LAMPIRAN CODING

Login.php

```
<?php
session_start() ;
include ("koneksi.php") ;
if ( isset($_SESSION["login"]) ) {
    header("location: index.php") ;
    exit ;
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKG
j7Sk" crossorigin="anonymous">
    <link rel="stylesheet" href="style/login.css">
    <script src="https://use.fontawesome.com/7f6003f9cc.js"></script>
    <title>Login</title>
</head>
<body>
    <div class="login-box">
        <form action="proseslogin.php" method="POST" class="form-login">
            <center><img src="asset/img/Kebap in full.png" class="logo mb-3">
</center>
            <center>
                <h3>Login</h3>
            </center>
            <div class="form-group">
```

```
                        <label for="">Username</label>
                        <input type="text" class="form-control" placeholder="Masukkan
Username" name="username" required>
                    </div>
                    <div class="form-group">
                        <label for="">Password</label>
                        <input type="password" class="form-control"
placeholder="Masukkan Password" name="password">
                    </div>
                    <div class="form-group" class="text-small">
                        <a href="lupapassword.php">Lupa Password ?</a>
                    </div>
                    <button type="submit" class="btn btn-lg btn-primary btn-block"
name="login">Login</button>
                </form>
            </div>
            <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
                integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRk
fj" crossorigin="anonymous">
            </script>
            <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
                integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo
" crossorigin="anonymous">
            </script>
            <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
                integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JK
I" crossorigin="anonymous">
            </script>
        </body>
        </html>
```

FPGrowth.php

```php
<?php
class FPGrowth
{
    protected $support;
    protected $confidence;
    protected $supportPercentage;
    protected $totalTransactions;
    protected $transactions = array();
    protected $itemSet = array();
    protected $orderedItemSet = array();
    protected $tree;
    private $patterns;
    private $rules;

    /**
     * @return mixed
     */
    public function getSupport()
    {
        return $this->support;
    }

    /**
     * @param mixed $support
     */
    public function setSupport($support)
    {
        $this->support = $support;
        return $this;
    }

    /**
     * @return mixed
```

```
 */
public function getConfidence()
{
    return $this->confidence;
}

/**
 * @param mixed $confidence
 */
public function setConfidence($confidence)
{
    $this->confidence = $confidence;
    return $this;
}

/**
 * @return mixed
 */
public function getPatterns()
{
    $freqPattern = [];
    $item = [];
    foreach ($this->patterns as $key => $value) {
        $key = explode(",", $key);
        if (count($key) > 1) {
            $item['item'] = end($key);
            $item['frequentPattern'] = implode(", ", $key);
            $item['frequent'] = $value;
            array_push($freqPattern, $item);
        }
    }
    return $freqPattern;
    // return $this->patterns ;

}
```

```
/**
 * @return mixed
 */
public function getRules()
{
   return $this->rules;
}
/**
 * @return mixed
 */
public function getFrequentItemSet()
{
   $frequentItemSet = array();
   foreach (array_keys($this->itemSet) as $item) {
      $frequentItemSet[$item] = array();
      $frequentItemSet[$item]['qty'] = $this->itemSet[$item];
      $frequentItemSet[$item]['support']  =  $this->itemSet[$item]  /  $this-
>totalTransactions;
   }
   return $frequentItemSet;
}

/**
 * @return mixed
 */
public function getOrderedItemSet()
{
   return $this->orderedItemSet;
}
public function getTree()
{
   return $this->tree;
}
/**
```

```
             * FPGrowth constructor.
             * @param $support 1, 2, 3 ...
             * @param $confidence 0 ... 1
             */
            public function __construct($transactions, $support, $confidence)
            {
                $this->transactions = $transactions;
                $this->totalTransactions = count($this->transactions);
                $this->confidence = $confidence;
                $this->supportPercentage = $support;
                $this->support = $support * $this->totalTransactions;
            }

            /**
             * Do algorithm
             * @param $transactions
             */
            public function run()
            {
                $this->patterns = $this->findFrequentPatterns($this->transactions, $this-
>support);
                $this->rules = $this->generateAssociationRules($this->patterns, $this-
>confidence);
            }

            protected function findFrequentPatterns($transactions, $support_threshold)
            {
                $tree = new FPTree($transactions, $support_threshold, null, null);
                $this->itemSet = $tree->getFrequentItemSet();
                $this->orderedItemSet = $tree->getOrderedItemSet();
                $this->tree = $tree->root;
                $patterns = $tree->minePatterns($support_threshold);
                return $patterns;
            }
```

```
    protected function generateAssociationRules($patterns, $confidence_threshold)
    {
        $rules = [];
        foreach (array_keys($patterns) as $itemsetStr) {
            $itemset = explode(',', $itemsetStr);
            $upper_support = $patterns[$itemsetStr];
            for ($i = 1; $i < count($itemset); $i++) {
                foreach (self::combinations($itemset, $i) as $antecedent) {
                    sort($antecedent);
                    $antecedentStr = implode(',', $antecedent);
                    // extract item except antecedent
                    $consequent = array_diff($itemset, $antecedent);
                    sort($consequent);
                    // convert from array to string
                    $consequentStr = implode(',', $consequent);
                    if (isset($patterns[$antecedentStr])) {
                        $lower_support = $patterns[$antecedentStr];
                        $confidence = (floatval($upper_support) / $lower_support);
                        $support = floatval($upper_support / $this->totalTransactions);
                        if (($confidence >= $confidence_threshold) && ($support >= $this->supportPercentage)) {
                            if (isset($patterns[$consequentStr])) {
                                $liftRatio = $confidence / $patterns[$consequentStr];
                            } else {
                                $liftRatio = $confidence / $this->itemSet[$consequentStr];
                            }
                            $rules[] = [
                                "antecedent" => str_replace(",", ", ", $antecedentStr),
                                "consequent" => $consequentStr,
                                "confidence" => $confidence,
                                "support" => $support,
                                "liftRatio" => number_format($liftRatio, 5)
                            ];
                            // $tes1[] = $antecedentStr;
                        // $tes2[] = $consequent;
                    }
                }
            }
        }
    }
    // return $tes2;
    return $rules;
}
public static function iter($var)
{
    switch (true) {
        case $var instanceof \Iterator:
            return $var;
        case $var instanceof \Traversable:
            return new \IteratorIterator($var);
        case is_string($var):
            $var = str_split($var);

        case is_array($var):
            return new \ArrayIterator($var);

        default:
            $type = gettype($var);
            throw new \InvalidArgumentException("'$type' type is not iterable");
    }
    return;
}
public static function combinations($iterable, $r)
{
    $pool          =          is_array($iterable)          ?          $iterable          :
iterator_to_array(self::iter($iterable));
    $n = sizeof($pool);

    if ($r > $n) {
            return;
        }
        $indices = range(0, $r - 1);
        yield array_slice($pool, 0, $r);

        for (;;) {
            for (;;) {
                for ($i = $r - 1; $i >= 0; $i--) {
                    if ($indices[$i] != $i + $n - $r) {
                        break 2;
                    }
                }
                return;
            }
            $indices[$i]++;
            for ($j = $i + 1; $j < $r; $j++) {
                $indices[$j] = $indices[$j - 1] + 1;
            }
            $row = [];
            foreach ($indices as $i) {
                $row[] = $pool[$i];
            }
            yield $row;
        }
    }
}
```

Fungsi_eclat.php

```
<?php

require("fungsi_kombinasi.php");
include("koneksi.php");

// Fungsi untuk mengubah data transaksi dari format horizontal ke format
vertikal
```

```
function transformTransactions($transactions)
{
    $items = [];

    // Looping untuk setiap transaksi
    foreach ($transactions as $transaction) {
        // Looping untuk setiap item dalam transaksi
        foreach ($transaction as $item) {
            // Cek apakah item tersebut sudah pernah ditemukan sebelumnya
            if (isset($items[$item])) {
                // Jika sudah, maka tambahkan transaksi tersebut ke array item
                $items[$item]['transactions'][] = $transaction;
            } else {
                // Jika belum, maka tambahkan item tersebut ke array dengan
transaksi yang sesuai
                $items[$item] = [
                    'item' => $item,
                    'transactions' => [$transaction]
                ];
            }
        }
    }

    // Urutkan item berdasarkan jumlah transaksi yang terkait
    uasort($items, function ($a, $b) {
        return count($b['transactions']) - count($a['transactions']);
    });

    return $items;
}
// Akhir transformasi horizontal ke vertical

function eclat($patterns2, $transactions, $support_input, $confidence_input,
$items)
{
        $final_rules = [];
        global $rules_3_pattern;

        //Cari Pattern dua pola
        $rules = carijarum($patterns2, $transactions, $support_input);
        $final_rules = array_merge($final_rules, $rules);

        if (count($final_rules) > 1) {
          $rules_3_pattern = carijarum2($rules, $transactions, $support_input); //
Pencarian Pola Selanjutnya diatas 2 pattern

          $final_rules = array_merge($final_rules, $rules_3_pattern); //
Memasukkan Value Second Rules pada fungsi carijarum2
        }

        //Tes Confidence
        $confidence = getConfidence($final_rules, $items, $transactions,
$confidence_input, $support_input);

        // Mengembalikan Hasil Akhir Pola

        return $confidence;
      }

      // Mencari Pola Transaksi
      function carijarum($patterns2, $transactions, $support_input)
      {
        // Memecah Patterns 2 dari string koma menjadi array
        $slices = [];
        $support = 0;
        $patterns_frequent = [];
        $rules = [];
        $jumlah = 1;
```

```
foreach ($patterns2 as $coba) {
    $coba_pisah = explode(", ", $coba);
    array_push($slices, $coba_pisah);
}



foreach ($transactions as $transaction) {
    foreach ($slices as $slice) {
        if (count(array_intersect($transaction, $slice)) == count($slice)) { //
Hasil Jadi Pola Dicari
            $support++;
            array_push($patterns_frequent, $slice);
        }
    }
}
foreach ($patterns_frequent as $pattern_frequent) {
    // menggabungkan pattern frequent agar dimasukkan kedalam associative
    $gabung = implode(", ", $pattern_frequent);

    if (in_array($pattern_frequent, $patterns_frequent)) {
        if ((isset($rules[$gabung]))) {
            $rules[$gabung]['jumlah'] += $jumlah;
            $rules[$gabung]['support'] += $jumlah / count($transactions) * 100;
        } else {

            $gabung = implode(", ", $pattern_frequent);

            //Jika Belum maka akan menambahkan Rules
            $rules[$gabung] = [
                'rules' => $gabung,
                'jumlah' => $jumlah,
                'support' => $jumlah / count($transactions) * 100
            ];
        }
    };
    }
    return $rules;
}
// Akhir pencarian Pola

// Fungsi Urai Pola Sebeluma masuk fungsi cari jarum 2
function urai($patterns2, $range)
{
    $patterns3 = [];
    // Menguraikan Pola Agar Bisa melakukan Kombinasi
    foreach ($patterns2 as $key => $value) { //
        $coba2 = explode(", ", $key);
        $coba[] = $coba2;
    }
    foreach ($coba as $x) {
        foreach ($x as $y) {
            $k[] = $y;
        }
    }
    $unik = array_unique($k);
    foreach (new Combinations($unik, $range) as $tes) {
        $tes2[] = $tes;
    }
    foreach ($tes2 as $key => $values) { // Menggabungkan Pola sepert pada saat
mencari pola untuk 2 patterns
        $gabung = implode(", ", $values);
        array_push($patterns3, $gabung);
        $merged = empty($merged);
    }

    return $patterns3;
}
// Akhir Urai
// Pola Transaksi Diatas 2
function carijarum2($patterns2, $transactions, $support_input, $range = 3)
```

```
{
    $range;
    $patterns3 = [];
    $second_rules = [];
    // print_r( $patterns3 ) ;
    // Akhir Urai Pola
    $patterns3 = urai($patterns2, $range);
    $rules3 = carijarum($patterns3, $transactions, $support_input);

    // Validasi Support Pola 3 apakah memenuhi syarat
    foreach ($rules3 as $key => $x) {
        if ($x['support'] < $support_input) {
            unset($rules3[$key]);      //Hapus Key Array yang tidak memenuhi
support
        }
    }
    if (count($rules3) > 0) {
        foreach ($rules3 as $key => $value) {
            $x = explode(", ", $key);
            $y = $x;                    // Dapat Jumlah Pola Untuk Nilai Range
        }
        $range = count($y) + 1; // Dapat Nilai Range
        $second_rules = array_merge($second_rules, $rules3);
        $rules3 = carijarum2($rules3, $transactions, $support_input, $range);
    }

    return $second_rules; //Mengembalikan Nilai Rules yang sud

}
// Akhir Pola Transaksi Diatas 2

//Fungsi Menentukan Nilai Confidence Final
function getConfidence($final_rules, $items, $transactions, $confidence_input,
$support_input)
{
```

```
$final_rules2 = [];
$items;
// $kombinasi = new KombinasiAkhir ;
foreach ($final_rules as $itemsetStr => $values) {
    $jumlah = 0;
    $itemset = explode(", ", $itemsetStr); //Memecah Pola pattern FinalRules
    $range = count($itemset);
    $antecedentFreq = 0;
    $upper_support = $final_rules[$itemsetStr];
    // print_r ( $final_rules[$itemsetStr] ) ;

    $antecedentStr =  implode(", ", array_slice($itemset, 0, $range - 1));
    $antecedent = array_slice($itemset, 0, $range - 1);
    $consequent = end($itemset);

    if ($final_rules[$itemsetStr]) {
        foreach ($transactions as $x) { // Cek adakah Antecedent Didalam
Transaksi
            if (count(array_intersect($antecedent, $x)) == count($antecedent)) {
                $jml_a = $final_rules[$itemsetStr]['jumlah'];
                $jumlah++;
            }
        }
        foreach ($transactions as $x) {
            $final_rules[$itemsetStr]['antecedent'] = $antecedentStr;
//Menambahkan Elemen baru pada array Final Rules
            $final_rules[$itemsetStr]['consequent'] = $consequent;
            $final_rules[$itemsetStr]['confidence'] = $jml_a / $jumlah * 100;
        }
    }
}
foreach ($final_rules as $key => $x) {
    if ($x['support'] < $support_input || $x['confidence'] <=
$confidence_input) {
```

```
                unset($final_rules[$key]);   //Hapus Key Array yang tidak memenuhi
support  dan Confidence
              }
            }
            return $final_rules;
        };
        // Contoh data transaksi
        $transactions = [];

        $sql = "SELECT kode_transaksi,GROUP_CONCAT(kode_menu) AS
kode_menu, GROUP_CONCAT(nama_menu) AS nama_menu FROM transaksi
GROUP BY kode_transaksi";
        $hasil = $conn->query($sql);

        while ($row = $hasil->fetch_assoc()) {
            $pisah = explode(",", $row["kode_menu"]);
            array_push($transactions, $pisah);
        }

        // Ambil Nilai Support dan Confidence
        $support_input = $_POST['support'];
        $confidence_input = $_POST['confidence'];


        $start = microtime(true);
        $memory_start = memory_get_usage();
        // Transform data transaksi dari format horizontal ke format vertikal
        $items = transformTransactions($transactions);

        // Array untuk menyimpan itemset frequent yang telah lolos minimum support
        $frequentItemsets = [];

        // Looping untuk setiap item dari data vertikal awal
        foreach ($items as $item) {
            // Hitung support dari item tersebut
            $support = count($item['transactions']) / count($transactions) * 100;

            // Tambahkan item tersebut ke array frequent itemsets jika support melebihi
minimum support yang ditentukan
            if ($support >= $support_input) { //ganti input
                $frequentItemsets[] = [ //nilai Support dari Frequent Itemset *Jadi Tabel
                    'itemset' => [$item['item']],
                    'support' => $support,
                    'jumlah' => count($item['transactions']),
                ];
            } else {
                $frequentItemsets2[] = [ //nilai Support dari Frequent Itemset yang tidak
memenuhi support
                    'itemset' => [$item['item']],
                    'support' => $support,
                    'jumlah' => count($item['transactions']),
                ];
            }
        }

        if (empty($frequentItemsets) || count($frequentItemsets) <= 1) { // Jika Tidak
Ada Item Yang Memenuhi Support Sama Sekali
            $frequentItemsets = array_merge($frequentItemsets, $frequentItemsets2);
        }

        // Membuat Kombinasi Pattern kedua dst
        if (!empty($frequentItemsets)) {
            $range = 2;
            $i = 0;
            $itemsets = [];
            $patterns = [];
            $patterns2 = [];
            $merged2 = [];

            foreach ($frequentItemsets as $itemset) {
                array_push($itemsets, $itemset['itemset']);
                $itemsets2[] = $itemset['itemset'];
                $merged2 = array_merge($merged2, $itemset['itemset']); //coba
            }

            if (count($merged2) > 2) { //Kalo Ada Lebih dari 2 item yang memenuhi
support masuk fungsi kombinasi
                foreach (new Combinations($merged2, 2) as $list) {
                    $patterns[] = $list;
                }
                foreach ($patterns as $key => $values) {
                    $gabung = implode(", ", $values);
                    array_push($patterns2, $gabung);  // Mengisi Variabel Patterns 2
                    //   Akhir   perulangan   masukkan   Pola   kedalam   VARIABLE
*$PATTERNS2*
                }
            } else {
                $patterns = $merged2;
                $gabung = implode(", ", $patterns);
                array_push($patterns2, $gabung);
            }
        }
        // Akhir cari kombinasi Pola
        // Cari Jarum Pola
        $rules_2_pattern  =  carijarum($patterns2,  $transactions,  $support_input);  //
Hasil Akhir Rules dengan 2 Pattern

        $final_rules      =      eclat($patterns2,      $transactions,      $support_input,
$confidence_input, $items); // Hasil Fungsi Eclat

        $end = microtime(true);
        $memory_end = memory_get_usage();

        $elapsed_time = $end - $start;
```

```
$memory_used = $memory_end - $memory_start;
foreach ($patterns as $pattern) {
    // echo count( $item['transactions']) ;
    // print_r ( $pattern ) ;
    // $tes = implode( ",", $pattern ) ;
}
// print_r( $patterns );
foreach ($frequentItemsets as $item) {
    // echo count( $item['transactions']) ;
};
```