Given an array a[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array. Example : Input: N = 5 a[] = {2,3,1,2,3} Output: 2 3 Explanation: 2 and 3 occur more than once in the given array.

<?php

function findDuplicates($arr) {

$n = count($arr);

$frequency = [];

echo "Duplicates: ";

for ($i = 0; $i < $n; $i++) {

$element = $arr[$i];

if (isset($frequency[$element])) {

// If the element is already in the frequency array, it's a duplicate

echo $element . " ";

} else {

// Add the element to the frequency array

$frequency[$element] = 1;

}

}

}

// Example usage

$N = 5;

$array = [2, 3, 1, 2, 3];

findDuplicates($array);

?>

Result:

Duplicates: 2 3

Challenge 3:

Create a migration file to create the below structure, use the fields that you consider necessary for each table .

php artisan make:migration create\_employees\_table

// create\_employees\_table.php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateEmployeesTable extends Migration

{

public function up()

{

Schema::create('employees', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

// Add other necessary fields for employees

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('employees');

}

}

php artisan migrate

php artisan make:migration create\_attendance\_table

// create\_attendance\_table.php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAttendanceTable extends Migration

{

public function up()

{

Schema::create('attendance', function (Blueprint $table) {

$table->id();

$table->unsignedBigInteger('employee\_id');

$table->foreign('employee\_id')->references('id')->on('employees')->onDelete('cascade');

$table->timestamp('checkin');

$table->timestamp('checkout')->nullable();

// Add other necessary fields for attendance

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('attendance');

}

}

php artisan migrate

Challenge 4:

Create an application service “groupByOwnersService” to process the array given and return it in the desired format with a test to check the functionality. For example, for associative array ["insurance.txt" => "Company A", "letter.docx" => "Company A", "Contract.docx" => "Company B"] the groupByOwnersService function should return ["Company A" => ["insurance.txt", "letter.docx"], "Company B" => ["Contract.docx"]]

// GroupByOwnersService.php

namespace App\Services;

class GroupByOwnersService

{

public function groupFilesByOwner(array $files): array

{

$groupedFiles = [];

foreach ($files as $file => $owner) {

$groupedFiles[$owner][] = $file;

}

return $groupedFiles;

}

}

// GroupByOwnersServiceTest.php

use App\Services\GroupByOwnersService;

use PHPUnit\Framework\TestCase;

class GroupByOwnersServiceTest extends TestCase

{

public function testGroupFilesByOwner()

{

$service = new GroupByOwnersService();

$files = [

"insurance.txt" => "Company A",

"letter.docx" => "Company A",

"Contract.docx" => "Company B",

];

$result = $service->groupFilesByOwner($files);

$expectedResult = [

"Company A" => ["insurance.txt", "letter.docx"],

"Company B" => ["Contract.docx"],

];

$this->assertEquals($expectedResult, $result);

}

}

vendor/bin/phpunit GroupByOwnersServiceTest.php

vendor/bin/phpunit GroupByOwnersServiceTest.php

composer require --dev phpunit/phpunit