Что требовалось сделать:

1. Собрал таблицу, в которой для каждого студента хранится его первая дата успешной (со статусом \*success\*) транзакции (из таблицы `payments`).

2. Собрал таблицу, которая хранит в себе все дни 2016 года. Для этого выбрал все даты из таблицы `classes`.

3. Собрал таблицу со списком успешных транзакций из таблицы `payments`. Сделал агрегацию, в которой для каждого клиента на каждую дату ввёл сумму поля `classes` (только для успешных транзакций).

4. Создал СТЕ для всех запросов из первой части:

- Первый запрос `first\_payments`

- Второй запрос `all\_dates`

- Третий запрос `payments\_by\_dates`

5. Создал СТЕ под названием `all\_dates\_by\_user`, в котором объединил `all\_dates` и `first\_payments`, где хранятся все даты жизни студента после того, как произошла его первая транзакция.

6.Создал CTE `classes\_by\_dates`, посчитал в таблице `classes` количество уроков за каждый день для каждого ученика.

7. Нашёл баланс студентов, который сформирован только транзакциями. Для этого я объединил `all\_dates\_by\_user`и `payments\_by\_dates`так, чтобы совпадали даты и `user\_id` . Использовал оконные функции чтобы найти кумулятивную сумму по полю `transaction\_balance\_change` для всех строк до текущей включительно с разбивкой по `user\_id` и сортировкой по `dt`.

8. По аналогии с уже проделанным шагом для оплат создал CTE `classes\_by\_dates\_dates\_cumsum` для хранения кумулятивной суммы количества пройденных уроков.

Для этого объединл таблицы `all\_dates\_by\_user` и `classes\_by\_dates` так, чтобы совпадали даты и `user\_id`. Использовал оконные выражения, чтобы найти кумулятивную сумму по полю `classes` для всех строк до текущей включительно с разбивкой по `user\_id` и сортировкой по `dt`.

9. Создал CTE `balances` с вычисленными балансами каждого студента. Для этого объединил таблицы `payments\_by\_dates\_cumsum` \*\*\*\*и `classes\_by\_dates\_dates\_cumsum` так, чтобы совпадали даты и `user\_id`.

10. Создал визуализацию (линейную диаграмму) итогового результата.

with first\_payments as

(select

user\_id

,min (date\_trunc ('day', transaction\_datetime )) as date1

from skyeng\_DB.payments

where status\_name = 'success'

group by user\_id

order by 1,2

),

all\_dates as

(select

distinct date\_trunc('day', class\_start\_datetime::date) as dt

from skyeng\_DB.classes

where class\_start\_datetime between '2016-01-01 00:00' and '2016-12-31 23:59'

order by dt),

all\_dates\_by\_user as

(select

first\_payments.user\_id

,all\_dates.dt

from first\_payments

right join all\_dates

on first\_payments.date1 <= all\_dates.dt

-- group by all\_dates.dt

),

classes\_by\_dates as

(select

user\_id

,date\_trunc ('day', class\_start\_datetime) as class\_date

,count(id\_class)\*-1 as classes

from skyeng\_db.classes

where class\_status in ('success','failed\_by\_student')

and class\_type <> 'trial'

group by user\_id,class\_date

),

payments\_by\_dates as

(select

user\_id

,date\_trunc ('day', transaction\_datetime) as payment\_date

,sum (classes) as transaction\_balance\_change

from skyeng\_db.payments

where status\_name = 'success'

group by user\_id,payment\_date

order by 1,2),

payments\_by\_dates\_cumsum as

(select

a.user\_id

,a.dt

,b.transaction\_balance\_change

,coalesce(sum(transaction\_balance\_change) over (partition by a.user\_id order by a.dt),0) as transaction\_balance\_change\_cs

from all\_dates\_by\_user a

left join payments\_by\_dates b

on a.user\_id=b.user\_id

and a.dt=b.payment\_date

),

classes\_by\_dates\_cumsum as

(select

a.user\_id

,a.dt

,b.classes

,coalesce(sum(classes) over (partition by a.user\_id order by a.dt),0) as classes\_cs

from all\_dates\_by\_user a

left join classes\_by\_dates b

on a.user\_id=b.user\_id

and a.dt=b.class\_date ),

balances as

(select a.\*

,b.classes

,b.classes\_cs

,a.transaction\_balance\_change\_cs + b.classes\_cs as balance

from payments\_by\_dates\_cumsum a

join classes\_by\_dates\_cumsum b

on a.user\_id = b.user\_id

and a.dt=b.dt)

select

dt

,sum(transaction\_balance\_change) as sum\_transaction\_balance\_change

,sum(transaction\_balance\_change\_cs) as sum\_transaction\_balance\_change\_cs

,sum(classes) as sum\_classes

,sum(classes\_cs) as sum\_classes\_cs

,sum(balance) as sum\_balance

from balances

group by dt

order by dt