LAMPIRAN

Lampiran 1. Dokumentasi

Lampiran 2. Tabel Rekapitulasi Data Penjulan

Table 1. Rekapitualsi Data Penjualan

NO	Nama Produk	Bulan												Jumlah
		1	2	3	4	5	6	7	8	9	10	11	12	
1	Jws endi	5	1	2	3	2	3	0	5	8	0	1	23	53
2	Modul jws 018	8	7	3	7	5	10	6	4	0	1	3	4	58
3	Jws 01	8	15	28	20	29	15	12	14	8	13	14	14	190
4	Jws 018	1	2	3	6	1	1	1	3	1	3	4	6	32
5	Modul jws kalix	1	1	1	1	1	1	1	0	1	0	0	0	8
6	Switching 5v/2a	2	1	3	3	0	4	0	0	1	0	2	1	17
7	P4	2	0	3	0	0	0	0	0	0	0	0	0	5
8	Taqwa media player	5	4	7	7	4	3	9	6	4	9	5	5	68
9	Modul jws-m3	1	0	4	6	2	1	0	1	2	0	1	1	19

10	Jws m3	5	7	15	17	4	3	4	6	2	5	7	6	81
11	Jam digital 08 hijau	1	0	1	0	0	0	3	1	0	1	0	0	7
12	Adaptor 5v/2a	1	0	0	0	0	0	0	0	0	0	0	0	1
13	Modul jws 01	3	7	13	13	2	4	6	4	1	4	4	4	65
14	Jam digital 08 merah	1	0	0	0	0	0	0	0	0	0	0	0	1
15	Papan acrilic cuci tangan sekarang	1	0	0	0	0	0	0	0	0	0	0	0	1
16	Modul dpt	5	16	29	17	7	4	9	7	11	13	9	14	141
17	Switching 5v/6a	1	1	3	1	1	2	0	0	0	0	2	0	11
18	Tf wifi s6u	1	0	1	0	0	0	0	0	0	0	0	0	2
19	Modul jws endi	1	1	3	7	0	2	3	0	2	0	2	2	23
20	Taqwa smart tv android	0	1	0	2	2	1	0	0	0	1	0	0	7
21	Modul jam 1,8	0	2	0	0	0	1	1	1	0	0	1	1	7
22	Modul tanggal 0,8	0	3	0	0	0	1	1	0	0	0	1	1	7
23	7 segment 0,56 inch double digit	0	1	0	0	0	0	0	0	0	0	0	0	1
24	Modul jadwal 0,8	0	1	2	2	0	1	0	1	0	1	0	1	9
25	Adaptor 5v/3a	0	0	2	0	1	0	0	0	0	0	0	0	3
26	Jws kalix	0	0	1	0	0	0	0	0	0	0	0	2	3
27	Papan acrilic dilarang parkir	0	0	1	0	0	0	0	0	0	0	0	0	1
28	Batre cr1220	0	0	2	0	0	0	0	0	0	0	0	0	2
29	remote	0	0	2	0	0	0	0	0	0	0	0	0	2
30	Relay 5v	0	0	1	0	0	0	0	0	0	0	0	0	1
31	Kabel data 10 pin 3m	0	0	1	0	0	0	0	0	0	0	0	0	1
32	7 segment 0,56 inch single digit	0	0	1	0	0	0	0	0	0	0	0	0	1

33	Speaker 4 inch	0	0	2	0	1	0	0	0	0	0	1	0	4
34	Speaker 2 inch	0	0	1	0	0	0	0	0	0	0	0	0	1
35	Tempat wudhu pria	0	0	1	0	0	1	0	0	0	0	0	1	3
36	Tempat wudhu wanita	0	0	1	0	0	1	0	1	0	0	0	1	4
37	Running teks p4	0	0	1	0	0	0	0	0	0	0	0	0	1
38	Modul running text p4 dip	0	0	1	0	0	0	0	0	0	0	0	0	1
39	Speaker 8 ohm 5watt	0	0	1	0	0	0	0	0	0	0	0	0	1
40	Papan acrilic free wifi	0	0	1	0	0	0	0	0	0	0	0	0	1
41	Papan acrilic toilet disiram	0	0	1	0	0	0	0	0	0	0	0	0	1
42	Dilarang duduk	0	0	0	1	0	0	0	1	0	1	0	0	3
43	Jangan buang sampah	0	0	0	1	0	0	0	0	0	0	0	0	1
44	Jam digital suhu	0	0	0	0	0	0	0	0	0	0	0	0	0
45	7-segment hijau 056 cath double digit 10pin	0	0	0	2	0	0	0	0	0	0	0	0	2
46	Remote 01	0	0	0	0	1	0	0	0	0	0	0	0	1
47	Bel sekolah otomatis	0	0	0	0	1	0	0	2	1	0	0	0	4
48	Modul jadwal 01	0	0	0	0	0	1	0	0	1	0	0	0	2
49	Modul 018 6 jadwal	0	0	0	0	1	0	0	0	0	0	0	0	1
50	Modul 018 7 jadwal	0	0	0	0	1	0	0	0	0	0	0	0	1
51	Remote ieqi 01	0	0	0	0	1	0	0	0	0	0	0	0	1
52	Jam digital 08	0	0	0	0	0	0	0	0	0	0	0	0	0

53	Remote jws m3	0	0	0	0	0	1	0	0	0	0	0	0	1
54	Remote jws 018	0	0	0	0	0	1	0	0	0	0	0	0	1
55	Kontrol icp	0	0	0	0	0	1	0	0	0	0	0	0	1
56	Remote jws 01	0	0	0	0	0	1	0	0	0	0	0	0	1
57	Pria	0	0	0	0	0	1	0	0	0	0	0	0	1
58	Wanita	0	0	0	0	0	1	0	0	0	0	0	0	1
59	Modul 01	0	0	0	0	0	1	0	0	0	0	1	0	2
60	Resistor 220 ohm 14watt	0	0	0	0	0	0	1	0	0	0	0	0	1
61	Running teks rgb 41x169	0	0	0	0	0	0	1	0	0	0	0	0	1
62	Jam digital 23	0	0	0	0	0	0	0	0	1	1	0	0	2
63	Modul bel sekolah otomatis	0	0	0	0	0	0	0	0	1	1	0	0	2
64	Ngamen gratis	0	0	0	0	0	0	0	0	0	1	0	1	2
65	Dot matrix 8x8 10 pin	0	0	0	0	0	0	0	0	0	1	0	0	1
66	Kabel data 1 meter	0	0	0	0	0	0	0	0	0	1	0	0	1
67	Kabel data 20cm	0	0	0	0	0	0	0	0	0	1	0	0	1
68	Dilarang kencing	0	0	0	0	0	0	0	0	0	0	2	0	2
69	kiblat	0	0	0	0	0	0	0	0	0	0	1	0	1
70	Modul tanggal 08	0	0	0	0	0	0	0	0	0	0	1	0	1
71	Ir receiver	0	0	0	0	0	0	0	0	0	0	0	1	1

Lampiran 3. *Source code* Proses Algoritma Apriori

```
<?php

namespace App\Http\Controllers;

use App\Models\DetailOrder;

use App\Models\HasilApriori;

use App\Models\Product;

use App\Models\ProductItemSet;
```

```
use App\Models\ProsesApriori;
use Illuminate\Http\Request;
use Illuminate\Support\Carbon;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
class ProsesAprioriController extends Controller
{
    public function index(Request $request)
    {
        $title = 'Proses Apriori';
        $startYear = Carbon::now()
            ->subYears(4)
            ->year;
        $endYear = $startYear + 3;
        $years = range($startYear, $endYear);
        $date = $request->input('date');
        $minSupport = $request->input('min_support');
        $minConfidence = $request->input('min_confidence');

        if ($date) {
            foreach (range(1, 12) as $month) {
                $products = DetailOrder::with('produk:id,nama')
                    ->whereHas('order', function ($query) use ($date, $month) {
                        $query->whereYear('tgl_kirim', $date)
                            ->whereMonth('tgl_kirim', $month)
                            ->where('status', 'Dikirim');
                    })
                    ->select('produk_id', DB::raw('SUM(qty) as total_qty'))
```

```
        ->groupBy('produk_id')
        ->orderByDesc('total_qty')
        ->take(3)
        ->get();

    if ($products->count() > 0) {
        $tanggal = $date . '-' . $month . '-' . date('d');
        foreach ($products as $product) {
            $prosesApriori = ProsesApriori::whereYear('date', $date)
                ->whereMonth('date', $month)
                ->productId($product->produk_id)
                ->firstOrNew();
            $prosesApriori->product_id = $product->produk_id;
            $prosesApriori->date = $tanggal;
            $prosesApriori->save();
        }
    }
}

$productAprioris = ProsesApriori::whereYear('date', $date)
    ->select('product_id')
    ->groupBy('product_id')
    ->get();

/*hasil 1 setitem*/
$satuSetItem = [];

foreach ($productAprioris as $productApriori) {
```

```
                $productCounts = [];
                foreach (range(1, 12) as $month) {
                    $proApriori = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
                        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
                        ->join('orders', 'orders.id', '=', 'detail_orders.id')
                        ->whereYear('proses_aprioris.date', $date)
                        ->whereMonth('proses_aprioris.date', $month)
                        ->where('products.id', $productApriori->product_id)
                        ->first();

                    $productCounts[] = $proApriori ? 1 : 0;
                }

                $totalStatus1 = array_sum($productCounts);
                $persentaseMinSupport = ($totalStatus1 / 12) * 100;

                if ($persentaseMinSupport > $minSupport) {
                    $satuSetItem[] = [
                        'product_id' => $productApriori->product_id,
                        'product_name' => $productApriori->product->nama,
                        'total_transaksi' => $totalStatus1,
                        'persentase' => $persentaseMinSupport,
                    ];
                }
                $productName1ItemSet = Product::where('id', $productApriori->product_id)
                    ->pluck('nama');
```

```
                $productName1ItemSetSlug = Str::slug($productName1ItemSet);
                $productItemSet = ProductItemSet::whereYear('tahun', $date)
                    ->where('kode_item_set', $productName1ItemSetSlug)
                    ->firstOrNew();
                $productItemSet->kode_item_set = $productName1ItemSetSlug;
                $productItemSet->total_transaksi = $totalStatus1;
                $productItemSet->persentase = $persentaseMinSupport;
                $productItemSet->kategori = '1_itemset';
                $productItemSet->tahun = $date;
                $productItemSet->save();
                /*===========*/
            }

            /*hasil 2 setitem*/
            $proses2SetItems = $this->proses2SetItem($satuSetItem);
            $filtered2NameCombinations = $proses2SetItems['filteredNameCombinations'];
            $filtered2Names = $proses2SetItems['filteredNames'];
            $total2YesPerIndex = $proses2SetItems['totalYesPerIndex'];
            $persentase2SetItems = $proses2SetItems['persentase2SetItems'];

            /*hasil 3 set item*/
            $proses3SetItems = $this->proses3SetItem($satuSetItem);
            $filtered3NameCombinations = $proses3SetItems['filteredNameCombinations'];
            $filtered3Names = $proses3SetItems['filteredNames'];
            $total3YesPerIndex = $proses3SetItems['totalYesPerIndex'];
            $persentase3SetItems = $proses3SetItems['persentase3SetItems'];
```

```
            /*hasil 4 set item*/
            $proses4SetItems = $this->proses4SetItem($satuSetItem);
            $filtered4NameCombinations = $proses4SetItems['filteredNameCombinations'];
            $filtered4Names = $proses4SetItems['filteredNames'];
            $total4YesPerIndex = $proses4SetItems['totalYesPerIndex'];
            $persentase4SetItems = $proses4SetItems['persentase4SetItems'];

            /*proses confidence*/
            $confidence = $this->confidence($date, $minConfidence, $satuSetItem);

            $tableConfidenceItemSets = $confidence['tableConfidenceItemSets'];
        } else {
            $products = null;
            $satuSetItem = null;

            /*2 set items*/
            $filtered2NameCombinations = null;
            $filtered2Names = null;
            $total2YesPerIndex = null;
            $persentase2SetItems = null;

            /*3 set items*/
            $filtered3NameCombinations = null;
            $filtered3Names = null;
            $total3YesPerIndex = null;
            $persentase3SetItems = null;
```

```
    /*4 set items*/
    $filtered4NameCombinations = null;
    $filtered4Names = null;
    $total4YesPerIndex = null;
    $persentase4SetItems = null;

    /*confidence 2 item sets*/
    $tableConfidenceItemSets = null;
}

return view('apriories.index', compact(
    'title',
    'products',
    'years',
    'satuSetItem',
    'filtered2NameCombinations',
    'filtered2Names',
    'total2YesPerIndex',
    'persentase2SetItems',
    'filtered3NameCombinations',
    'filtered3Names',
    'total3YesPerIndex',
    'persentase3SetItems',
    'filtered4NameCombinations',
    'filtered4Names',
    'total4YesPerIndex',
    'persentase4SetItems',
    'tableConfidenceItemSets'
```

```
    ));
}

public function proses2SetItem($satuSetItem)
{
    $date = \request('date');
    $minSupport = \request('min_support');

    $productIds = [];
    foreach ($satuSetItem as $item) {
        $productIds[] = $item['product_id'];
    }

    $product2Sets = [];
    $combination2Sets = [];

    foreach (range(1, 12) as $month) {
        $combinations = [];

        $productCount = count($productIds);

        for ($i = 0; $i < $productCount - 1; $i++) {
            for ($j = $i + 1; $j < $productCount; $j++) {
                $productId1 = $productIds[$i];
                $productId2 = $productIds[$j];

                $product1 = Product::find($productId1);
                $product2 = Product::find($productId2);
```

```
                $combinations[] = [
                    'product_id_1' => $productId1,
                    'product_name_1' => $product1->nama,
                    'product_id_2' => $productId2,
                    'product_name_2' => $product2->nama,
                ];
            }
        }

        $combination2Sets = array_merge($combination2Sets, $combinations);

        $results = [];

        foreach ($combinations as $combination) {
            $transaksiItem1 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
                ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
                ->join('orders', 'orders.id', '=', 'detail_orders.id')
                ->whereYear('proses_aprioris.date', $date)
                ->whereMonth('proses_aprioris.date', $month)
                ->where('products.id', $combination['product_id_1'])
                ->first();

            $transaksiItem2 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
                ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
                ->join('orders', 'orders.id', '=', 'detail_orders.id')
```

```
                ->whereYear('proses_aprioris.date', $date)
                ->whereMonth('proses_aprioris.date', $month)
                ->where('products.id', $combination['product_id_2'])
                ->first();

            $results[] = $transaksiItem1 && $transaksiItem2 ? 'Y' : 'N';
        }

        $product2Sets[$month] = $results;
    }

    $uniqueCombinations = array_unique($combination2Sets, SORT_REGULAR);

    // Menghitung jumlah "Y" per kombinasi produk dengan indeks yang sama selama 12 bulan
    $totalYesPerIndex = array_fill(0, count($combinations), 0);

    foreach ($product2Sets as $monthResults) {
        foreach ($monthResults as $index => $result) {
            if ($result === 'Y') {
                $totalYesPerIndex[$index]++;
            }
        }
    }

    $persentase2SetItems = [];
    foreach ($totalYesPerIndex as $persentaseTotalYes) {
        $totalStatus = (int) $persentaseTotalYes;
```

```
            $persentase2SetItems[] = ($totalStatus / 12) * 100;
        }

        $filteredNameCombinations = array_filter($uniqueCombinations, function ($combination, $index) use ($persentase2SetItems, $minSupport) {
            return $persentase2SetItems[$index] >= $minSupport;
        }, ARRAY_FILTER_USE_BOTH);

        $filteredNames = array_map(function ($combination) {
            return [
                'product_name_1' => $combination['product_name_1'] . ' => ',
                'product_name_2' => $combination['product_name_2'],
            ];
        }, $filteredNameCombinations);

        /*simpan ke database untuk kategori 2 itemset*/
        foreach ($filteredNameCombinations as $key => $filteredNameCombination) {
            $mergedName = [];
            foreach ($filteredNames as $index => $namesArray) {
                $mergedName[$index] = Str::slug($namesArray['product_name_1'] . $namesArray['product_name_2']);
            }

            $productItemSet = ProductItemSet::whereYear('tahun', $date)
                ->where('kode_item_set', $mergedName[$key])
                ->firstOrNew();
            $productItemSet->kode_item_set = $mergedName[$key];
            $productItemSet->total_transaksi = $totalYesPerIndex[$key];
```

```
            $productItemSet->persentase = $persentase2SetItems[$key];
            $productItemSet->kategori = '2_itemset';
            $productItemSet->tahun = $date;
            $productItemSet->save();
        }
        /*===========*/

        return compact('filteredNames', 'totalYesPerIndex', 'persentase2SetItems',
'filteredNameCombinations');
    }

    public function proses3SetItem($satuSetItem)
    {
        $date = \request('date');
        $minSupport = \request('min_support');

        $productIds = [];
        foreach ($satuSetItem as $item) {
            $productIds[] = $item['product_id'];
        }

        $product3Sets = [];
        $combination3Sets = [];

        foreach (range(1, 12) as $month) {
            $combinations = [];

            $productCount = count($productIds);
```

```
                for ($i = 0; $i < $productCount - 2; $i++) {
                    for ($j = $i + 1; $j < $productCount - 1; $j++) {
                        for ($k = $j + 1; $k < $productCount; $k++) {
                            $productId1 = $productIds[$i];
                            $productId2 = $productIds[$j];
                            $productId3 = $productIds[$k];

                            $product1 = Product::find($productId1);
                            $product2 = Product::find($productId2);
                            $product3 = Product::find($productId3);

                            $combinations[] = [
                                'product_id_1' => $productId1,
                                'product_name_1' => $product1->nama,
                                'product_id_2' => $productId2,
                                'product_name_2' => $product2->nama,
                                'product_id_3' => $productId3,
                                'product_name_3' => $product3->nama,
                            ];
                        }
                    }
                }

                $combination3Sets = array_merge($combination3Sets, $combinations);

                $results = [];
```

```
foreach ($combinations as $combination) {
    $transaksiItem1 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
        ->whereMonth('proses_aprioris.date', $month)
        ->where('products.id', $combination['product_id_1'])
        ->first();

    $transaksiItem2 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
        ->whereMonth('proses_aprioris.date', $month)
        ->where('products.id', $combination['product_id_2'])
        ->first();

    $transaksiItem3 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
        ->whereMonth('proses_aprioris.date', $month)
        ->where('products.id', $combination['product_id_3'])
        ->first();
```

```
            $results[] = $transaksiItem1 && $transaksiItem2 && $transaksiItem3 ? 'Y' : 'N';
        }

        $product3Sets[$month] = $results;
    }

    $uniqueCombinations = array_unique($combination3Sets, SORT_REGULAR);

    // Menghitung jumlah "Y" per kombinasi produk dengan indeks yang sama selama 12 bulan
    $totalYesPerIndex = array_fill(0, count($combinations), 0);

    foreach ($product3Sets as $monthResults) {
        foreach ($monthResults as $index => $result) {
            if ($result === 'Y') {
                $totalYesPerIndex[$index]++;
            }
        }
    }

    $persentase3SetItems = [];
    foreach ($totalYesPerIndex as $persentaseTotalYes) {
        $totalStatus = (int) $persentaseTotalYes;
        $persentase3SetItems[] = ($totalStatus / 12) * 100;
    }
```

```
$filteredNameCombinations = array_filter($uniqueCombinations, function ($combination, $index) use ($persentase3SetItems, $minSupport) {
    return $persentase3SetItems[$index] >= $minSupport;
}, ARRAY_FILTER_USE_BOTH);

$filteredNames = array_map(function ($combination) {
    return [
        'product_name_1' => $combination['product_name_1'] . ' => ',
        'product_name_2' => $combination['product_name_2'] . ' => ',
        'product_name_3' => $combination['product_name_3'],
    ];
}, $filteredNameCombinations);

/*simpan ke database untuk kategori 3 itemset*/
foreach ($filteredNameCombinations as $key => $filteredNameCombination) {
    $mergedName = [];
    foreach ($filteredNames as $index => $namesArray) {
        $mergedName[$index] = Str::slug($namesArray['product_name_1'] . $namesArray['product_name_2'] . $namesArray['product_name_3']);
    }

    $productItemSet = ProductItemSet::whereYear('tahun', $date)
        ->where('kode_item_set', $mergedName[$key])
        ->firstOrNew();
    $productItemSet->kode_item_set = $mergedName[$key];
    $productItemSet->total_transaksi = $totalYesPerIndex[$key];
    $productItemSet->persentase = $persentase3SetItems[$key];
    $productItemSet->kategori = '3_itemset';
```

```
            $productItemSet->tahun = $date;
            $productItemSet->save();
        }
        /*===========*/

        return compact('filteredNames', 'totalYesPerIndex', 'persentase3SetItems',
'filteredNameCombinations');
    }

    public function proses4SetItem($satuSetItem)
    {
        $date = \request('date');
        $minSupport = \request('min_support');

        $productIds = [];
        foreach ($satuSetItem as $item) {
            $productIds[] = $item['product_id'];
        }

        $product4Sets = [];
        $combination4Sets = [];

        foreach (range(1, 12) as $month) {
            $combinations = [];

            $productCount = count($productIds);

            for ($i = 0; $i < $productCount - 3; $i++) {
```

```
    for ($j = $i + 1; $j < $productCount - 2; $j++) {
        for ($k = $j + 1; $k < $productCount - 1; $k++) {
            for ($l = $k + 1; $l < $productCount; $l++) {
                $productId1 = $productIds[$i];
                $productId2 = $productIds[$j];
                $productId3 = $productIds[$k];
                $productId4 = $productIds[$l];

                $product1 = Product::find($productId1);
                $product2 = Product::find($productId2);
                $product3 = Product::find($productId3);
                $product4 = Product::find($productId4);

                $combinations[] = [
                    'product_id_1' => $productId1,
                    'product_name_1' => $product1->nama,
                    'product_id_2' => $productId2,
                    'product_name_2' => $product2->nama,
                    'product_id_3' => $productId3,
                    'product_name_3' => $product3->nama,
                    'product_id_4' => $productId4,
                    'product_name_4' => $product4->nama,
                ];
            }
        }
    }
}
```

```
$combination4Sets = array_merge($combination4Sets, $combinations);

$results = [];

foreach ($combinations as $combination) {
    $transaksiItem1 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
        ->whereMonth('proses_aprioris.date', $month)
        ->where('products.id', $combination['product_id_1'])
        ->first();

    $transaksiItem2 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
        ->whereMonth('proses_aprioris.date', $month)
        ->where('products.id', $combination['product_id_2'])
        ->first();

    $transaksiItem3 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
        ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
        ->join('orders', 'orders.id', '=', 'detail_orders.id')
        ->whereYear('proses_aprioris.date', $date)
```

```
                ->whereMonth('proses_aprioris.date', $month)
                ->where('products.id', $combination['product_id_3'])
                ->first();

            $transaksiItem4 = ProsesApriori::join('products', 'proses_aprioris.product_id', 'products.id')
                ->join('detail_orders', 'detail_orders.produk_id', '=', 'products.id')
                ->join('orders', 'orders.id', '=', 'detail_orders.id')
                ->whereYear('proses_aprioris.date', $date)
                ->whereMonth('proses_aprioris.date', $month)
                ->where('products.id', $combination['product_id_4'])
                ->first();

            $results[] = $transaksiItem1 && $transaksiItem2 && $transaksiItem3 && $transaksiItem4 ? 'Y' : 'N';
        }

        $product4Sets[$month] = $results;
    }

    $uniqueCombinations = array_unique($combination4Sets, SORT_REGULAR);

    // Menghitung jumlah "Y" per kombinasi produk dengan indeks yang sama selama 12 bulan
    $totalYesPerIndex = array_fill(0, count($combinations), 0);

    foreach ($product4Sets as $monthResults) {
        foreach ($monthResults as $index => $result) {
```

```
                if ($result === 'Y') {
                    $totalYesPerIndex[$index]++;
                }
            }
        }

        $persentase4SetItems = [];
        foreach ($totalYesPerIndex as $persentaseTotalYes) {
            $totalStatus = (int) $persentaseTotalYes;
            $persentase4SetItems[] = ($totalStatus / 12) * 100;
        }

        $filteredNameCombinations = array_filter($uniqueCombinations, function ($combination, $index) use ($persentase4SetItems, $minSupport) {
            return $persentase4SetItems[$index] >= $minSupport;
        }, ARRAY_FILTER_USE_BOTH);

        $filteredNames = array_map(function ($combination) {
            return [
                'product_name_1' => $combination['product_name_1'] . ' => ',
                'product_name_2' => $combination['product_name_2'] . ' => ',
                'product_name_3' => $combination['product_name_3'] . ' => ',
                'product_name_4' => $combination['product_name_4'],
            ];
        }, $filteredNameCombinations);

        /*simpan ke database untuk kategori 4 itemset*/
        foreach ($filteredNameCombinations as $key => $filteredNameCombination) {
```

```
            $mergedName = [];
            foreach ($filteredNames as $index => $namesArray) {
                $mergedName[$index] = Str::slug($namesArray['product_name_1'] . $namesArray['product_name_2'] . $namesArray['product_name_3'] . $namesArray['product_name_4']);
            }

            $productItemSet = ProductItemSet::whereYear('tahun', $date)
                ->where('kode_item_set', $mergedName[$key])
                ->firstOrNew();
            $productItemSet->kode_item_set = $mergedName[$key];
            $productItemSet->total_transaksi = $totalYesPerIndex[$key];
            $productItemSet->persentase = $persentase4SetItems[$key];
            $productItemSet->kategori = '4_itemset';
            $productItemSet->tahun = $date;
            $productItemSet->save();
        }
        /*==========*/

        return compact('filteredNames', 'totalYesPerIndex', 'persentase4SetItems', 'filteredNameCombinations');
    }

    public function confidence($date, $minConfidence, $satuSetItem)
    {
        /*hasil 2 item set*/
        $proses2SetItems = $this->proses2SetItem($satuSetItem);
        $filtered2NameCombinations = $proses2SetItems['filteredNameCombinations'];
```

```
            $filtered2Names = $proses2SetItems['filteredNames'];
            $total2YesPerIndex = $proses2SetItems['totalYesPerIndex'];
            $persentase2SetItems = $proses2SetItems['persentase2SetItems'];

            /*hasil 3 set item*/
            $proses3SetItems = $this->proses3SetItem($satuSetItem);
            $filtered3NameCombinations = $proses3SetItems['filteredNameCombinations'];
            $filtered3Names = $proses3SetItems['filteredNames'];
            $total3YesPerIndex = $proses3SetItems['totalYesPerIndex'];
            $persentase3SetItems = $proses3SetItems['persentase3SetItems'];

            /*hasil 4 set item*/
            $proses4SetItems = $this->proses4SetItem($satuSetItem);
            $filtered4NameCombinations = $proses4SetItems['filteredNameCombinations'];
            $filtered4Names = $proses4SetItems['filteredNames'];
            $total4YesPerIndex = $proses4SetItems['totalYesPerIndex'];
            $persentase4SetItems = $proses4SetItems['persentase4SetItems'];

            /*confidence*/
            $confidenceItemSets = [];
            $nameProductConfidence = [];
            $persentaseMinSupportConfidence = [];
            $tableConfidenceItemSets = [];

            if (count($filtered4NameCombinations) > 0) {
                foreach ($filtered4NameCombinations as $key => $value) {
                    $totalYes = $total4YesPerIndex[$key];
```

```
                /*product1 1 => 2 => 3*/
                $productName1 = Str::slug($value['product_name_1'] . '-' . $value['product_name_2'] . '-' . $value['product_name_3']);
                $product1 = ProductItemSet::kodeItemSet($productName1)
                    ->filterKategori('3_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product1->kode_item_set] = ($totalYes / $product1->total_transaksi) * 100;

                /*product2 1 => 2 => 4*/
                $productName2 = Str::slug($value['product_name_1'] . '-' . $value['product_name_2'] . '-' . $value['product_name_4']);
                $product2 = ProductItemSet::kodeItemSet($productName2)
                    ->filterKategori('3_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product2->kode_item_set] = ($totalYes / $product2->total_transaksi) * 100;

                /*product3 2 => 3 => 4*/
                $productName3 = Str::slug($value['product_name_2'] . '-' . $value['product_name_3'] . '-' . $value['product_name_4']);
                $product3 = ProductItemSet::kodeItemSet($productName3)
                    ->filterKategori('3_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product3->kode_item_set] = ($totalYes / $product3->total_transaksi) * 100;
```

```
                /*product3 1 => 3 => 4*/
                $productName4 = Str::slug($value['product_name_1'] . '-' . $value['product_name_3'] . '-' . $value['product_name_4']);
                $product4 = ProductItemSet::kodeItemSet($productName4)
                    ->filterKategori('3_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product4->kode_item_set] = ($totalYes / $product4->total_transaksi) * 100;

                $nameProductConfidence[$key] = $filtered4Names[$key];
                $persentaseMinSupportConfidence[$key] = $persentase4SetItems[$key];
            }

            foreach ($confidenceItemSets as $key => $values) {
                $productNames = $nameProductConfidence[$key];
                $persentaseHasilSupport = $persentaseMinSupportConfidence[$key];

                // Check if both arrays have data for the given key
                if (count($values) > 0 && count($productNames) > 0) {
                    // Assuming there are two items in each array for a key
                    $keys = array_keys($values);
                    $product1 = $keys[0];
                    $product2 = $keys[1];
                    $product3 = $keys[2];
                    $product4 = $keys[3];
```

```
                $confidence1 = $values[$product1];
                $confidence2 = $values[$product2];
                $confidence3 = $values[$product3];
                $confidence4 = $values[$product4];

                // Removing "=>" character from the strings
                $productNames['product_name_1'] = str_replace(" =>", "", $productNames['product_name_1']);
                $productNames['product_name_2'] = str_replace(" =>", "", $productNames['product_name_2']);
                $productNames['product_name_3'] = str_replace(" =>", "", $productNames['product_name_3']);
                $productNames['product_name_4'] = str_replace(" =>", "", $productNames['product_name_4']);

                if ($confidence1 >= $minConfidence) {
                    $tableConfidenceItemSets[] = [
                        'nama_product' => $productNames['product_name_1'] . ", " . $productNames['product_name_2'] . ", " . $productNames['product_name_3'] . " => " . $productNames['product_name_4'],
                        'persentase_hasil_support' => $persentaseHasilSupport,
                        'persentase_hasil_confidence' => $confidence1
                    ];
                }

                if ($confidence2 >= $minConfidence) {
                    $tableConfidenceItemSets[] = [
                        'nama_product' => $productNames['product_name_1'] . ", " . $productNames['product_name_2'] . ", " . $productNames['product_name_4'] . " => " . $productNames['product_name_3'],
```

```
                        'persentase_hasil_support' => $persentaseHasilSupport,
                        'persentase_hasil_confidence' => $confidence2
                    ];
                }

                if ($confidence3 >= $minConfidence) {
                    $tableConfidenceItemSets[] = [
                        'nama_product' => $productNames['product_name_2'] . ", " . $productNames['product_name_3'] . ", " . $productNames['product_name_4'] . " => " . $productNames['product_name_1'],
                        'persentase_hasil_support' => $persentaseHasilSupport,
                        'persentase_hasil_confidence' => $confidence3
                    ];
                }

                if ($confidence4 >= $minConfidence) {
                    $tableConfidenceItemSets[] = [
                        'nama_product' => $productNames['product_name_1'] . ", " . $productNames['product_name_3'] . ", " . $productNames['product_name_4'] . " => " . $productNames['product_name_2'],
                        'persentase_hasil_support' => $persentaseHasilSupport,
                        'persentase_hasil_confidence' => $confidence4
                    ];
                }
            }
        }
    } elseif (count($filtered3NameCombinations) > 0) {
        foreach ($filtered3NameCombinations as $key => $value) {
            $totalYes = $total3YesPerIndex[$key];
```

```
                // product1 1 => 2
                $productName1 = Str::slug($value['product_name_1'] . '-' . $value['product_name_2']);
                $product1 = ProductItemSet::kodeItemSet($productName1)
                    ->filterKategori('2_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product1->kode_item_set] = ($totalYes / $product1->total_transaksi) * 100;

                // product2 1 => 3
                $productName2 = Str::slug($value['product_name_1'] . '-' . $value['product_name_3']);
                $product2 = ProductItemSet::kodeItemSet($productName2)
                    ->filterKategori('2_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product2->kode_item_set] = ($totalYes / $product2->total_transaksi) * 100;

                // product3 2 => 3
                $productName3 = Str::slug($value['product_name_2'] . '-' . $value['product_name_3']);
                $product3 = ProductItemSet::kodeItemSet($productName3)
                    ->filterKategori('2_itemset')
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$product3->kode_item_set] = ($totalYes / $product3->total_transaksi) * 100;
```

```
                $nameProductConfidence[$key] = $filtered3Names[$key];
                $persentaseMinSupportConfidence[$key] = $persentase3SetItems[$key];
            }

            foreach ($confidenceItemSets as $key => $values) {
                $productNames = $nameProductConfidence[$key];
                $persentaseHasilSupport = $persentaseMinSupportConfidence[$key];

                // Check if both arrays have data for the given key
                if (count($values) > 0 && count($productNames) > 0) {
                    // Assuming there are two items in each array for a key
                    $keys = array_keys($values);
                    $product1 = $keys[0];
                    $product2 = $keys[1];
                    $product3 = $keys[2];

                    $confidence1 = $values[$product1];
                    $confidence2 = $values[$product2];
                    $confidence3 = $values[$product3];

                    // Removing "=>" character from the strings
                    $productNames['product_name_1'] = str_replace(" =>", "", $productNames['product_name_1']);
                    $productNames['product_name_2'] = str_replace(" =>", "", $productNames['product_name_2']);
                    $productNames['product_name_3'] = str_replace(" =>", "", $productNames['product_name_3']);
```

```
                    if ($confidence1 >= $minConfidence) {
                        $tableConfidenceItemSets[] = [
                            'nama_product' => $productNames['product_name_1'] . ", " . $productNames['product_name_2'] . " => " . $productNames['product_name_3'],
                            'persentase_hasil_support' => $persentaseHasilSupport,
                            'persentase_hasil_confidence' => $confidence1
                        ];
                    }

                    if ($confidence2 >= $minConfidence) {
                        $tableConfidenceItemSets[] = [
                            'nama_product' => $productNames['product_name_1'] . ", " . $productNames['product_name_3'] . " => " . $productNames['product_name_2'],
                            'persentase_hasil_support' => $persentaseHasilSupport,
                            'persentase_hasil_confidence' => $confidence2
                        ];
                    }

                    if ($confidence3 >= $minConfidence) {
                        $tableConfidenceItemSets[] = [
                            'nama_product' => $productNames['product_name_2'] . ", " . $productNames['product_name_3'] . " => " . $productNames['product_name_1'],
                            'persentase_hasil_support' => $persentaseHasilSupport,
                            'persentase_hasil_confidence' => $confidence3
                        ];
                    }
                }
            }
```

```
        } elseif (count($filtered2NameCombinations) > 0) {
            foreach ($filtered2NameCombinations as $key => $value) {
                $totalYes = $total2YesPerIndex[$key];
                // item set ke 1
                $product1 = Product::where('id', $value['product_id_1'])
                    ->pluck('nama');
                $codeProduct1 = Str::slug($product1);
                $productItemSet1 = ProductItemSet::kodeItemSet($codeProduct1)
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$productItemSet1->kode_item_set] = ($totalYes / $productItemSet1->total_transaksi) * 100;
                // item set ke 2
                $product2 = Product::where('id', $value['product_id_2'])
                    ->pluck('nama');
                $codeProduct2 = Str::slug($product2);
                $productItemSet2 = ProductItemSet::kodeItemSet($codeProduct2)
                    ->where('tahun', $date)
                    ->first();
                $confidenceItemSets[$key][$productItemSet2->kode_item_set] = ($totalYes / $productItemSet2->total_transaksi) * 100;

                $nameProductConfidence[$key] = $filtered2Names[$key];
                $persentaseMinSupportConfidence[$key] = $persentase2SetItems[$key];
            }
            foreach ($confidenceItemSets as $key => $values) {
                $productNames = $nameProductConfidence[$key];
```

```
                $persentaseHasilSupport = $persentaseMinSupportConfidence[$key];
                // Check if both arrays have data for the given key
                if (count($values) > 0 && count($productNames) > 0) {
                    // Assuming there are two items in each array for a key
                    $keys = array_keys($values);
                    $product1 = $keys[0];
                    $product2 = $keys[1];

                    $confidence1 = $values[$product1];
                    $confidence2 = $values[$product2];

                    // Removing "=>" character from the strings
                    $productNames['product_name_1'] = str_replace(" =>", "", $productNames['product_name_1']);
                    $productNames['product_name_2'] = str_replace(" =>", "", $productNames['product_name_2']);
                    if ($confidence1 >= $minConfidence) {
                        $tableConfidenceItemSets[] = [
                            'nama_product' => $productNames['product_name_1'] . " => " . $productNames['product_name_2'],
                            'persentase_hasil_support' => $persentaseHasilSupport,
                            'persentase_hasil_confidence' => $confidence1
                        ];
                    }
                    if ($confidence2 >= $minConfidence) {
                        $tableConfidenceItemSets[] = [
                            'nama_product' => $productNames['product_name_2'] . " => " . $productNames['product_name_1'],
                            'persentase_hasil_support' => $persentaseHasilSupport,
```

```
                        'persentase_hasil_confidence' => $confidence2
                    ];
                }
            }
        }

        if (count($tableConfidenceItemSets) > 0) {
            $lastBatchNumber = HasilApriori::max('no_urut');
            $lastSequenceNumber = $lastBatchNumber ?: 0;
            foreach ($tableConfidenceItemSets as $tableConfidenceItemSet) {
                $hasilApriori = new HasilApriori();
                $hasilApriori->no_urut = $lastSequenceNumber + 1;
                $hasilApriori->kode_pengujian = 'PNG' . str_pad($hasilApriori->no_urut, 5, '0', STR_PAD_LEFT);
                $hasilApriori->penguji = auth()->user()->id;
                $hasilApriori->nama_produk = $tableConfidenceItemSet['nama_product'];
                $hasilApriori->persentase_hasil_support = $tableConfidenceItemSet['persentase_hasil_support'];
                $hasilApriori->persentase_hasil_confidence = $tableConfidenceItemSet['persentase_hasil_confidence'];
                $hasilApriori->tanggal = date('Y-m-d');
                $hasilApriori->save();
            }
        }
        return compact('tableConfidenceItemSets');
    }
}
```