1998
        4 | Google       |       2000
```

Q1:count members who ever moved from MS to google

```
select count(c2.member_id)
```

```
from companies c1, companies c2 where c1.member_d=c2.member_id and
c1.company_name='Google' and c2.company_name='Mic…' and
c1.year_start>c2.year_start
```

Q2: count members who directly moved from ms to google
row_number should do the trick

```
dbname=> select *, row_number() over (partition by member_id order by
year_start) from companies;
 member_id | company_name | year_start | row_number
-----------+--------------+------------+------------
         1 | Google       |       1990 |          1
         1 | Microsoft    |       2000 |          2
         2 | Microsoft    |       2000 |          1
         2 | Google       |       2001 |          2
         3 | Microsoft    |       1997 |          1
         3 | Google       |       1998 |          2
         4 | Microsoft    |       1997 |          1
         4 | LinkedIn     |       1998 |          2
         4 | Google       |       2000 |          3
```

consecutive row_number

5. table orders
customer | product | amount
1 A x1
2 B y2

output
customer | product.A| product.B| product.C
1 x1

2 y2

one customer might have more than one order of the same product?

created a table like:

```
dbname=> select * from orders;
 id | product | amount
----+---------+--------
  1 | a       |      1
  1 | b       |      2
  1 | c       |      3
```

```
  2 | c       |      4
  2 | b       |      3
  2 | a       |      2
```

```
select id,
sum(case when product='a' then amount else 0 end) as 'product_a',
sum(case when product='b' then amount else 0 end) as 'product_b',
sum(case when product='c' then amount else 0 end) as 'product_c'
from orders
group by 1

fairly easy using python instead
orders.pivot_table(amount, ['customer'],'produtc').reset_index()
```

6. sales
table
product_id | quantity

product table
product_id | name

output name, quantity

```
select
    p.name, sum(s.quantity)
from  sales s join product p on s.id=p.id

group by 1
```

7.

```
dbname=> select * from products;
        date         | qty_a | qty_b | qty_c
---------------------+-------+-------+-------
 2013-01-01 00:00:00 |    10 |    20 |    30
 2013-01-02 00:00:00 |    10 |     0 |     1
 2013-01-03 00:00:00 |     0 |     2 |    10
```

```
Q1: table shows cumulative quantity per day
now reformat to

date  | product_name | quantity
```

use union!

```
select date, ‘A’ as product_name, qty_a as quantity from products
union
select date, ‘B’ as product_name, qty_b as quantity from products
union
select date, ‘C’ as product_name, qty_c as quantity from products
```

8. calculate running totals/avaergaes

```
sales_id | sales_cust_id | sales_emp_id | sales_date | sales_total | payment_method
---------+---------------+--------------+------------+-------------+---------------
       63 |             3 |         1115 | 2017-10-01 |        84.3 | Card
        3 |             3 |         1115 | 2017-10-03 |        84.3 | Card
       43 |             3 |         1115 | 2017-10-12 |        84.3 | Card
       83 |             3 |         1115 | 2017-10-21 |        9.99 | Card
       12 |             3 |         1115 | 2017-10-21 |       100.2 | Card
       90 |             3 |         1115 | 2017-10-21 |         5.3 | Card
       11 |             1 |         1115 | 2017-10-21 |        18.2 | Card
        1 |             1 |         1115 | 2017-10-02 |         5.8 | Card
       55 |             3 |         1127 | 2018-1-1   |   333333335 | Cash
```

```
select sales_cust_id, sales_date,
sum(sales_total) over (partition by sales_cust_id order by sales_date)
as total_sum
from sales2;
```

result:

```
sales_cust_id | sales_date |  total_sum
--------------+------------+--------------
            1 | 2017-10-02 |          5.8
            1 | 2017-10-21 |           24
            3 | 2017-10-01 |         84.3
            3 | 2017-10-03 |        168.6
            3 | 2017-10-12 |        252.9
            3 | 2017-10-21 |       368.39
            3 | 2017-10-21 |       368.39
            3 | 2017-10-21 |       368.39
            3 | 2018-1-1   | 333333703.39
```

exactly same as following:

```
select sales_cust_id, sales_date, sum(sales_total) over (partition by
sales_cust_id order by sales_date rows between unbounded preceding and
current row) as running_sum from sales2;
```

but you can also do

```
dbname=> select sales_cust_id, sales_date,sales_total,
sum(sales_total) over (partition by sales_cust_id order by sales_date
rows between 1 preceding and current row) as running_sum
from sales2;
```

this gave:

```
sales_cust_id | sales_date | sales_total | running_sum
---------------+------------+-------------+-------------
             1 | 2017-10-02 |         5.8 |         5.8
             1 | 2017-10-21 |        18.2 |          24
             3 | 2017-10-01 |        84.3 |        84.3
             3 | 2017-10-03 |        84.3 |       168.6
             3 | 2017-10-12 |        84.3 |       168.6
             3 | 2017-10-21 |       100.2 |       184.5
             3 | 2017-10-21 |        9.99 |      110.19
             3 | 2017-10-21 |         5.3 |       15.29
             3 | 2018-1-1   |   333333335 | 333333340.3
```

only add two consecutive rows' sales_total

```
select sales_cust_id, sales_date,sales_total,
sum(sales_total) over (partition by sales_cust_id order by sales_date
rows between 1 preceding and 1 following) as running_sum
from sales2;
```

gave sum between one row preivous and one fow following

```
sales_cust_id | sales_date | sales_total | running_sum
---------------+------------+-------------+--------------
             1 | 2017-10-02 |         5.8 |           24
             1 | 2017-10-21 |        18.2 |           24
             3 | 2017-10-01 |        84.3 |        168.6
             3 | 2017-10-03 |        84.3 |        252.9
             3 | 2017-10-12 |        84.3 |        268.8
             3 | 2017-10-21 |       100.2 |       194.49
             3 | 2017-10-21 |        9.99 |       115.49
             3 | 2017-10-21 |         5.3 | 333333350.29
             3 | 2018-1-1   |   333333335 |  333333340.3
```

9.
Update newly registered fb users to an existing table. Add new column called flag indicating if the user Is new or existing. Calculate rate of these newly added users after two weeks

Churn is defined as user not using fb in the most recent 7 days

Table: users columns

id| fname| lname| registered date | last login date

```
Alter table users
        Add column flag char(3) not null;
Update table users
        Set flag='old'    # default alue
        Where registered_date<current_date()
```

Add new users to existing table

Usually id has the default incremental attribute

Churn rate: no. of churn users/no. of new users

With table1 as(select count(*) from users where flag='new' and datediff (week, registered_date, last_login_date)>=2 and datediff(day, last_login_date, current_date()>=7))

With table2 as (

```
Select count(*) as new_users
From users
Where flag='new' and datediff(week, registered_date, last_logindate)>=2
```

)

```
Select table1.churn_users/table2.new_users as churn_rate
From table1 table2
```

Churn users also subset of new users

Datediff(day, date, row_number() over (partition by user order by date))

10. comments

table
name | posts | comments
u1 page1 90
u2 page 2 55
u1 page2 50

Q1: calculate the average comments for users with >=2 posts and each post has comments >=40

if you run:

```
with temp as (select name, count(posts) from comments group by 1
having count(posts)>=2)
select * from temp t, comments c where t.name=c.name and
c.comments>=40;
```

this will give you u4 too, which is incorrect because u4's comments are not both >=40

the right order is filter comments>=40 then having count() >=2
it needs each post has comments>=40

```
dbname=> with temp as(select name from comments where comments>=40
group by 1 having count(distinct posts)>=2)
select t.name,avg(comments)
from comments c join temp t on c.name=t.name
group by 1;
```

result:

```
name  |         avg
------+--------------------
 u1   | 60.0000000000000000
 u2   | 50.0000000000000000
```

11. Comments 2

table content
content_id | content_type (comment/post) | target_id

if it is comment, target id is the userid who posts it
if it is post then target id is null

Q1: what is the distribution of comments

```
select cnt, count(*) as freq
from (
select content_id, count(content_type) as cnt

from comment

where content_type='comment'

group by 1
)
group by cnt
```

Q2: now what if content_type becomes {comment, video, photo, article} what is the comment distribution for content type?

```
select type, cnt, count(cnt) as freq

(select id, type, count(*) as cnt
from comment
group by 1,2) as t
```

group by 1,2

12. create bins/histogram
revenue table

```
id | revenue
---+---------
 1 |      32
 2 |      36
 3 |      42
 4 |      44
 4 |      55
 4 |      57
 4 |      16
 4 |      12
```

want to look at histogram 0–5, 5–10,10–15..

first create new columns to range the revenue

```
dbname=> select revenue, floor(revenue/5.00)*5 as floor,
ceiling(revenue/5.00)*5 as ceiling from revenue
group by 1 order by 1;
```

```
revenue | floor | ceiling
---------+-------+---------
      12 |    10 |      15
      16 |    15 |      20
      32 |    30 |      35
      36 |    35 |      40
      42 |    40 |      45
      44 |    40 |      45
      55 |    55 |      55
      57 |    55 |      60
```

and then group floor and ceiling to count the number of occurrences

```
dbname=> with temp as(select revenue, floor(revenue/5.00)*5 as floor,
ceiling(revenue/5.00)*5 as ceiling from revenue
) select floor, ceiling , count(*) from temp group by 1,2 ;
```

```
floor | ceiling | count
-------+---------+-------
    55 |      60 |     1
    10 |      15 |     1
    35 |      40 |     1
    40 |      45 |     2
    30 |      35 |     1
    55 |      55 |     1
    15 |      20 |     1
```

13. Funnel metrics/ drop rates

completion rate: this number tells you what percentage of users potential customers who enter the funnel also successfully emerge

for example a funnel with 9 steps, to calculate the completion rate

```
select
        count(distinct case when event='step 9' then user_id else null end) /count(distinct
user_id)

from event_logs
```

now to create a funnel drop rate
need to calculate drop rate step by step

```
with temp as(select event, count(distinct id) as count from events
group by 1 order by 2 desc)
select event, count, cast(count as float)/cast(lag(count) over (order
by count desc)as float) as rate
from temp;
```

14. AB test

table1

User_id	test_name	date	ab test version
1	new_funnel	2017_7_1	control(or test)

table 2

Conversion id	user id	conversion date
1	1	2017-12-25

Q1: data validation, check for orphaned users, that is between two tables, how many users fall into the three possible buckets, entered but not converted, entered and converted, and converted but not entered (orphaned users)?

outer join!

```
Select
    Count(
Case when
    e.id is not null and c.id is null then 1 else null end

) as users_entered_but_not_converted
,

Count(
    Case when e.id is not null and c.id is not null then 1 else null end
```

) as users_entered_and_converted
,

Count(

    Case when e.id is null and c.id is not null then 1 else null end
) as users_converted_but_not_entered

From events e full outer join conversion c on e.id=c.id

Q2: what is the distribution of days it takes to get from entry to conversion?

join two tables, get the time difference between conversion and entry and then find the distribution

Select datediff('day', date(c.date), date(e.id)) as days_from_entry_to_conversion, count(*)

From conversion c left join events e on c.id=e.id
Group by 1

Q3: for test and control groups how many users in each bucket?/ have the expected number of versions, test or control?

Select version,
    Count(distinct e.id) as test_entries,
    Count (distinct c.id) as convert

From events e left join conversion c on e.id=c.id
Group by 1

Q3: find cumulative ab test conversion rates over time
Table events
Id| date | version

Table conversion
Id | date_converted

```
create table events(id int, date timestamp,version varchar(5));
insert into events(id, date, version) values (1, '2017-7-1','test') ;
insert into events(id, date, version) values (2, '2017-7-1','con') ;
insert into events(id, date, version) values (3, '2017-7-2','con');
insert into events(id, date, version) values (4, '2017-7-2','test');
insert into events(id, date, version) values (4, '2017-7-3','test');


create table conversion(id, int);
insert into conversion (id) values (1);
insert into conversion (id) values (2);
insert into conversion (id) values (3);


with convertion_overtime as (select
                date, version, count (e.id) as entry, count(c.id) as convert

                from events e left join conversion c on e.id=c.id
                group by 1,2
)
```

```
       date         | version | entry_number | conversion_number
--------------------+---------+--------------+-------------------
 2017-07-03 00:00:00 | test    |            1 |                 0
 2017-07-01 00:00:00 | test    |            1 |                 1
 2017-07-02 00:00:00 | con     |            1 |                 1
 2017-07-02 00:00:00 | test    |            1 |                 0
   2017-07-01 00:00:00 | con     |            1 |                 1
```

then get the cumulative rates

```
Select
date, version,
sum(entry_numer) over (partition by version order by date) as cumulative_entry_counts,
sum(conversion_number) over (partition by version order by date) as convert_sum as
cumulative_conversion_counts
From conversion_overtime
```

Result:

date	version	cumulative_entry	cumulative_conversions
2017-07-01 00:00:00	con	1	1
2017-07-02 00:00:00	con	2	2
2017-07-01 00:00:00	test	1	1
2017-07-02 00:00:00	test	2	1
2017-07-03 00:00:00	test	3	1

15. tracking application page load time
table

date | app duration(response time)

```
create table timings (date timestamp, duration int);
insert into timings(date, duration) values('2015-06-01',2336);
insert into timings(date, duration) values('2015-06-01',1633);
insert into timings(date, duration) values('2015-06-01',1215);
insert into timings(date, duration) values('2015-06-02',2436);
insert into timings(date, duration) values('2015-06-02',1320);
insert into timings(date, duration) values('2015-06-02',295);
```

Q1: bin users and look at different level of users average response time everyday

ntile can help to split the users into different groups

with temp as(select date, duration, ntile(2) over (partition by date order by duration desc) as ntile from timings) select date, temp.ntile, avg(duration) from temp group by 1,2 order by 1,2;

Q2: tracking percentiles over time, get the quantiles min, Q1, median, Q3, max

With temp as (
    Select date, duration, ntile(4) over (order by duration desc) as ntile from …..
)

Select date, ntile, max(duration) as value from temp group by 1,2

16. analyzing long term/short term blog metrics
Table post

Post | created_at(date and time)

Table pageviews

cookie | post| date

Q1: Define a table called visitors, attribute a new reader to the first post they read

Create table visitors as

```
Select * from(
        Select *, row_number() over (partition by cookie order by date) from pageviews

) as ordered_pageviews

Where row_number=1
```

new_visitor table

Visitor | date_joined

Q2: split visitors into short term and long term buckets based on when they read the first post 30 days as threshold and count how many users in each bucket

```
Select date_trunk('week', 'created_date') as week, case when (vistors.created_at-
posts.created_at)< interval '30 ' days then 'shhort_term' else 'long term' end as period,

Count(1) as visitor_count   # this gives the number of visitors in each group in different
#dates
From visitors join posts on visitors.post=posts.post
Group by 1,2
```

Q3: sign up table

Cookie | date

Result should like

Week | period (long/short term) | number of sign ups

```
Select
    Date_trunc('week', visitors_crated) as week,
Case when ( ….differnce) <= interval '30 days' then 'short_term' else 'long_term' end as
period,
 Count(distinct sign_up.cookie) as signups
From visitors join posts using post join signups using cookie
Group by 1,2
```

17. create a histogram in Redshift

table users

User | age |zipcode

Q1:
solution: use case when to group the data and then count , such as

```
Select
        Case
                When age between 0 and 9
                        Then '0-9'
                When age between 10 and 19
                        Then '10-19'
                When…
        Count(1)

From users
Group by 1
```

Q2: histogram of how often the same zipcode appear

! this is to look at the distribution of counts of zipcodes not distribution of zipcode

With temp as (select zipcode, count(*) as count_columns from users where zipcode is not null group by 1)

Select count_columns, count(*) from temp group by 1 order by 1