-- ## ОСНОВНОЕ ЗАДАНИЕ КУРСОВОЙ РАБОТЫ

-- Наша задача — смоделировать изменение балансов студентов. Баланс — это количество уроков, которое есть у каждого студента.

-- Чтобы проверить, всё ли в порядке с нашими данными, составьте список гипотез и вопросов. Нам важно понимать:

-- \* - сколько всего уроков было на балансе \*\*всех учеников\*\* за каждый календарный день;

-- \* - как это количество менялось под влиянием транзакций (оплат, начислений, корректирующих списаний) и уроков (списаний с баланса по мере прохождения уроков).

-- Также мы хотим создать таблицу, где будут балансы \*\*каждого студента\*\* за каждый день.

-- ## Результат курсовой

-- В результате должен получиться запрос, который собирает данные о балансах студентов за каждый "прожитый" ими день.

-- Отличным бонусом будет, если вы сделаете визуализации и на ее основании соберете гипотезы.

-- В каком формате сдавать курсовую

-- Соберите единый файл Excel, в котором будут данные о балансах, добавьте туда же ссылку на ваш запрос из Metabase (как вы делали это в домашних работах). На отдельных листах:

-- В ПРОДЛЁНКЕ ПО КУРСОВОЙ ЕСТЬ ИНФОРМАЦИЯ, ЧТО ФАЙЛ EXCEL ДЕЛАТЬ НЕ НУЖНО. ДОСТАТОЧНО METABASE !!!

-- \* - соберите вопросы к дата-инженерам и владельцам таблицы payments (назовите этот лист \*Задание 1\*);

-- \* - соберите выводы из визуализации (линейной диаграммы) получившегося результата (назовите этот лист \*Задание 2\*).

-- Изучим наши БД

-- select \* from skyeng\_db.payments

-- select \* from skyeng\_db.classes

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

WITH -- формируем CTE

-- \*\*\* ШАГ 1

-- Узнаем, когда была первая транзакция для каждого студента. Начиная с этой даты, мы будем собирать его баланс уроков.

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

first\_payments as

(

select user\_id -- ID ученика

, min(transaction\_datetime::date) as first\_payment\_date -- первая (минимальная) дата транзакции для каждого студента

-- date\_trunc ('day', min(transaction\_datetime)) as first\_payment\_date -- можно сделать так

from skyeng\_db.payments

where status\_name = 'success' -- только "успешные"

group by user\_id -- группировка по ID студента

order by user\_id asc -- и сортировка по возрастанию

),

-- \*\*\* 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 -- только уникальные даты

-- distinct (class\_start\_datetime::date) as dt -- можно сделать так

from skyeng\_db.classes

where class\_start\_datetime between '2016-01-01' and '2017-01-01' -- только за 2016 год

-- class\_start\_datetime >= '2016-01-01' and class\_start\_datetime < '2017-01-01' -- можно сделать так

group by dt -- группируем по дате

),

-- \*\*\* 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. Используем JOIN по неравенству.

all\_dates\_by\_user as

(

select user\_id -- ID ученика

, dt -- даты

from all\_dates -- соединяем два предыдущих СТЕ из 1 и 2 шага

join first\_payments

on all\_dates.dt >= first\_payments.first\_payment\_date -- для каждого ученика только уникальные даты начиная с первого платежа

order by user\_id, dt -- можно использовать 1,2

),

-- \*\*\* 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 -- ID ученика

, transaction\_datetime::date as payment\_date -- даты покупки уроков, можно сделать через date\_trunc

, sum (classes) as transaction\_balance\_change -- баланс количества уроков на дату платежа

from skyeng\_db.payments

where status\_name = 'success' -- только 'успешные'

group by user\_id, payment\_date -- группировка можно использовать 1,2

order by user\_id asc -- сортировка учеников по возростанию

),

-- \*\*\* 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 ad.user\_id -- ID ученика

, dt -- даты года

, coalesce(transaction\_balance\_change,0) as transaction\_balance\_change -- изменение баланса. coalesce выставляет 0, если был NULL (пусто)

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

from all\_dates\_by\_user as ad -- таблица 3 ШАГ и

left join payments\_by\_dates as pd -- таблица 4 ШАГ объединим

on ad.user\_id = pd.user\_id -- равенство по ID ученика

and ad.dt = pd.payment\_date -- равенство по дате

),

-- \*\*\* 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 -- ID ученика

, date\_trunc('day', class\_start\_datetime) as class\_date -- дата урока (без времени)

, count(id\_class) \* -1 as classes -- количество проведенных уроков с минусом - , так как это "списание"

from skyeng\_db.classes

where class\_type <> 'trial' -- не берём триальные или через !=

and (class\_status = 'success' or class\_status = 'failed\_by\_student') -- только успешные или завершенные по "вине" ученика, можно сделать через in ('','')

group by 1, 2 -- группируем по ID и дате

),

-- \*\*\* 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 ad.user\_id -- ID ученика

, dt -- даты года

, coalesce(classes,0) as classes -- количество проведенных уроков в день

, sum(coalesce(classes,0)) over (partition by ad.user\_id order by dt) as classes\_cs -- кумулятивная сумма уроков на дату

from all\_dates\_by\_user as ad -- таблица 3 ШАГ и

left join classes\_by\_dates as cd -- таблица 6 ШАГ объединим

on ad.user\_id = cd.user\_id -- равенство по ID ученика

and ad.dt = cd.class\_date -- равенство по дате

),

-- \*\*\* 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 pdc.user\_id -- ученик из 5 шага

, pdc.dt -- дата из 5 шага

, transaction\_balance\_change -- оплаты из 5 шага

, transaction\_balance\_change\_cs -- кумулятивная сумма количества оплат из 5 шага

, classes -- уроки из 7 шага

, classes\_cs -- кумулятивная сумма количества уроков из 7 шага

, classes\_cs + transaction\_balance\_change\_cs as balance -- вычисление баланса на каждый день после оплаты урока

from payments\_by\_dates\_cumsum as pdc -- таблица 5 и

left join classes\_by\_dates\_dates\_cumsum as cddc -- таблица 7 ШАГА объединим

on pdc.user\_id = cddc.user\_id -- равенство по ученику и

and pdc.dt = cddc.dt -- дате

-- using (user\_id,dt) -- условие по равенству можно заменить

)

-- ### Задание 1

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

---------- Какие вопросы стоит задавать дата-инженерам и владельцам таблицы payments?

-- select \*

-- from balances

-- order by

-- -- user\_id, dt

-- balance

-- limit 1000

-- Изучив баланс студентов приходим к выводу, что существуют "отрицательные" балансы, что по сути не должно быть в системе или иметь не высокие значения!

-- При сортировке по balance Особенно заинтересовал студент с ID 197757075 с балансом -59 уроков. Изучим БД "первоисточники":

-- select \* from skyeng\_db.payments

-- where user\_id = 197757075

-- order by transaction\_datetime asc

-- Видим, что в 2016 году практически все операции 6/7 status\_name = failed и почти половина из них 4/7 "Начисление корпоративному клиенту"

-- select \* from skyeng\_db.classes

-- where user\_id = 197757075

-- order by class\_start\_datetime asc

-- В тоже время уроки проходили регулярно и многие успешно.

-- Так же стоит заметить что некоторые уроки имеют "отрицательную" длительность, но отмечены как успешные.

-- Передадим информацию дата-инженерам и владельцам таблицы payments для дальнейшего анализа и корректировки.

----------

-- \*\*\* 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

-- ### Задание 2

-- Создайте визуализацию (линейную диаграмму) итогового результата.

---------- Какие выводы можно сделать из получившейся визуализации?

-- По итогам визуализации общий баланс на 31.12.2016 -> оплачено уроков 21798 количество пройденых уроков 17264 баланс 4534 уроков оплачены. Тенденция положительная.

-- Оплаты - Максимумы оплат в основном приходятся на конец и начало месяцев, что может быть связано с денежными выплатами или маркетинговыми акциями компании. Минимальные значения имеют сезонный (летний) и "праздничный" фактор

-- Прохождение уроков - Заметна "недельная" 7-8 дневная периодичность. Студенты в основном предпочитают активно заниматься в будние дни.