

Шаги выполнения задания:

0) Подготовка таблицы

Первым делом (перед началом работы с таблицей) была изменена кодировка файла с Windows-1251 в UTF-8 (так как в Windows-1251 не отображались русские символы), удалены лишние столбцы и заменён символ разделителя с ";" на "," (так как "," поддерживается как стандартный делимитр в большем количестве редакторов). Всё эти шаги представлены в файле `./preprocessing.ipynb`.

Полученная таблица `./data/table_refactored.csv`:

CRED_ID	DATA	OPER	SUMMA
0	1 2020-01-01	Вынос на просрочку	100
1	1 2020-01-06	Вынос на просрочку	20
2	1 2020-01-11	Гашение просрочки	120
3	1 2020-01-16	Вынос на просрочку	100
4	1 2020-01-21	Гашение просрочки	20
5	1 2020-01-26	Гашение просрочки	20
6	1 2020-01-31	Гашение просрочки	20
7	1 2020-02-05	Вынос на просрочку	20
8	1 2020-02-10	Вынос на просрочку	20
9	1 2020-02-15	Гашение просрочки	80
10	2 2020-01-05	Вынос на просрочку	50
11	2 2020-01-10	Вынос на просрочку	10
12	2 2020-01-15	Вынос на просрочку	20
13	2 2020-01-20	Вынос на просрочку	10
14	3 2020-01-01	Вынос на просрочку	60
15	3 2020-01-15	Гашение просрочки	50
16	3 2020-01-29	Вынос на просрочку	15
17	3 2020-02-12	Гашение просрочки	24

1) Открытие таблицы в SQL редакторе

Тут возможны несколько случаев, либо import заготовленного файла `./data/table_refactored.csv` (если позволяет редактор), либо создание аналогичной таблицы следующей командой:

```
CREATE TABLE table_refactored (
    CRED_ID INT,
    DATA DATE,
    OPER VARCHAR(30),
    SUMMA INT
);

INSERT INTO table_refactored (CRED_ID, DATA, OPER, SUMMA)
VALUES
    ('1','2020-01-01','Вынос на просрочку','100'),
    ('1','2020-01-06','Вынос на просрочку','20'),
    ('1','2020-01-11','Гашение просрочки','120'),
    ('1','2020-01-16','Вынос на просрочку','100'),
    ('1','2020-01-21','Гашение просрочки','20'),
    ('1','2020-01-26','Гашение просрочки','20'),
    ('1','2020-01-31','Гашение просрочки','20'),
    ('1','2020-02-05','Вынос на просрочку','20'),
    ('1','2020-02-10','Вынос на просрочку','20'),
    ('1','2020-02-15','Гашение просрочки','80'),
    ('2','2020-01-05','Вынос на просрочку','50'),
    ('2','2020-01-10','Вынос на просрочку','10'),
    ('2','2020-01-15','Вынос на просрочку','20'),
    ('2','2020-01-20','Вынос на просрочку','10'),
    ('3','2020-01-01','Вынос на просрочку','60'),
    ('3','2020-01-15','Гашение просрочки','50'),
    ('3','2020-01-29','Вынос на просрочку','15'),
    ('3','2020-02-12','Гашение просрочки','24');
```

2) Решение задания

2.1) Создание переменных для итерации по заёмщикам

SET

```
@cred_id := 0, -- переменная для отслеживания "группы" заёмщика  
@balance := 0, -- текущий баланс заёмщика  
@date_begin := "-"; -- дата последнего непогашенного кредита
```

2.2) Создание табличного выражения для произведения последующих выборок

WITH

credit_information AS (

SELECT

CRED_ID,
DATA,
OPER,
SUMMA,

CASE -- считаем баланс заёмщика после данной операции

WHEN CRED_ID <> @cred_id AND OPER = "Вынос на просрочку" THEN @balance := -SUMMA -- начали рассматривать нового заёмщика → баланс равен -сумма_заёма

WHEN CRED_ID = @cred_id AND OPER = "Вынос на просрочку" THEN @balance := @balance - SUMMA -- заёмщик старый, но кредит новый → добавляем долг

WHEN CRED_ID = @cred_id AND OPER = "Гашение просрочки" THEN @balance := @balance + SUMMA -- заёмщик старый, перевёл деньги → гасим долг

END AS balance,

CASE -- дата начала текущего кредита

WHEN CRED_ID <> @cred_id AND OPER = "Вынос на просрочку" THEN @date_begin := DATA -- начали рассматривать нового заёмщика → запоминаем дату его первого кредитования

WHEN CRED_ID = @cred_id AND OPER = "Вынос на просрочку" AND @balance = -SUMMA THEN
@date_begin := DATA -- клиент старый, уже гасил просрочку, но решил взять новый кредит → запоминаем новую дату кредитования

ELSE @date_begin -- для всех остальных строк — просто дублируем данные из последней обработанной строки (они те же, что в переменной), так как они относятся к одному кредитному делу

END AS DATA_BEGIN,

@cred_id := CRED_ID AS CRED_ID_ -- обновляем следящую за "группой" переменную

FROM table_refactored

)

SELECT * FROM credit_information;

Результат этого запроса:

CRED_ID	DATA	OPER	SUMMA	balance	DATA_BEGIN	CRED_ID_
1	2020-01-01	Вынос на просрочку	100	-100	2020-01-01	1
1	2020-01-06	Вынос на просрочку	20	-120	2020-01-01	1
1	2020-01-11	Гашение просрочки	120	0	2020-01-01	1
1	2020-01-16	Вынос на просрочку	100	-100	2020-01-16	1
1	2020-01-21	Гашение просрочки	20	-80	2020-01-16	1
1	2020-01-26	Гашение просрочки	20	-60	2020-01-16	1
1	2020-01-31	Гашение просрочки	20	-40	2020-01-16	1
1	2020-02-05	Вынос на просрочку	20	-60	2020-01-16	1
1	2020-02-10	Вынос на просрочку	20	-80	2020-01-16	1
1	2020-02-15	Гашение просрочки	80	0	2020-01-16	1
2	2020-01-05	Вынос на просрочку	50	-50	2020-01-05	2
2	2020-01-10	Вынос на просрочку	10	-60	2020-01-05	2
2	2020-01-15	Вынос на просрочку	20	-80	2020-01-05	2
2	2020-01-20	Вынос на просрочку	10	-90	2020-01-05	2
3	2020-01-01	Вынос на просрочку	60	-60	2020-01-01	3
3	2020-01-15	Гашение просрочки	50	-10	2020-01-01	3
3	2020-01-29	Вынос на просрочку	15	-25	2020-01-01	3
3	2020-02-12	Гашение просрочки	24	-1	2020-01-01	3

2.3) Добавление в табличное выражение нумерации строк, чтобы можно было легко взять последнюю операцию по каждому заемщику

WITH

credit_information AS (

SELECT

CRED_ID,

DATA,

OPER,

SUMMA,

CASE

WHEN CRED_ID <> @cred_id AND OPER = "Вынос на просрочку" THEN @balance := -SUMMA

WHEN CRED_ID = @cred_id AND OPER = "Вынос на просрочку" THEN @balance := @balance - SUMMA

WHEN CRED_ID = @cred_id AND OPER = "Гашение просрочки" THEN @balance := @balance + SUMMA

END AS balance,

CASE

WHEN CRED_ID <> @cred_id AND OPER = "Вынос на просрочку" THEN @date_begin := DATA

WHEN CRED_ID = @cred_id AND OPER = "Вынос на просрочку" AND @balance = -SUMMA THEN

@date_begin := DATA

ELSE @date_begin

END AS DATA_BEGIN,

@cred_id := CRED_ID AS CRED_ID_

FROM table_refactored

),
credit_information_numerated AS (

SELECT

*

ROW_NUMBER() OVER (PARTITION BY CRED_ID ORDER BY DATA DESC) AS oper_order -- нумеруем строки по
очередности операций в порядке уменьшения даты (чтобы можно было легко взять последнюю операцию у
заемщика, она будет под номером 1)

FROM credit_information

ORDER BY CRED_ID ASC, DATA ASC -- возвращаем прежний порядок строк

)

SELECT * FROM credit_information_numerated;

Результат этого запроса:

CRED_ID	DATA	OPER	SUMMA	balance	DATA_BEGIN	CRED_ID_	oper_order
1	2020-01-01	Вынос на просрочку	100	-100	2020-01-01	1	10
1	2020-01-06	Вынос на просрочку	20	-120	2020-01-01	1	9
1	2020-01-11	Гашение просрочки	120	0	2020-01-01	1	8
1	2020-01-16	Вынос на просрочку	100	-100	2020-01-16	1	7
1	2020-01-21	Гашение просрочки	20	-80	2020-01-16	1	6
1	2020-01-26	Гашение просрочки	20	-60	2020-01-16	1	5
1	2020-01-31	Гашение просрочки	20	-40	2020-01-16	1	4
1	2020-02-05	Вынос на просрочку	20	-60	2020-01-16	1	3
1	2020-02-10	Вынос на просрочку	20	-80	2020-01-16	1	2
1	2020-02-15	Гашение просрочки	80	0	2020-01-16	1	1
2	2020-01-05	Вынос на просрочку	50	-50	2020-01-05	2	4
2	2020-01-10	Вынос на просрочку	10	-60	2020-01-05	2	3
2	2020-01-15	Вынос на просрочку	20	-80	2020-01-05	2	2
2	2020-01-20	Вынос на просрочку	10	-90	2020-01-05	2	1
3	2020-01-01	Вынос на просрочку	60	-60	2020-01-01	3	4
3	2020-01-15	Гашение просрочки	50	-10	2020-01-01	3	3
3	2020-01-29	Вынос на просрочку	15	-25	2020-01-01	3	2
3	2020-02-12	Гашение просрочки	24	-1	2020-01-01	3	1

2.4) Использование посчитанной на прошлом шаге таблицы (с нумерацией строк и балансом при каждой операции) для получения требуемого вывода

Заменяем последний выделенный `SELECT * FROM credit_information_numerated;` на следующую команду:

```
SELECT
    CRED_ID,
    DATA_BEGIN,
    CASE -- в качестве DATA_END используется значение DATA из строки (если кредит закрылся на данной
          -- операции) или NULL
        WHEN balance = 0 THEN DATA
        ELSE NULL
    END AS DATA_END
FROM credit_information_numerated
WHERE
    balance = 0 -- информация об операциях по полному закрытию кредита
    OR
    (balance < 0 AND oper_order = 1); -- информация о ещё не закрытых кредитах (строки с oper_order = 1 это
    -- последняя операция заёмщика)
```

Результат этого запроса:

CRED_ID	DATA_BEGIN	DATA_END
1	2020-01-01	2020-01-11
1	2020-01-16	2020-02-15
2	2020-01-05	NULL
3	2020-01-01	NULL

Общий вид решения можно найти в файле *solution.sql*