

ПРОЕКТНАЯ РАБОТА ПО МОДУЛЮ SQL

Курс Scypro «Аналитик данных»

Используется схема skyeng_db. СУБД PostgreSQL

В схеме skyeng_db существуют такие таблицы:

- Students - данные о студентах (потенциальных, которые только оставили заявку, и тех, кто действительно оплатил обучение),
- Orders - данные о заявках на обучение,
- Classes - данные об уроках,
- Payments - данные об оплате,
- Teachers - данные об учителях.

ЗАДАЧА КУРСОВОЙ — 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.

РЕШЕНИЕ:

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

```
WITH first_payments AS (  
    SELECT user_id,  
    MIN(transaction_datetime) AS first_payment_date  
    FROM skyeng_db.payments  
    GROUP BY user_id  
)
```

- -- Соберем таблицу с датами за каждый календарный день 2016 года.

```
all_dates AS (  
    SELECT DISTINCT DATE(class_start_datetime) AS dt
```

FROM skyeng_db.classes

WHERE EXTRACT(YEAR FROM class_start_datetime) = 2016

),

- ☛ -- Узнаем, за какие даты имеет смысл собирать баланс для каждого студента (за те даты, которые следуют после даты оплаты).

all_dates_by_user **AS** (

SELECT user_id,

dt

FROM first_payments

JOIN all_dates **ON DATE**(first_payment_date) <= dt

),

- ☛ -- Найдем все изменения балансов, связанные с успешными транзакциями.

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'

GROUP BY user_id, **DATE**(transaction_datetime)

),

- ☛ -- Найдем баланс студентов, который сформирован только транзакциями.

payments_by_dates_cumsum **AS** (

SELECT all_dates_by_user.user_id user_id,

dt,

transaction_balance_change,

SUM(**COALESCE**(transaction_balance_change, 0)) **OVER** (

PARTITION BY all_dates_by_user.user_id order by dt) **AS**
transaction_balance_change_cs

FROM all_dates_by_user

LEFT JOIN payments_by_dates **ON** payments_by_dates.user_id =

all_dates_by_user.user_id

```
AND payments_by_dates.payment_date = all_dates_by_user.dt
),
```

👤 -- Найдем изменения балансов из-за прохождения уроков.

```
classes_by_dates AS (
    SELECT      user_id,
                DATE(class_start_datetime) AS class_date,
                -1*COUNT(id_class) as classes
    FROM skyeng_db.classes
    WHERE class_status IN ('success', 'failed_by_student') AND class_type != 'trial'
    GROUP BY user_id, DATE(class_start_datetime)
),
```

👤 -- По аналогии с уже проделанным шагом для оплат создадим CTE для хранения кумулятивной суммы количества пройденных уроков.

```
classes_by_dates_dates_cumsum AS (
    SELECT all_dates_by_user.user_id,
            dt,
            classes,
            SUM(COALESCE(classes, 0)) OVER (PARTITION
    BY all_dates_by_user.user_id ORDER BY dt) classes_cs
    FROM all_dates_by_user
    LEFT JOIN classes_by_dates ON class_date = dt AND
classes_by_dates.user_id = all_dates_by_user.user_id
),
```

👤 -- Создадим CTE 'balances' с вычисленными балансами каждого студента.

```
balances AS (
    SELECT payments_by_dates_cumsum.user_id,
            payments_by_dates_cumsum.dt,
            transaction_balance_change,
```

```

transaction_balance_change_cs,
classes,
classes_cs,
(classes_cs + transaction_balance_change_cs) AS balance
FROM payments_by_dates_cumsum
JOIN classes_by_dates_dates_cumsum ON classes_by_dates_dates_cumsum.dt =
payments_by_dates_cumsum.dt AND classes_by_dates_dates_cumsum.user_id =
payments_by_dates_cumsum.user_id
)

SELECT *
FROM balances
ORDER BY user_id, dt
LIMIT 1000;

```

🐞 -- Посмотрим, как менялось общее количество уроков на балансах студентов.

```

SELECT dt,
SUM(transaction_balance_change) sum_transaction_balance_change,
SUM(transaction_balance_change_cs) sum_transaction_balance_change_cs,
SUM(classes) sum_classes,
SUM(classes_cs) sum_classes_cs,
SUM(balance) sum_balance
FROM balances
GROUP BY dt
ORDER BY dt;

```

ВЫВОДЫ

В ходе анализа полученных данных о балансах студентов, обнаружено, что есть студенты с отрицательными балансами, т.е. проходившие уроки, предварительно их не оплатив. В связи с этим возникают вопросы к коллегам:

- ⚙ Отслеживается ли доступ учеников к урокам исходя из факта оплаты?
- ⚙ Учитывается ли в выручке "авансовое" прохождение уроков учениками?
- ⚙ Как отслеживается факт оплаты учеником пройденных "авансом" уроков?

На основе данных, полученных из запроса можно построить линейную диаграмму (построено в Metabase):



Исходя из линейной диаграммы, можно сделать следующие выводы:

- ⚙ Количество оплаченных уроков выше количества проведенных уроков
- ⚙ Количество оплат имеет некоторую сезонность: осенью их больше
- ⚙ Студенты к концу года (конец октября-декабрь) учатся усерднее: количество уроков в этот период выше, чем в другие

ОБЩИЙ ЗАПРОС

WITH first_payments **AS** (

```
    SELECT user_id,  
    MIN(transaction_datetime) AS first_payment_date  
    FROM skyeng_db.payments  
    GROUP BY user_id  
    ),
```

all_dates **AS** (

```
    SELECT DISTINCT DATE(class_start_datetime) AS dt  
    FROM skyeng_db.classes  
    WHERE EXTRACT(YEAR FROM class_start_datetime) = 2016  
    ),
```

all_dates_by_user **AS** (

```
    SELECT    user_id,  
             dt  
    FROM first_payments  
    JOIN all_dates ON DATE(first_payment_date) <= 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'  
    GROUP BY user_id, DATE(transaction_datetime)  
    ),
```

payments_by_dates_cumsum **AS** (

```
    SELECT all_dates_by_user.user_id user_id,  
           dt,  
           transaction_balance_change,  
           SUM(COALESCE(transaction_balance_change, 0)) OVER (  
        PARTITION BY all_dates_by_user.user_id order by dt) AS  
    transaction_balance_change_cs  
    FROM all_dates_by_user
```

LEFT JOIN payments_by_dates **ON** payments_by_dates.user_id =
all_dates_by_user.user_id **AND** payments_by_dates.payment_date = all_dates_by_user.dt

),

classes_by_dates **AS** (

SELECT user_id,

DATE(class_start_datetime) **AS** class_date,

-1***COUNT**(id_class) as classes

FROM skyeng_db.classes

WHERE class_status **IN** ('success', 'failed_by_student') **AND** class_type !=
'trial'

GROUP BY user_id, **DATE**(class_start_datetime)

),

classes_by_dates_dates_cumsum **AS** (

SELECT all_dates_by_user.user_id,

dt,

classes,

SUM(**COALESCE**(classes, 0)) **OVER** (**PARTITION**
BY all_dates_by_user.user_id **ORDER BY** dt) classes_cs

FROM all_dates_by_user

LEFT JOIN classes_by_dates **ON** class_date = dt **AND**
classes_by_dates.user_id = all_dates_by_user.user_id

),

balances **AS** (

SELECT payments_by_dates_cumsum.user_id,

payments_by_dates_cumsum.dt,

transaction_balance_change,

transaction_balance_change_cs,

classes,

classes_cs,

(classes_cs + transaction_balance_change_cs) **AS** balance

FROM payments_by_dates_cumsum

JOIN classes_by_dates_dates_cumsum **ON** classes_by_dates_dates_cumsum.dt =
payments_by_dates_cumsum.dt **AND** classes_by_dates_dates_cumsum.user_id =
payments_by_dates_cumsum.user_id

)

SELECT dt,

```
SUM(transaction_balance_change) sum_transaction_balance_change,  
SUM(transaction_balance_change_cs) sum_transaction_balance_change_cs,  
SUM(classes) sum_classes,  
SUM(classes_cs) sum_classes_cs,  
SUM(balance) sum_balance
```

```
FROM balances
```

```
GROUP BY dt
```

```
ORDER BY dt;
```