/\*Шаг 1. Создадим CTE first\_payments с двумя полями: user\_id и first\_payment\_date (дата первой успешной транзакции).\*/

with first\_payments as (

select

user\_id

,date\_trunc('day', min(transaction\_datetime)) as first\_payment\_date

from

skyeng\_db.payments

where status\_name = 'success' and id\_transaction is not null

group by 1

),

/\*Шаг 2. Соберем таблицу с датами за каждый календарный день 2016 года. Есть разные способы это сделать, но мы воспользуемся тем, который уже знаем.

Выберем все даты из таблицы 'classes', создадим CTE 'all\_dates' с полем 'dt', где будут храниться уникальные даты (без времени) уроков.\*/

all\_dates as (

select

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

from

skyeng\_db.classes

where date\_trunc('year', class\_start\_datetime) = '2016-01-01'

order by dt asc

),

/\*Шаг 3. Узнаем, за какие даты имеет смысл собирать баланс для каждого студента.

Для этого объединим таблицы и создадим CTE all\_dates\_by\_user, где будут храниться все даты жизни студента после того, как произошла его первая транзакция.

В таблице должны быть такие поля: user\_id, dt.\*/

all\_dates\_by\_user as (

select

user\_id

,dt

from first\_payments as fp

inner join all\_dates as ad on fp.first\_payment\_date <= ad.dt

),

/\*Шаг 4. Найдем все изменения балансов, связанные с успешными транзакциями. Выберем все транзакции из таблицы 'payments', сгруппируем их по 'user\_id' и дате транзакции (без времени) и найдем сумму по полю 'classes'.

В результате получим CTE 'payments\_by\_dates' с полями: 'user\_id', 'payment\_date', 'transaction\_balance\_change' (сколько уроков было начислено или списано в этот день).\*/

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' and id\_transaction is not null

group by 1, 2

),

/\*Шаг 5. Найдем баланс студентов, который сформирован только транзакциями. Для этого объединим 'all\_dates\_by\_user' и 'payments\_by\_dates' так, чтобы совпадали даты и 'user\_id'.

Используем оконные выражения (функцию 'sum'), чтобы найти кумулятивную сумму по полю 'transaction\_balance\_change' для всех строк до текущей включительно с разбивкой по 'user\_id' и сортировкой по 'dt'.

В результате получим CTE 'payments\_by\_dates\_cumsum' с полями: 'user\_id', 'dt', 'transaction\_balance\_change' — 'transaction\_balance\_change\_cs' (кумулятивная сумма по 'transaction\_balance\_change').

При подсчете кумулятивной суммы можно заменить пустые значения нулями.\*/

payments\_by\_dates\_cumsum as (

select

adbu.user\_id

,dt

,transaction\_balance\_change

,sum(case when transaction\_balance\_change is null then 0 else transaction\_balance\_change end) over (partition by adbu.user\_id order by dt asc) as transaction\_balance\_change\_cs

from

all\_dates\_by\_user as adbu

left join payments\_by\_dates as pbd on payment\_date = dt and pbd.user\_id = adbu.user\_id

),

/\*Шаг 6. Найдем изменения балансов из-за прохождения уроков.

Создадим CTE 'classes\_by\_dates', посчитав в таблице 'classes' количество уроков за каждый день для каждого ученика.

Нас не интересуют вводные уроки и уроки со статусом, отличным от 'success' и 'failed\_by\_student'.

Получим результат с такими полями: 'user\_id', 'class\_date', 'classes' (количество пройденных в этот день уроков). Причем 'classes' мы умножим на '-1', чтобы отразить, что '-' — это списания с баланса.\*/

classes\_by\_dates as (

select

user\_id

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

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

from

skyeng\_db.classes

where class\_status in ('success', 'failed\_by\_student') and class\_type <> 'trial'

group by 1, 2

),

/\*Шаг 7. По аналогии с уже проделанным шагом для оплат создадим CTE для хранения кумулятивной суммы количества пройденных уроков.

Для этого объединим таблицы 'all\_dates\_by\_user' и 'classes\_by\_dates' так, чтобы совпадали даты и 'user\_id'. Используем оконные выражения (функцию 'sum'),

чтобы найти кумулятивную сумму по полю 'classes' для всех строк до текущей включительно с разбивкой по 'user\_id' и сортировкой по 'dt'.

В результате получим CTE 'classes\_by\_dates\_dates\_cumsum' с полями: 'user\_id', 'dt', 'classes' — 'classes\_cs' (кумулятивная сумма по 'classes').

При подсчете кумулятивной суммы обязательно нужно заменить пустые значения нулями.\*/

classes\_by\_dates\_dates\_cumsum as (

select

adbu.user\_id

,dt

,classes

,sum(case when classes is null then 0 else classes end) over (partition by adbu.user\_id order by dt asc) as classes\_cs

from

all\_dates\_by\_user as adbu

left join classes\_by\_dates as cbd on class\_date = dt and cbd.user\_id = adbu.user\_id

),

/\*Шаг 8. Создадим CTE 'balances' с вычисленными балансами каждого студента.

Для этого объединим таблицы 'payments\_by\_dates\_cumsum' и 'classes\_by\_dates\_dates\_cumsum' так, чтобы совпадали даты и 'user\_id'.

Получим такие поля: 'user\_id', 'dt', 'transaction\_balance\_change', 'transaction\_balance\_change\_cs', 'classes', 'classes\_cs', 'balance' ('classes\_cs' + 'transaction\_balance\_change\_cs').\*/

balances as (

select

pbdc.user\_id

,pbdc.dt

,transaction\_balance\_change

,transaction\_balance\_change\_cs

,classes

,classes\_cs

,classes\_cs + transaction\_balance\_change\_cs as balance

from

payments\_by\_dates\_cumsum as pbdc

inner join classes\_by\_dates\_dates\_cumsum as cbddc on pbdc.dt = cbddc.dt and pbdc.user\_id = cbddc.user\_id

order by pbdc.user\_id asc, pbdc.dt asc

)

/\*select

\*

from balances

order by user\_id asc, dt asc

limit 1000\*/

/\*Шаг 9. Посмотрим, как менялось общее количество уроков на балансах студентов.

Для этого просуммируем поля `transaction\_balance\_change`, `transaction\_balance\_change\_cs`, `classes`, `classes\_cs`, `balance` из CTE `balances` с группировкой и сортировкой по `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 asc