Calculate Available Balance from Wallet system excluding Expired Credits

Asked 6 years, 1 month ago Modified 5 years ago Viewed 3k times Part of PHP Collective

7

In **credits** table I have

> (id, user_id, process, amount, date_add, date_exp, date_redeemed, remark)

```
SELECT * FROM credits WHERE user_id = 2;
```

```
+----+---------+---------+--------+------------+------------+---------------+--
--------+
| id | user_id | process | amount |  date_add  |  date_exp  | date_redeemed |
remark  |
+----+---------+---------+--------+------------+------------+---------------+--
--------+
| 22 |       2 | Add     | 200.00 | 2018-01-01 | 2019-01-01 |               |
Credit1  |
| 23 |       2 | Add     | 200.00 | 2018-03-31 | 2019-03-31 |               |
Credit2  |
| 24 |       2 | Deduct  | 200.00 |            |            | 2018-04-28    |
Redeemed |
| 25 |       2 | Add     | 200.00 | 2018-07-11 | 2018-10-11 |               |
Campaign |
| 26 |       2 | Deduct  | 50.00  |            |            | 2018-08-30    |
Redeemed |
| 27 |       2 | Add     | 200.00 | 2018-10-01 | 2019-09-30 |               |
Credit3  |
| 28 |       2 | Deduct  | 198.55 |            |            | 2018-10-20    |
Redeemed |
+----+---------+---------+--------+------------+------------+---------------+--
--------+
```

The following query I wrote will only calculate the balance, but I don't know whether the credit is expired and used before expired.

```
SELECT
    u.id,
    email,
    CONCAT(first_name, ' ', last_name) AS name,
    type,
    (CASE
        WHEN (SUM(amount) IS NULL) THEN 0.00
        ELSE CASE
            WHEN
                (SUM(CASE
                    WHEN process = 'Add' THEN amount
                END) - SUM(CASE
```

```
                    WHEN process = 'Deduct' THEN amount
                END)) IS NULL
            THEN
                SUM(CASE
                    WHEN process = 'Add' THEN amount
                END)
            ELSE SUM(CASE
                WHEN process = 'Add' THEN amount
            END) - SUM(CASE
                WHEN process = 'Deduct' THEN amount
            END)
        END
    END) AS balance
FROM
    users u
        LEFT JOIN
    credits c ON u.id = c.user_id
GROUP BY u.id;
```

Or I am doing it in the wrong way? Maybe I should have done the calculation in my backend instead of SQL?

EDIT 1:

I want to calculate the balance of every user's e-wallet, but the credit will expire,

IF it was expired AND not redeemed then exclude from the balance

ELSE IF used before expired AND redeem amount < expire amount THEN (balance - (expire amount - redeem amount))

ELSE IF used before expired AND redeem amount > expire amount THEN the usable balance will be deducted as expire amount is not enough to deduct redeemed amount

EDIT 2:

The query above will output 351.45, my expected output is 201.45. Which will not calculate the redemption on 2018-08-30 as the redeem amount is lower than the expired amount

EDIT 3:

User table:

```
+----+------------+-----------+----------+----------------+----------+
| id | first_name | last_name |   type   |     email      | password |
+----+------------+-----------+----------+----------------+----------+
|  2 | Test       | Oyster    | Employee | test@gmail.com | NULL     |
+----+------------+-----------+----------+----------------+----------+
```

My output:

```
+----+----------------+-------------+----------+---------+
| id |     email      |    name     |   type   | balance |
+----+----------------+-------------+----------+---------+
|  2 | test@gmail.com | Test Oyster | Employee |  351.45 |
+----+----------------+-------------+----------+---------+
```

Expected output:

total (200+200+200) 600

redeemed amount 448.55 (200+50+198.55)

Remaining balance is 151.45

```
+----+----------------+-------------+----------+---------+
| id |     email      |    name     |   type   | balance |
+----+----------------+-------------+----------+---------+
|  2 | test@gmail.com | Test Oyster | Employee |  151.45 |
+----+----------------+-------------+----------+---------+
```

PHP php mysql sql

Share

Improve this question

Follow

edited Nov 23, 2018 at 12:51

Madhur Bhaiya

28.8k ● 10 ● 52 ● 60

asked Nov 22, 2018 at 0:28

OysterD3

282 ● 6 ● 20

I have no idea what you are trying to do. That makes it really hard to determine if your code is correct. – Gordon Linoff Nov 22, 2018 at 0:31

Can you make sample data? And your expected result will be help.. Too hard just see the query.. – dwir182 Nov 22, 2018 at 2:17

@dwir182 I have added sample data, my current output and expected output – OysterD3 Nov 22, 2018 at 3:00

@OysterD3 i mean with table format.. You can use This for formatted table.. – dwir182 Nov 22, 2018 at 3:13

@dwir182 How about now? – OysterD3 Nov 22, 2018 at 3:32

2 Answers

Sorted by: Highest score (default)

11

There are basic structural problems with your current table. So I would propose some alterations to the table structure and subsequently application code. Table structures for Wallet systems can be very detailed; but I would suggest minimum possible

changes here. *I am not suggesting that it would be ideal way; but it should work*. Initially, I will layout some of the problems with the current approach.

Problems:

- What if there are multiple Credits available which are not yet expired ?
- Out of these available Credits, some may actually have been utilized already, but not yet expired. How do we ignore them for available balance ?
- Also, some may have been partially utilized. How do we account for partial utilization ?
- There may be a scenario where a redemption amount spans across multiple unexpired credits. Some may get partially utilized; while some may get fully utilized.

General Practice:

We generally follow **FIFO** (*First In First Out*) approach, to give maximum benefit to the customer. So the older credits (which has higher chance of getting expired without utilization) gets utilized first.

In order to follow FIFO, we will have to effectively use a Looping technique in the query/application code every-time, in order to compute basic things, such as, "Available Wallet Balance", "Expired and Underutilized Credit", etc. Writing a query for this will be cumbersome and *possibly inefficient at bigger scales*

Solution:

We can add one more column `amount_redeemed` in your current table. It basically represent the **amount which has already been redeemed** against a specific credit.

```
ALTER TABLE credits ADD COLUMN amount_redeemed DECIMAL (8,2);
```

So, a filled table would look something like below:

```
+----+---------+---------+--------+-----------------+------------+-------------
--+---------------+----------+
| id | user_id | process | amount | amount_redeemed |  date_add  |  date_exp
| date_redeemed |  remark  |
+----+---------+---------+--------+-----------------+------------+-------------
--+---------------+----------+
| 22 |       2 | Add     | 200.00 |      200.00     | 2018-01-01 | 2019-01-01
|               | Credit1  |
| 23 |       2 | Add     | 200.00 |      200.00     | 2018-03-31 | 2019-03-31
|               | Credit2  |
| 24 |       2 | Deduct  | 200.00 |                 |            |
| 2018-04-28    | Redeemed |
| 25 |       2 | Add     | 200.00 |      0.00       | 2018-07-11 | 2018-10-11
```

```
|               | Campaign |
| 26 |       2 | Deduct   | 50.00  |                 |            |
| 2018-08-30    | Redeemed |
| 27 |       2 | Add      | 200.00 |      48.55      | 2018-10-01 | 2019-09-30
|               | Credit3  |
| 28 |       2 | Deduct   | 198.55 |                 |            |
| 2018-10-20    | Redeemed |
+----+---------+---------+--------+-----------------+------------+-------------
--+---------------+----------+
```

Notice that the `amount_redeemed` against Credit of `id = 25` is **0.00**, using FIFO approach. It got a chance for redemption on `2018-10-20`, but by that time, it has expired already (`date_exp = 2018-10-11`)

So, now once we have this setup, you can do the following things in your application code:

1. **Populate `amount_redeemed` value in existing rows in the table**:

This will be a one time activity. For this, formulating a single query would be difficult (that is why we are here in the first place). So I would suggest you to do it one time in your application code (eg: PHP), using Loops and FIFO approach. Look at Point 3 below, to get an idea of how to do it in application code.

2. **Get Current Available Balance**:

Query for this becomes trivial now, as we just need to calculate Sum of `amount - amount_redeemed` for all `Add` process, which are not yet expired.

```
SELECT SUM(amount - amount_redeemed) AS total_available_credit
FROM credits
WHERE process = 'Add' AND
      date_exp > CURDATE() AND
      user_id = 2
```

3. **Update `amount_redeemed` at the time of redemption**:

In this, you can firstly get all available Credits, which has amount available for redemption, and not yet expired.

```
SELECT id, (amount - amount_redeemed) AS available_credit
FROM credits
WHERE process = 'Add' AND
      date_exp > CURDATE() AND
      user_id = 2 AND
      amount - amount_redeemed > 0
ORDER BY id
```

Now, we can loop over the above query results, and utilize the amount accordingly

```php
// PHP code example

// amount to redeem
$amount_to_redeem = 100;

// Map storing amount_redeemed against id
$amount_redeemed_map = array();

foreach ($rows as $row) {

    // Calculate the amount that can be used against a specific credit
    // It will be the minimum of available credit and amount left to redeem
    $amount_redeemed  = min($row['available_credit'], $amount_to_redeem);

    // Populate the map
    $amount_redeemed_map[$row['id']] = $amount_redeemed;

    // Adjust the amount_to_redeem
    $amount_to_redeem -= $amount_redeemed;

    // If no more amount_to_redeem, we can finish the loop
    if ($amount_to_redeem == 0) {
        break;
    } elseif ($amount_to_redeem < 0) {

       // This should never happen, still if it happens, throw error
       throw new Exception ("Something wrong with logic!");
       exit();
    }

    // if we are here, that means some more amount left to redeem
}
```

Now, you can use two Update queries. First one would update `amount_redeemed` value against all the Credit id(s). Second one would `Insert` the Deduct row using the sum of all individual `amount_redeemed` value.

Share

Improve this answer

Follow

edited Nov 23, 2018 at 12:42

answered Nov 23, 2018 at 12:36

Madhur Bhaiya

28.8k ● 10 ● 52 ● 60

1 Thanks mate, my company decided to use FIFO logic, you answer is really helpful – OysterD3 Dec 8, 2018 at 12:08

I using this approach too, but instead of using 'amount_redeemed', I name the new column as 'credit_available' and store available/remaining credit. Should be faster as only refer to one column during query balance; and easier when using SQL statement as not need to minus. – jeff forest Apr 9, 2019 at 6:27

```sql
SELECT `id`, `email`, `NAME`, `type`,
    (
        ( SELECT SUM(amount) FROM credit_table AS ct1 WHERE u.id = ct1.id AND
```

4

```
process = 'ADD' AND date_exp > CURDATE()) -
        ( SELECT SUM(amount) FROM credit_table AS ct2 WHERE u.id = ct2.id AND
process = 'Deduct' )
    ) AS balance
FROM
    `user_table` AS u
WHERE
    id = 2;
```

Hope it works as you wish

Share

Improve this answer

Follow

edited Nov 23, 2018 at 18:48

Ilyes

14.9k ● 4 ● 30 ● 59

answered Nov 23, 2018 at 5:23

Sachin Sarola

1,031 ● 1 ● 10 ● 28