



# データ分析

# データ分析



CRM(顧客管理システム)の機能の一つ  
たくさんのデータを分析して視覚化する

分析する方法は多数ありますが今回は、

日別、月別、年別、デシル分析、RFM分析を  
主に対応していきます。

# 準備

ルート->コントローラ->ビュー (コントローラはクエリのテスト用)

## **routes/web.php**

```
import App\Http\Controllers\AnalysisController;
```

```
Route::get('analysis', [AnalysisController::class, 'index'])->name('analysis');
```

```
php artisan make:controller Analysis Controller
```

## **AnalysisController**

```
public function index()
{
    return Inertia::render('Analysis');
}
```

Pages/Analysis.vue を作成 (Dashboard.vueをコピー)



# 準備

ナビゲーション

Layouts/Authenticated.vue

顧客管理の下に データ分析のリンクを作成

```
<BreezeNavLink :href="route('analysis')" :active="route().current('analysis')">データ分析</BreezeNavLink>
```



# 期間指定

# 期間指定

どの分析においても、何年何月日から 何年何月日 までという情報は必要

**App\Models\Order.php**

```
public function scopeBetweenDate($query, $startDate = null, $endDate = null)
{
    if(is_null($startDate) && is_null($endDate))
    { return $query; }

    if(!is_null($startDate) && is_null($endDate))
    { return $query->where('created_at', ">=", $startDate); }

    if(is_null($startDate) && !is_null($endDate))
    {
        return $query->where('created_at', '<=', $endDate);
    }

    if(!is_null($startDate) && !is_null($endDate))
    {
        return $query->where('created_at', ">=", $startDate)
        ->where('created_at', '<=', $endDate);
    }
}
```



# 期間指定

コントローラで表示確認

**AnalysisController.php**

```
use App\Models\Order;
```

```
$startDate = '2022-08-01';
```

```
$endDate = '2022-08-10';
```

```
$period = Order::betweenDate($startDate, $endDate)-
```

```
>groupBy('id')
```

```
->selectRaw('id, sum(subtotal) as total,  
customer_name, status, created_at')
```

```
->orderBy('created_at')
```

```
->paginate(50);
```

# 期間指定 (Vue側)

View側 formから設定し表示確認

**Pages/Analysis.vue**

```
import { reactive, onMounted } from 'vue'  
import { getToday } from '@/common'
```

```
onMounted(() => {  
  form.startDate = getToday() form.endDate = getToday()})
```

```
const form = reactive({ startDate: null, endDate: null})
```

```
<template>  
<form @submit.prevent="">  
  From: <input type="date" name="startDate" v-model="form.startDate">  
  To: <input type="date" name="endDate" v-model="form.endDate">  
  <button>分析する</button>  
</form>
```



# 期間指定 修正

日付を指定すると 2022-08-31 00:00:00 になり  
2022-08-31 14:00:00 などが含まれなくなるのでクエリを修正

## App\Models\Order.php

```
use Carbon\Carbon;

public function scopeBetweenDate($query, $startDate = null, $endDate = null)
{
    if(is_null($startDate) && !is_null($endDate))
    {
        $endDate1 = Carbon::parse($endDate)->addDays(1);
        return $query->where('created_at', '<=', $endDate1);
    }

    if(!is_null($startDate) && !is_null($endDate))
    {
        $endDate1 = Carbon::parse($endDate)->addDays(1);
        return $query->where('created_at', ">=", $startDate)
            ->where('created_at', '<=', $endDate1);
    }
}
```



# Ajaxで情報取得

# Ajaxで情報取得の準備

## Routes/api.php

```
use App\Http\Controllers\Api\AnalysisController;
```

```
Route::middleware('auth:sanctum')->get('/analysis',  
[ AnalysisController::class, 'index'])  
->name('api.analysis');
```

Api用のコントローラ作成

```
php artisan make:controller Api/AnalysisController
```



# Ajaxで情報取得の準備

```
App\Http\Controllers\Api\AnalysisController.php
```

```
use App\Models\Order;
```

```
use Illuminate\Http\Request;
```

```
use Illuminate\Http\Response;
```

```
public function index(Request $request)
```

```
{
```

```
    // Ajax通信なのでJsonで返却する必要がある
```

```
    return response()->json([
```

```
        'data' => $request->startDate // 仮設定
```

```
    ], Response::HTTP_OK);
```

```
}
```

# Ajaxで情報取得の準備

Pages\Analysis.vue

```
const form = reactive({ type: 'perDay' }) // 仮で直入力
```

```
const getData = async () => {  
  try{  
    await axios.get('/api/analysis/', {  
      params: {  
        startDate: form.startDate,  
        endDate: form.endDate,  
        type: form.type  
      }  
    })  
    .then( res => {  
      // data.value = res.data  
      console.log(res.data)  
    })  
  } catch (e){  
    console.log(e.message)  
  }  
}
```



# 日別売上のクエリ



# 日別売上のクエリ(練習)

App\Http\Controllers\AnalysisController.php

```
use Illuminate\Support\Facades\DB;
```

1. 購買id毎の売上をまとめ, dateをフォーマットした状態のサブクエリをつくる
2. サブクエリをgroupByで日毎にまとめる

```
public function index()
{
    // 期間指定
    $startDate = '2022-08-01'; $endDate = '2022-08-31';

    // 日別
    $subQuery = Order::betweenDate($startDate, $endDate)
        ->where('status', true)->groupBy('id')
        ->selectRaw('id, SUM(subtotal) as totalPerPurchase,
DATE_FORMAT(created_at, "%Y%m%d") as date');

    $data = DB::table($subQuery)
        ->groupBy('date')
        ->selectRaw('date, sum(totalPerPurchase) as total')
        ->get();

    dd($data);
}
```

# 日別売上のクエリ(本番API)

```
App\Http\Controllers\Api\AnalysisController.php
```

```
use App\Models\Order;
```

```
use Illuminate\Support\Facades\DB;
```

```
public function index(Request $request)
```

```
{
```

```
    $subQuery = Order::betweenDate($request->startDate, $request->endDate);
```

```
    if($request->type === 'perDay')
```

```
{
```

```
        $subQuery->where('status', true)->groupBy('id')->selectRaw('SUM(subtotal) AS  
totalPerPurchase, DATE_FORMAT(created_at, "%Y%m%d") AS date')->groupBy('date');
```

```
        $data = DB::table($subQuery)
```

```
            ->groupBy('date')
```

```
            ->selectRaw('date, sum(totalPerPurchase) as total')
```

```
            ->get();
```


```
}
```

```
    return response()->json([ 'data' => $data, 'type' => $request->type ],
```

```
        Response::HTTP_OK);
```

```
}
```





# 取得したデータを テーブル表示



# テーブル表示

テーブルはPages/Items/Index.vueなどからコピー

## Pages/Analysis.vue

```
import { reactive } from 'vue'
const data = reactive({})
```

```
const getData = async () => {
  略
  .then( res => {
    data.data = res.data.data
  })
}
```

```
<div v-show="data.data">
  <tr v-for="item in data.data" :key="item.date">
    <td>{{ item.date }} </td>
    <td>{{ item.total }} </td>
  </tr>
</div>
```



# vue-charts-3

# vue3-charts




Chart.jsのVue.js3用ラッパー

<https://vue-chart-3.netlify.app/>

<https://www.chartjs.org/docs/latest/>

```
npm i vue-chart-3@3.1.8 chart.js@3.9.1
```



# まずは描画確認

参考

<https://zenn.dev/sa2knight/scraps/1ec7a24b9a36ba>

## Pages/Analysis.vue

```
import Chart from '@/Components/Chart.vue'
```

```
<Chart />
```

## Components/Chart.vue

```
<template>
```

```
  <div>
```

```
    <BarChart :chartData="barData" />
```

```
  </div>
```

```
</template>
```

# まずは描画確認

## Components/Chart.vue

```
<script setup>
```


```
import { Chart, registerables } from "chart.js";
```

```
import { BarChart } from "vue-chart-3";
```

```
import { reactive } from "vue"
```

```
Chart.register(...registerables);
```

```
const barData = reactive({  
  labels: ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul"],  
  datasets: [  
    {  
      label: '売上',  
      data: [65, 59, 80, 81, 56, 55, 40],  
      backgroundColor: "rgb(75, 192, 192)",  
      tension: 0.1,  
    }  
  ]  
})
```



# 日別データを グラフ表示



# 配列を作成

App/Http/Controllers/Api/AnalysisController  
略

\$data ・ ・ 日別集計 コレクション型

```
$labels = $data->pluck('date');  
$totals = $data->pluck('total');
```

```
return response()->json([  
    略  
    'labels' => $labels,  
    'totals' => $totals  
],  
    略);
```

# propsで受け取れるように

## Pages/Analysis.vue

```
const getData = async => {
```

略

```
.then( res => {
```

```
  data.lables = res.data.labels
```

```
  data.totals = res.data.totals
```

```
})
```

```
<div v-show="data.data">
```

```
<Chart :data="data" />  · · props
```

```
</div>
```

# propsとcomputed追加

Propsの内容が変わるのでcomputedでリアルタイム検知

## Pages/Components/Chart.vue

```
import { computed } from 'vue'
const props = defineProps({ 'data' : Object })

const labels = computed(() => props.data.labels )
const totals = computed(() => props.data.totals )

const barData = reactive({
  labels: labels,
  datasets: [
    { data: totals, 略 } ] })

<template>
<div v-show="props.data">
  <BarChart :chartData="barData" />
</div>
</template>
```





# Servicesに 切り分け

# サービスへ切り離す

少しコード量が増えてきたので  
ファットコントローラ防止のためサービスに切り離す事にします。

app\Services\AnalysisService.php

```
<?php
namespace App\Services;
use Illuminate\Support\Facades\DB;

class AnalysisService
{
    public static function perDay($subQuery)
    {
        $query = //略

        return [$data, $labels, $totals ]; //複数の変数を渡すので一旦配列に入れる
    }
}
```

# コントローラをすっきり

```
App\Http\Controllers\Api\AnalysisController.php
```

```
use App\Services\AnalysisService;
```

```
public function index(Request $request)
```

```
{
```

```
    $subQuery = Order::betweenDate($request->startDate, $request->endDate);
```

```
    if($request->type === 'perDay')
```

```
{
```

```
        // 配列を受け取り変数に格納するため list() を使う
```

```
        list($data, $labels, $totals) = AnalysisService::perDay($subQuery);
```

```
}
```

```
    return response()->json([
```

```
        'data' => $data,
```

```
        'type' => $request->type,
```

```
        'labels' => $labels,
```


```
        'totals' => $totals
```

```
    ],
```

```
    Response::HTTP_OK);
```

```
}
```





# 月別・年別分析を 追加

# 月別・年別を追加

App\Http\Controllers\Api\AnalysisController.php

```
public function index(Request $request)
```

```
{
```

```
    $subQuery = Order::betweenDate($request->startDate, $request->endDate);
```

```
    if($request->type === 'perDay')
```

```
    { //略 }
```

```
    if($request->type === 'perMonth')
```

```
    {
```

```
        list($data, $labels, $totals) = AnalysisService::perMonth($subQuery);
```

```
    }
```

```
    if($request->type === 'perYear')
```

```
    {
```

```
        list($data, $labels, $totals) = AnalysisService::perYear($subQuery);
```

```
    }
```

```
        return response()->json(略);
```

```
}
```

# サービスに月別・年別を追加

app\Services\AnalysisService.php

```
<?php
class AnalysisService
{
    public static function perMonth($subQuery)
    {
        // 略
        DATE_FORMAT(created_at, "%Y%m") AS date');
    }
    public static function perYear($subQuery)
    {
        //略
        DATE_FORMAT(created_at, "%Y") AS date');
    }
}
```



# input type="radio"を追加

Pages/Analysis.vue

```
<form @submit.prevent="getData">
```

分析方法<br>

```
<input type="radio" v-model="form.type" value="perDay"
checked><span class="mr-4">日別</span>
```

```
<input type="radio" v-model="form.type"
value="perMonth"><span class="mr-4">月別</span>
```

```
<input type="radio" v-model="form.type"
value="perYear"><span class="mr-4">年別</span>
```

# jsconfig.json

```
"compilerOptions" : {  
  "checkJs" : false // JSはエラーを出さない  
}
```



# デシル分析



# デシル分析

デシル・・・ラテン語で10等分

データを10分割してグループに分ける分析手法

一般的に購入金額に応じてグループ分け

ex)

上位グループの特典をアップ

下位グループにDM送付

# デシル分析の流れ

1. 購買ID毎にまとめる
2. 会員毎にまとめて購入金額順にソートする
3. 購入順に連番を振る
4. 全体の件数を数え、1/10の値や合計金額を取得
5. 10分割しグループ毎に数字を振る
6. 各グループの合計金額・平均金額を表示
7. 構成比を表示 (おまけ)

※Mysql8.0のWindow関数 (ntile())が使えると  
4,5をまとめられるけれど 今回はcase文で頑張ります

# 1.2 会員毎にまとめてソート

練習兼ねて

App\Http\Controller\AnalysisController

// 1. 購買ID毎にまとめる

```
$subQuery = Order::betweenDate($startDate, $endDate)
    ->groupBy('id')
    ->selectRaw('id, customer_id, customer_name, SUM(subtotal) as
totalPerPurchase');
```

// 2. 会員毎にまとめて購入金額順にソートする

```
$subQuery = DB::table($subQuery)
    ->groupBy('customer_id')
    ->selectRaw('customer_id, customer_name, sum(totalPerPurchase)
as total')
    ->orderBy('total', 'desc');
```



### 3. 購入順に連番を振る

```
// statementで変数を設定できる
// set @変数名 = 値 (mysqlの書き方)
// 3. 購入順に連番を振る
DB::statement('set @row_num = 0;');
    $subQuery = DB::table($subQuery)
        ->selectRaw('
            @row_num:= @row_num+1 as row_num,
            customer_id,
            customer_name,
            total');
```

## 4. 全体の件数を数える

// 4. 全体の件数を数え、1/10の値や合計金額を取得

```
$count = DB::table($subQuery)->count();
```

```
$total = DB::table($subQuery)->selectRaw('sum(total) as total')->get();
```

```
$total = $total[0]->total; // 構成比用
```

```
$decile = ceil($count / 10); // 10分の1の件数を変数に入れる
```

```
$bindValues = [];
```

```
$tempValue = 0;
```

```
for($i = 1; $i <= 10; $i++)
```

```
{
```

```
    array_push($bindValues, 1 + $tempValue);
```

```
    $tempValue += $decile;
```

```
    array_push($bindValues, 1 + $tempValue);
```

```
}
```

# 5. 10分割しグループ毎に数字を振る

```
// 5 10分割しグループ毎に数字を振る
DB::statement('set @row_num = 0;');
$subQuery = DB::table($subQuery)
->selectRaw("
    row_num,
    customer_id,
    customer_name,
    total,
    case
        when ? <= row_num and row_num < ? then 1
        when ? <= row_num and row_num < ? then 2
        when ? <= row_num and row_num < ? then 3
        when ? <= row_num and row_num < ? then 4
        when ? <= row_num and row_num < ? then 5
        when ? <= row_num and row_num < ? then 6
        when ? <= row_num and row_num < ? then 7
        when ? <= row_num and row_num < ? then 8
        when ? <= row_num and row_num < ? then 9
        when ? <= row_num and row_num < ? then 10
    end as decile
", $bindValues); // SelectRaw第二引数にバインドしたい数値(配列)をいれる
```



## 6. グループ毎の合計・平均

```
// round, avg はmysqlの関数
// 6. グループ毎の合計・平均
$subQuery = DB::table($subQuery)
    ->groupBy('decile')
    ->selectRaw('decile,
        round(avg(total)) as average,
        sum(total) as totalPerGroup');
```

# 7. 構成比

構成比を出すために変数を使う

// 7 構成比

```
DB::statement("set @total = ${total} ;");
    $data = DB::table($subQuery)
        ->selectRaw('decile,
            average,
            totalPerGroup,
            round(100 * totalPerGroup / @total, 1) as
totalRatio
        ')
        ->get();
```



# デシル分析 画面描画



# View側の調整

テーブルのコードが増えるのでResultTableコンポーネントに分離

## ResultTable.vue

```
const props = defineProps({ 'data' : Object })
```

```
<div v-if="data.type === perDay || data.type === perMonth || data.type  
=== perYear">
```

## Analysis.vue

```
.then( res => {  
  略  
  data.type = res.data.type  
}
```

```
)
```

inputタグ追加

```
<input type="radio" v-model="form.type" value="decile">デシル分析
```

# Laravel側

**Api\AnalysisController.php**

```
use App\Services\DecileService;
```

```
if($request->type === 'decile')
{
    list($data, $labels, $totals) =
DecileService::decile($subQuery);
}
```

**App\Services\DecileService;**

```
class DecileService{
public function decile($subQuery)
{
    $subQuery = 略
}
}
```

# Vue側

## Components/ResultTable.vue

```
<div v-if="data.type === 'decile' ">
<tr>
<th>グループ</th>
<th>平均</th>
<th>金額</th>
<th>構成比</th>
</tr>
<tr>
  <td>{{ item.decile }}</td>
  <td>{{ item.average }}</td>
  <td>{{ item.totalPerGroup }}</td>
  <td>{{ item.totalRatio }}</td>
</tr>
```





# RFM分析

# デシル分析の弱点

検索期間が長期間・・・

過去は優良顧客だったけど現在は通っていない、  
というユーザーも含まれてしまう

検索期間が短期間・・・

得られるデータが少ない

定期的に購入する安定顧客が含まれず、  
一時的に大きな買い物をしたユーザーが優良と扱  
われる

# RFM分析について

Recency 最新購入日

Frequency 購入回数

Monetary 購入金額合計

3つの軸に分ける事で、  
1回のみ高額で購入したユーザー と  
定期的に高額ではない商品を購入しているユーザーは それぞれ別のグループとして扱われる



# RFM分析の流れ



1. 購買ID毎にまとめる
2. 会員毎にまとめて最終購入日、回数、合計金額を取得
3. RFMランクを仮設定する
4. 会員毎のRFMランクを計算する
5. ランク毎の数を計算する(3を再調整)
6. RとFで2次元で表示してみる

# 1. 購買ID毎にまとめる

練習兼ねて

App\Http\Controller\AnalysisController

// 1. 購買ID毎にまとめる

```
$subQuery = Order::betweenDate($startDate,  
$endDate)
```

```
->groupBy('id')
```

```
->selectRaw('id, customer_id,  
customer_name, SUM(subtotal) as  
totalPerPurchase, created_at');
```

## 2. RFMに必要な情報を取得

// datediffで日付の差分, maxで日付の最新日  
// 2. 会員毎にまとめて最終購入日、回数、合計金額を  
取得

```
$subQuery = DB::table($subQuery)  
->groupBy('customer_id')  
->selectRaw('customer_id, customer_name,  
max(created_at) as recentDate,  
datediff(now(), max(created_at)) as recency,  
count(customer_id) as frequency,  
sum(totalPerPurchase) as monetary')
```



# 3. RFMランクを定義する

3つの指標を5つのグループに分ける

2にorderBy()をつけてあたりをつけ、仮で設定しておく

orderBy('recency'), orderBy('frequency', 'desc'), orderBy('monetary', 'desc')

(今回は 2021-09-01 ~ 2022-08-31 の1年間と想定)

手順5 で各ランクの人数を確認して あきらかに偏りがあれば調整する  
後程inputタグでユーザー側から変更できるようにする

ランク	R 最新購入日	F 累計購入回数	M 累計購入金額
5	14日以内	7回以上	30万円以上
4	28日以内	5回以上	20万円以上
3	60日以内	3回以上	10万円以上
2	90日以内	2回以上	3万円以上
1	91日以上	1回のみ	3万円未満

# 4. 会員毎のRFMランクを計算する

```
// 4. 会員毎のRFMランクを計算
$subQuery = DB::table($subQuery)
->selectRaw('customer_id, customer_name,
recentDate, recency, frequency, monetary,
case
    when recency < 14 then 5
    when recency < 28 then 4
    when recency < 60 then 3
    when recency < 90 then 2
    else 1 end as r,
case
    when 7 <= frequency then 5
    when 5 <= frequency then 4
    when 3 <= frequency then 3
    when 2 <= frequency then 2
    else 1 end as f,
case
    when 300000 <= monetary then 5
    when 200000 <= monetary then 4
    when 100000 <= monetary then 3
    when 30000 <= monetary then 2
    else 1 end as m')

```

# 5. ランク毎の数を計算する

```
// 5.ランク毎の数を計算する
```

```
$rCount = DB::table($subQuery)
->groupBy('r')
->selectRaw('r, count(r)')
->orderBy('r', 'desc')
->get();
```

```
$fCount = DB::table($subQuery)
->groupBy('f')
->selectRaw('f, count(f)')
->orderBy('f', 'desc')
->get();
```

```
$mCount = DB::table($subQuery)
->groupBy('m')
->selectRaw('m, count(m)')
->orderBy('m', 'desc')
->get();
```



## 6. RとFで2次元で表示してみる

```
// concatで文字列結合
// 6. RとFで2次元で表示してみる
$data = DB::table($subQuery)
  ->groupBy('r')
  ->selectRaw('concat("r_", r) as rRank,
count(case when f = 5 then 1 end ) as f_5,
count(case when f = 4 then 1 end ) as f_4,
count(case when f = 3 then 1 end ) as f_3,
count(case when f = 2 then 1 end ) as f_2,
count(case when f = 1 then 1 end ) as f_1')
  ->orderBy('rRank', 'desc')
  ->get();
```

# 分析結果イメージ

## 分析結果イメージ

R	F 5	F 4	F 3	F 2	F 1
5	10				
4			20		
3					
2					
1					



# RFM分析

## 画面描画



# クエリの調整 (プレースホルダ)

```
// View側から調整できるように対応  
// 動作確認用の仮配列を作る
```

```
// 4. 会員毎のRFMランクを計算  
$rfmPrms = [  
14, 28, 60, 90, 7, 5, 3, 2, 300000, 200000, 100000,  
30000 ];
```

```
selectRaw(' 略  
case when recency < ? then 5 略  
case when ? <= frequency then 5 略  
case when ? <= monetary then 5 略  
, $rfmPrms);
```

# RFMランク毎の人数 配列作成

// View側に渡す情報 RFMランク毎の人数  
1つの配列で用意するとVue側でv-forで簡単に表示できる

// 5.ランク毎の数を計算する  
\$rCount = 略->pluck('count(r)'); \$fCount = 略->pluck('count(f)');  
\$mCount = 略->pluck('count(m)');

\$eachCount = []; // Vue側に渡すような空の配列  
\$rank = 5; // 初期値5

```
for($i = 0; $i < 5; $i++)  
{  
    array_push($eachCount, [  
        'rank' => $rank, 'r' => $rCount[$i], 'f' => $fCount[$i], 'm' => $mCount[$i], ]);  
    $rank--; // rankを1ずつ減らす  
}
```

dd(\$total, \$eachCount, \$rCount, \$fCount, \$mCount);

# Analysis.vue

```
const form = reactive([
```

```
  略
```

```
  rfmPrms: [
```

```
    14, 28, 60, 90, 7, 5, 3, 2, 300000, 200000, 100000,  
    30000
```

```
  ],
```

```
])
```

```
<input type="radio" name="type" v-model="form.type"  
value="rfm"><span class="mr-4">RFM分析</span>
```

```
<div v-if="form.type === 'rfm' ">
```

```
  RFMランク設定のテーブルを表示
```

```
</div>
```



# Analysis.vue RFMランク

// 配列の順番がRの5 -> 4 -> 3 -> 2 次が Fの5 -> 4 -> 3 -> 2 の順になるので注意

```
<table class="mx-auto">
```

```
<thead>
```

```
<tr>
```

```
<th>ランク</th>
```

```
<th>R (○日以内)</th>
```

```
<th>F (○回以上)</th>
```

```
<th>M (○円以上)</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
<tr>
```

```
<td>5</td>
```

```
<td><input type="number" v-model="form.rfmPrms[0]"></td>
```

```
<td><input type="number" v-model="form.rfmPrms[4]"></td>
```

```
<td><input type="number" v-model="form.rfmPrms[8]"></td>
```

```
</tr>
```

```
<tr>
```

```
<td>4</td>
```

```
<td><input type="number" v-model="form.rfmPrms[1]"></td>
```

```
<td><input type="number" v-model="form.rfmPrms[5]"></td>
```

```
<td><input type="number" v-model="form.rfmPrms[9]"></td>
```

```
</tr>
```

```
略
```

```
</table>
```

# 送信時の処理

## Pages/Analysis.vue

```
const getData = async() => {  
  略  
  params: { rfmPrms: form.rfmPrms }  
}
```

## Api/AnalysisController.php

```
use App\Services\RFMService;  
if($request->type === 'rfm')  
{  
  list($data, $totals, $eachCount) = RFMService::rfm($subQuery, $request->rfmPrms);  
  
  return response()->json([  
    'data' => $data,  
    'type' => $request->type,  
    'eachCount' => $eachCount,  
    'totals' => $totals,  
  ], Response::HTTP_OK);  
}]  
}
```

# サービスに分離

App\Services\RFMService.php

```
class RFMService
{
    public static function rfm($subQuery, $rfmPrms){
        略
        $totals // 変数名変更

        return [ $data, $totals, $eachCount];
    }
}
```



# Vue側で受け取り

RFMとそれ以外でreturnが変わるのでif文で切り分け

## Pages/Analysis.vue

```
const getData = async() => {
```

略

```
.then( res => {
```

略

```
  console.log(res.data)
```

```
  if(res.data.labels){ data.labels = res.data.labels }
```

```
  if(res.data.eachCount){ data.eachCount = res.data.eachCount }
```

```
}
```

```
}
```

```
<div v-show="data.data">
```

```
  <div v-if="data.type !== 'rfm' ">
```

```
    <Chart :data="data" />
```

```
  </div>
```

```
</div>
```

# ResultTable.vue

```
<div v-if="data.type === 'rfm'" >  
  合計 {{ data.totals }} 人
```

RFMランク毎の人数

```
<tr v-for="rfm in data.eachCount" :key="rfm.rank">  
  <td>{{ rfm.rank }}</td>  
  <td>{{ rfm.r }}</td>  
  <td>{{ rfm.f }}</td>  
  <td>{{ rfm.m }}</td>
```

RとFの集計表

```
<tr v-for="rf in data.data" :key="rm.rRank">  
  <td>{{ rf.rRank }}</td>  
  <td>{{ rf.f_5 }}</td>  
  <td>{{ rf.f_4 }}</td>  
  <td>{{ rf.f_3 }}</td>  
  <td>{{ rf.f_2 }}</td>  
  <td>{{ rf.f_1 }}</td>  
</tr>
```



# デバッグ



# デバッグ

RFM分析の期間が短いとエラーが発生  
API通信の場合はdd()で表示できないので  
Logを使って原因調査

```
use Illuminate\Support\Facades\Log;
```

```
Log::debug($subQuery->get());
```

などでログ吐き出す

storage/logs/laravel.log

に保存される

(一度削除した方が見やすい)

(設定ファイルはconfig/logging.php)

# 表示されない原因・対策

## 原因

クエリ取得時に

RFMのランクが欠けていると

\$eachCountの数がずれてうまく表示されない

## 対策

RFMのランクがない場合も0として設定が必要

外部結合(right outer join)を使い

ランクがなければnullと表示させる

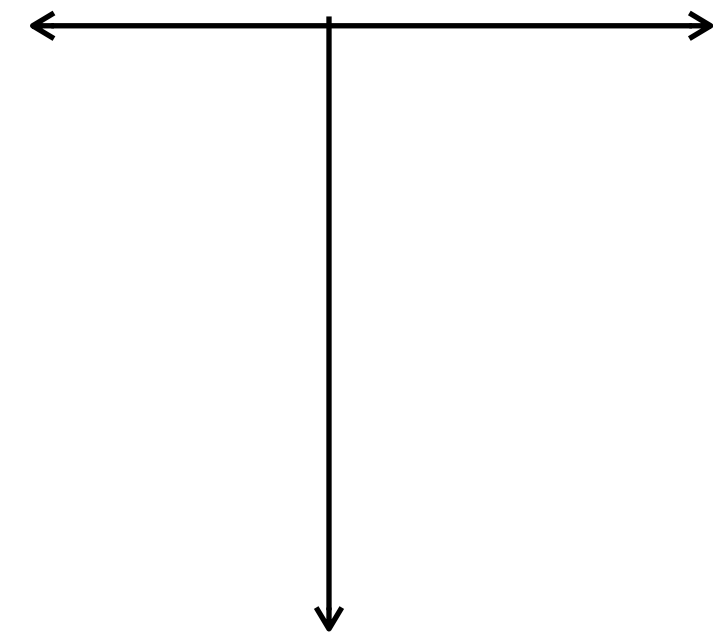
# 外部結合

\$subQuery

customer_id	r
10	1
11	2
12	3
13	1
14	2

ranks

Rank
1
2
3
4
5



count(r)	r	Rank
2	1	1
2	2	2
1	3	3
0	null	4
0	null	5

新たにテーブルを用意  
外部結合(今回はright)し  
groupByすると  
値がない場合はnullと表示される



# テーブル・ダミーデータ作成

```
php artisan make:migration create_ranks_table
```

```
Schema::create('ranks', function (Blueprint $table) {  
    $table->integer('rank');  
});
```

```
php artisan make:seed RankSeeder
```

```
public function run()  
{  
    DB::table('ranks')->insert([  
        ['rank' => 1],  
        ['rank' => 2],  
        ['rank' => 3],  
        ['rank' => 4],  
        ['rank' => 5],  
    ]);  
}
```

DatabaseSeeder.php にも追記しダミーデータ追加

# 動作検証



Apiじゃない方の  
AnalysisControllerに  
rfm処理部分を再度コピペ

(dd で見た方がわかりやすいため)

# クエリ修正

// 5. ランク毎の数を計算する

```
$rCount = DB::table($subQuery)
->rightJoin('ranks', 'ranks.rank', '=', 'r')
->groupBy('rank')
->selectRaw('rank as r, count(r)')
->orderBy('r', 'desc')
->pluck(count(r));
```

残りのf,mにも追記



# クエリ修正

```
// 6. RとFで2次元で表示してみる
$data = DB::table($subQuery)
->rightJoin('ranks', 'ranks.rank', '=', 'r')
->groupBy('rank')
->selectRaw('concat("r_", rank) as rRank,
count(case when f = 5 then 1 end ) as f_5,
```

略

# コード反映



Apiじゃない方の  
AnalysisControllerに書いていたので、

RFMService.phpに反映