**Курсовая работа “Моделирование изменения балансов студентов”**

Наш курс по SQL подошел к концу. Чтобы структурировать полученные знания, проверить себя и собрать небольшой кейс в портфолио, предлагаем выполнить курсовую работу.

**Для чего это всё?**

**Чтобы проверить себя по следующим пунктам:**

* [ ] Умение писать запросы без логических и синтаксических ошибок.
* [ ] Умение видеть в данных проблемы и говорить о них.
* [ ] Умение писать код, который выводит правильный результат в том виде, в котором его запросил заказчик.
* [ ] Умение писать читаемый код, который может легко понять коллега, руководитель или ревьюер.

**Отличным бонусом будет, если вы:**

* [ ] сделаете визуализации и на их 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:**

* + соберите вопросы к дата-инженерам и владельцам таблиц (назовите этот лист **Задание 1**);
  + соберите выводы из визуализации (линейной диаграммы) получившегося результата (назовите этот лист **Задание 2**).

**Основное задание курсовой**

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

Чтобы проверить, всё ли в порядке с нашими данными, составить 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.

**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?

**Задание 2**

Создайте визуализацию (линейную диаграмму) итогового результата.

Какие выводы можно сделать из получившейся визуализации?

**Вывод:**

В целом тенденция положительная.

Среднее количество оплаченных уроков довольно стабильно росло в течение года, был спад в марте, июле и декабре.

**Решение:**

with

first\_payments as

(select user\_id

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

from skyeng\_db.payments

where status\_name = 'success'

and operation\_name = 'Покупка уроков' --'Начисление корпоративному клиенту','Оплата другим способом'???

group by user\_id

order by user\_id

),

all\_dates as

(select date(class\_start\_datetime) as dt

from skyeng\_db.classes

where class\_start\_datetime >= '2016-01-01'

and class\_start\_datetime < '2017-01-01'

group by dt

),

all\_dates\_by\_user as

(select user\_id

, dt

from first\_payments as f\_p

join all\_dates as a\_d

on f\_p.first\_payment\_date <= a\_d.dt

),

payments\_by\_dates as

(select user\_id

, date (transaction\_datetime) as payment\_date

, sum (classes) as transaction\_balance\_change

from skyeng\_db.payments

where status\_name = 'success'

and transaction\_datetime >='2016-01-01'

and transaction\_datetime < '2017-01-01'

group by user\_id, payment\_date

),

payments\_by\_dates\_cumsum as

(select adbu.user\_id

, dt

, transaction\_balance\_change

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

from all\_dates\_by\_user as adbu

left join payments\_by\_dates as pbd

on adbu.user\_id = pbd.user\_id

and adbu.dt = pbd.payment\_date

),

classes\_by\_dates as

(select user\_id

, class\_start\_datetime::date as class\_date

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

from skyeng\_db.classes

where class\_type != 'trial'

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

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

group by user\_id, class\_date

),

classes\_by\_dates\_dates\_cumsum as

(select adbu.user\_id

, dt

, classes

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

from all\_dates\_by\_user as adbu

left join classes\_by\_dates as cbd

on adbu.dt = cbd.class\_date

and adbu.user\_id = cbd.user\_id

),

balances as

(select pbdcs.user\_id

, pbdcs.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 pbdcs

left join classes\_by\_dates\_dates\_cumsum as cbddcs

on pbdcs.user\_id = cbddcs.user\_id

and pbdcs.dt = cbddcs.dt

)

-- select \*

-- from balances

-- order by user\_id, dt

-- limit 1000

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