**Задание 1**

Посмотрите на изменения балансов студентов (на примере топ-1000 строк), собранных из CTE.

Какие данные вас смущают? Какие вопросы стоит задавать дата-инженерам и владельцам таблиц?

*- Из 1000 строк 142 строки имеют отрицательный баланс (14,2% - довольно высокий процент).*

Возможно данные в таблицах были некорректно внесены или во время переноса данных произошел сбой?

Ранне (на примере предыдущего 2015 года) было такой высокий процент отрицательных балансов студентов?

*-Обращает на себя внимание также неравномерная оплата студентами занятий: кто-то платит каждые 2-3 месяца, кто-то 2 раза в год.*

Во время ли отражается информация об оплатах студентов? Есть ли задержка с отражением указанной информации и по каким причинам? В прошлых периодах были такие же задержки с отражением поступлением денег? С чем были связаны такие задержки?

**Задание 2**

Создайте визуализацию (линейную диаграмму) итогового результата.

Какие выводы можно сделать из получившейся визуализации?

Комулятивная сумма баланса до сентября месяца равномерно увеличивалась (с 11 на 06.01.16 до 2876 на 01.09.16), c сентября 2016 года рост ее ускорился до значения 4534 по состоянию на 31.12.16, что связано с опережающим ростом поступлений платежей в сентябре-декабре 2016 года по сравнению с ростом списания уроков в соответствующий период.

Количество транзакций подвержено значительным колебаниям в течении всего года, скорее всего можно проследить зависимость резких скачков по количеству оплаченных уроков от рекламных акций и скидок.

-- Соберите таблицу, в которой для каждого студента будет храниться

-- его первая дата успешной (со статусом success) транзакции (из таблицы payments).

with first\_payments as

(

select user\_id

,min (transaction\_datetime::date) as first\_payment\_date

from SKYENG\_DB.payments

where status\_name = 'success'

group by user\_id

order by user\_id

)

,

-- Соберите таблицу, которая будет хранить в себе все дни 2016 года. Для этого необходимо выбрать все даты из

-- таблицы classes.

all\_dates as

(

select distinct class\_start\_datetime::date as dt

from SKYENG\_DB.classes

where date\_part('year', class\_start\_datetime) = 2016

)

,

-- Соберите таблицу со списком успешных транзакций из таблицы payments. Сделайте агрегацию,

-- в которой для каждого клиента на каждую дату необходимо вывести сумму поля classes (только для успешных транзакций).

-- transaction\_balance\_change (сколько уроков было начислено или списано в этот день)

payments\_by\_dates as

(select user\_id

,transaction\_datetime::date 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 user\_id

)

,

-- all\_dates\_by\_user, в котором объедините all\_dates и first\_payments, где будут

-- храниться все даты жизни студента после того, как произошла его первая транзакция.

all\_dates\_by\_user as

(

select first\_payments.user\_id as user\_id

,all\_dates.dt as dt

from first\_payments

left join all\_dates on all\_dates.dt >= first\_payments.first\_payment\_date

)

,

-- Найдите изменения балансов из-за прохождения уроков.

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

-- Вас не должны интересовать вводные уроки и уроки со статусом, отличным от success и failed\_by\_student.

classes\_by\_dates as

(

select user\_id

, class\_start\_datetime::date 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

)

,

-- Найдите баланс студентов, который сформирован только транзакциями. Для этого объедините all\_dates\_by\_user и payments\_by\_dates

-- так, чтобы совпадали даты и user\_id . Используйте оконные функции (в частности, 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.

payments\_by\_dates\_cumsum as

(

select a.user\_id

,a.dt

,b.transaction\_balance\_change

,sum(case when b.transaction\_balance\_change is null then 0

else b.transaction\_balance\_change

end) over (partition by a.user\_id order by a.dt rows between unbounded preceding and current row) as transaction\_balance\_change\_cs

from all\_dates\_by\_user as a

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

)

,

-- создайте CTE classes\_by\_dates\_dates\_cumsum для хранения кумулятивной суммы количества пройденных уроков.

classes\_by\_dates\_cumsum as

(

select a.user\_id

,a.dt

,b.classes

,sum(case when b.classes is null then 0

else b.classes

end) over (partition by a.user\_id order by a.dt rows between unbounded preceding and current row) as classes\_cs

from all\_dates\_by\_user as a

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

)

,

-- Создайте CTE balances с вычисленными балансами каждого студента. Для этого объедините

-- таблицы payments\_by\_dates\_cumsum и classes\_by\_dates\_dates\_cumsum так, чтобы совпадали даты и user\_id.

balances as

(

select a.user\_id

,a.dt

,coalesce(a.transaction\_balance\_change,0) as transaction\_balance\_change

,a.transaction\_balance\_change\_cs

,coalesce(b.classes,0) as classes

,b.classes\_cs

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

from payments\_by\_dates\_cumsum as a

join classes\_by\_dates\_cumsum as b on a.user\_id = b.user\_id and a.dt = b.dt

)

-- Выберите топ-1000 строк из CTE balances с сортировкой по user\_id и dt. Посмотрите на изменения балансов студентов.

-- select \*

-- from balances

-- order by dt, user\_id

-- limit 1000

-- Посмотрите, как менялось общее количество уроков на балансах студентов.

-- Для этого просуммируйте 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