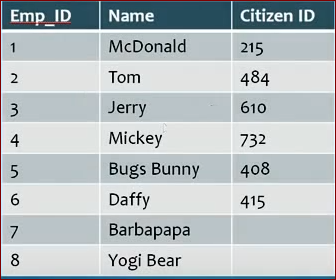
1. **Behavior:** Primary Key is used to identify a row (record) in a table, whereas Unique-key is to prevent duplicate values in a column (with the exception of a null entry).

* Think the table name is employe.
* Primary key
* Primary key can not accept null values. primary key enforces uniqueness of a column. We can have only one Primary key in a table.
* Unique key
* Unique key can accept null values. unique key also enforces uniqueness of a column.you can think if unique key contains null values then why it can be unique ? yes, though it can accept null values it enforces uniqueness of a column. just have a look on the picture.hereEmp\_ID is primary and Citizen ID is unique. Hope you understand. We can use multiple unique key in a table. [](https://i.stack.imgur.com/8Yyax.png)

**What is namespace in Laravel?**

A namespace allows a user to group the functions, classes, and constants under a specific name.

**Candidate Key in DBMS**

**Definition of Candidate Key in DBMS**: A [super key](https://beginnersbook.com/2015/04/super-key-in-dbms/) with no redundant attribute is known as candidate key. Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is that the candidate key should not have any redundant attributes. That’s the reason they are also termed as minimal super key.

**Candidate Key Example**

Lets take an example of table “Employee”. This table has three attributes: Emp\_Id, Emp\_Number&Emp\_Name. Here Emp\_Id&Emp\_Number will be having unique values and Emp\_Name can have duplicate values as more than one employees can have same name.

Emp\_Id Emp\_Number Emp\_Name

------ ---------- --------

E01 2264 Steve

E22 2278 Ajeet

E23 2288 Chaitanya

E45 2290 Robert

**Note**: A [primary key](https://beginnersbook.com/2015/04/primary-key-in-dbms/) is selected from the set of candidate keys. That means we can either have Emp\_Id or Emp\_Number as primary key. The decision is made by DBA (Database administrator)

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

PHP Data Types: Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

### **24) What does isset() function?**

The isset() function checks if the variable is defined and not null.

### **What is the array in PHP?**

An array is used to store multiple values in a single value. In PHP, it orders maps of pairs of keys and values. It saves the collection of the data type.

## Advantage of PHP Array

**Less Code**: We don't need to define multiple variables.

**Easy to traverse**: By the help of single loop, we can traverse all the elements of an array.

**Sorting**: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way:

1. $season=**array**("summer","winter","spring","autumn");

2nd way:

1. $season[0]="summer";
2. $season[1]="winter";
3. $season[2]="spring";
4. $season[3]="autumn";

Example

*File: array1.php*

1. <?php
2. $season=**array**("summer","winter","spring","autumn");
3. echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
4. ?>

Output:

Season are: summer, winter, spring and autumn

*File: array2.php*

1. <?php
2. $season[0]="summer";
3. $season[1]="winter";
4. $season[2]="spring";
5. $season[3]="autumn";
6. echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
7. ?>

Output:

Season are: summer, winter, spring and autumn

## PHP Associative Array

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

1. $salary=**array**("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");

2nd way:

1. $salary["Sonoo"]="350000";
2. $salary["John"]="450000";
3. $salary["Kartik"]="200000";

Example

*File: arrayassociative1.php*

1. <?php
2. $salary=**array**("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
3. echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
4. echo "John salary: ".$salary["John"]."<br/>";
5. echo "Kartik salary: ".$salary["Kartik"]."<br/>";
6. ?>

Output:

Sonoo salary: 350000

John salary: 450000

Kartik salary: 200000

*File: arrayassociative2.php*

1. <?php
2. $salary["Sonoo"]="350000";
3. $salary["John"]="450000";
4. $salary["Kartik"]="200000";
5. echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
6. echo "John salary: ".$salary["John"]."<br/>";
7. echo "Kartik salary: ".$salary["Kartik"]."<br/>";
8. ?>

Output:

Sonoo salary: 350000

John salary: 450000

Kartik salary: 200000

### **Explain some of the PHP string functions?**

There are many array functions in PHP:

* strtolower()
* strtoupper()
* ucfirst()
* lcfirst()
* ucwords()
* strrev()
* strlen()

### **What is the difference between session and cookie?**

The main difference between session and cookie is that cookies are stored on user's computer in the text file format while sessions are stored on the server side.

Cookies can't hold multiple variables, on the other hand, Session can hold multiple variables.

You can manually set an expiry for a cookie, while session only remains active as long as browser is open.

### **Explain PHP explode() function.**

The PHP explode() function breaks a string into an array.

Reverse String

A string can be reversed either using strrev() function or simple PHP code.

For example, on reversing JAVATPOINT it will become TNIOPTAVAJ.

**Logic:**

* Assign the string to a variable.
* Calculate length of the string.
* Declare variable to hold reverse string.
* Run for loop.
* Concatenate string inside for loop.
* Display reversed string.

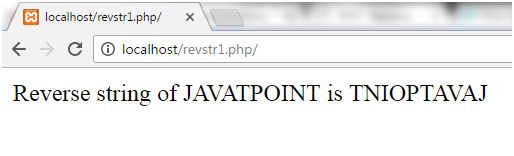
Reverse String using strrev() function

A reverse string program using strrev() function is shown.

**Example:**

1. <?php
2. $string = "JAVATPOINT";
3. echo "Reverse string of $string is " .strrev ( $string );
4. ?>

**Output:**



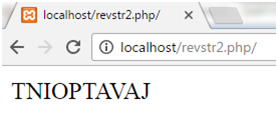
Reverse String Without using strrev() function

A reverse string program without using strrev() function is shown.

**Example:**

1. <?php
2. $string = "JAVATPOINT";
3. $length = strlen($string);
4. **for** ($i=($length-1) ; $i >= 0 ; $i--)
5. {
6. echo $string[$i];
7. }
8. ?>

**Output:**



# Reverse number

A number can be written in reverse order.

For example

12345 = 54321

**Logic:**

* Declare a variable to store reverse number and initialize it with 0.
* Multiply the reverse number by 10, add the remainder which comes after dividing the number by 10.

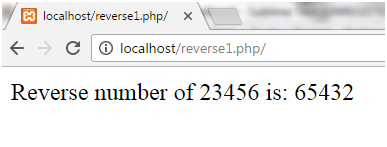
## Reversing Number in PHP

**Example:**

Below progrem shows digits reversal of 23456.

1. <?php
2. $num = 23456;
3. $revnum = 0;
4. **while** ($num > 1)
5. {
6. $rem = $num % 10;
7. $revnum = ($revnum \* 10) + $rem;
8. $num = ($num / 10);
9. }
10. echo "Reverse number of 23456 is: $revnum";
11. ?>

**Output:**



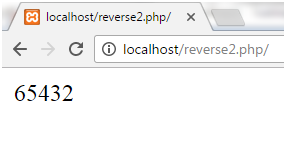
## Reversing Number With strrev () in PHP

**Example:**

Function strrev() can also be used to reverse the digits of 23456.

1. <?php
2. **function** reverse($number)
3. {
4. /\* writes number into string. \*/
5. $num = (string) $number;
6. /\* Reverse the string. \*/
7. $revstr = strrev($num);
8. /\* writes string into int. \*/
9. $reverse = (int) $revstr;
10. **return** $reverse;
11. }
12. echo reverse(23456);
13. ?>

**Output:/strong>**



Next Topic[Reve](https://www.javatpoint.com/php-reverse-string)

Swapping two numbers

Two numbers can be swapped or interchanged. It means first number will become second and second number will become first.

For example

1. a = 20, b = 30
2. After swapping,
3. a = 30, b = 20

There are two methods for swapping:

* By using third variable.
* Without using third variable.

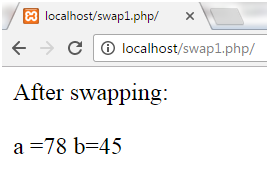
Swapping Using Third Variable

Swap two numbers 45 and 78 using a third variable.

**Example:**

1. <?php
2. $a = 45;
3. $b = 78;
4. // Swapping Logic
5. $third = $a;
6. $a = $b;
7. $b = $third;
8. echo "After swapping:<br><br>";
9. echo "a =".$a."  b=".$b;
10. ?>

**Output:**



Swapping Without using Third Variable

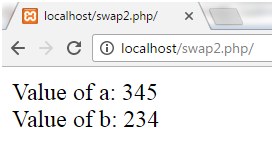
Swap two numbers without using a third variable is done in two ways:

* Using arithmetic operation + and ?
* Using arithmetic operation \* and /

**Example for (+ and -):**

1. <?php
2. $a=234;
3. $b=345;
4. //using arithmetic operation
5. $a=$a+$b;
6. $b=$a-$b;
7. $a=$a-$b;
8. echo "Value of a: $a</br>";
9. echo "Value of b: $b</br>";
10. ?>

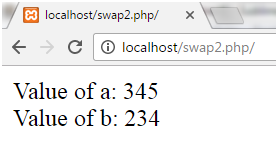
**Output:**



**Example for (\* and /):**

1. <?php
2. $a=234;
3. $b=345;
4. // using arithmetic operation
5. $a=$a\*$b;
6. $b=$a/$b;
7. $a=$a/$b;
8. echo "Value of a: $a</br>";
9. echo "Value of b: $b</br>";
10. ?>

**Output:**



**what is the static variable in function useful for?**

A static variable is defined within a function only the first time, and its value can be modified during function calls as follows:

<!--?php function

testFunction()

{

static $testVariable = 1;

echo $testVariable;

$testVariable++;

}

testFunction();

//1 testFunction();

//2 testFunction(); //3 ?-->

In the first class, sayHi() is actually an instance method which you are calling as a static method and you get away with it because sayHi() never refers to $this.

Static functions are associated with the class, not an instance of the class. As such, $this is not available from a static context ($this isn't pointing to any object).

Simply, static functions function independently of the class where they belong.

$this means, this is an object of this class. It does not apply to static functions.

class test {

publicfunctionsayHi($hi = "Hi") {

$this->hi = $hi;

return $this->hi;

}

}

class test1 {

publicstaticfunctionsayHi($hi) {

$hi = "Hi";

return $hi;

}

}

// Test

$mytest = new test();

print $mytest->sayHi('hello'); // returns 'hello'

print test1::sayHi('hello'); // returns 'Hi'

## What are Abstract Classes and Methods?

Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.

An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.

An abstract class or method is defined with the abstract keyword:

<?php

// Parent class

abstract class Car {

public $name;

public function \_\_construct($name) {

$this->name = $name;

}

abstract public function intro();

}

// Child classes

class Audi extends Car {

public function intro() {

return "Choose German quality! I'm an $this->name!";

}

}

class Volvo extends Car {

public function intro() {

return "Proud to be Swedish! I'm a $this->name!";

}

}

class Citroen extends Car {

public function intro() {

return "French extravagance! I'm a $this->name!";

}

}

// Create objects from the child classes

$audi = new audi("Audi");

echo $audi->intro();

echo "<br>";

$volvo = new volvo("Volvo");

echo $volvo->intro();

echo "<br>";

$citroen = new citroen("Citroen");

echo $citroen->intro();

?>

<?php

## PHP - What are Traits?

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

Traits are declared with the trait keyword:

<?php

trait message1 {

public function msg1() {

echo "OOP is fun! ";

}

}

class Welcome {

use message1;

}

$obj = new Welcome();

$obj->msg1();

?>

## How to select random rows in Laravel?

Sometime we have to get records from database in random order, here I am going to show some ways to get / select random records from database in Laravel.

In laravel 5.2 or above you can use comething like below to get random rows in Laravel.

YourModal::inRandomOrder()->get();

**What does the array operator '===' means?**

$a === $b TRUE if $a and $b have the same key/value pairs in the same order and of the same types.

**What are named routes In laravel?**

**Named routing** is another amazing feature of Laravel framework. Named routes allow referring to routes when generating redirects or Urls more comfortably.  
**You can specify named routes by chaining the name method onto the route definition:**

Route::get('user/profile', function () {

//

})->name('profile');

**You can specify route names for controller actions:**

Route::get('user/profile', 'UserController@showProfile')->name('profile');

**What Laravel's** **EloquentORM?**

Laravel's **Eloquent ORM** is simple Active Record implementation for working with your database. Laravel provide many different ways to interact with your database, Eloquent is most notable of them. Each database table has a corresponding “Model” which is used to interact with that table. Models allow you to query for data in your tables, as well as insert new records into the table.

Below is sample usage for querying and inserting new records in Database with Eloquent.

// Querying or finding records from products table where tag is 'new'

$products= Product::where('tag','new');

// Inserting new record

$product =new Product;

$product->title="Iphone 7";

$product->price="$700";

$product->tag='iphone';

$product->save();

**Explain in bundles?**

In Laravel, **bundles** are also called packages. Packages are the primary way to extend the functionality of Laravel. Packages might be anything from a great way to work with dates like Carbon, or an entire BDD testing framework like Behat.InLaravel, you can create your custom packages too. You can read more about packages from [here](https://laravel.com/docs/5.8/packages)

**Does Laravel Supports Caching?**

Yes, Laravel supports popular caching backends like **Memcached**and **Redis**.  
By default, Laravel is configured to use the **file cache driver**, which stores the serialized, cached objects in the file system.For large projects, it is recommended to use Memcached or Redis.

**How to check request is Ajax or not ?**

In Laravel, we can use $request->ajax() method to check request is ajax or not.

**Example:**

public function saveData(Request $request)

{

if($request->ajax()){

return "Request is of Ajax Type";

}

return "Request is of Http type";

}

### **[27. List types of relationships available in Laravel Eloquent?](https://www.onlineinterviewquestions.com/laravel-interview-questions/" \l "collapseUnfiled27)**

Below are types of relationships supported by Laravel Eloquent ORM.

* One To One
* One To Many
* One To Many (Inverse)
* Many To Many
* Has Many Through
* Polymorphic Relations
* Many To Many Polymorphic Relations

### **[Explain Laravel’s Middleware?](https://www.onlineinterviewquestions.com/laravel-interview-questions/" \l "collapseUnfiled30)**

As the name suggests, Middleware acts as a middleman between request and response. It is a type of filtering mechanism. For example, Laravel includes a middleware that verifies whether the user of the application is authenticated or not. If the user is authenticated, he will be redirected to the home page otherwise, he will be redirected to the login page.

There are two types of Middleware in Laravel.  
Global Middleware: will run on every HTTP request of the application.  
**Route Middleware**: will be assigned to a specific route.

### **[What is composer ?](https://www.onlineinterviewquestions.com/laravel-interview-questions/" \l "collapseUnfiled31)**

**Composer**is a tool for managing dependency in PHP. It allows you to declare the libraries on which your project depends on and will manage (install/update) them for you.  
**Laravel**utilizes Composer to manage its dependencies.

## Laravel composer dump-autoload

Whenever we run **"composer dump-autoload"**

Composer re-reads the**composer.json** file to build up the list of files to autoload.

## Reverse Routing in Laravel

In Laravel reverse routing is generating URL's based on route declarations.**Reverse routing** makes your application so much more flexible.

For example, the below route declaration tells Laravel to execute the action "login" in the user's controller when the request’s URI is **'login'.**

//http://mysite.com/login

Route::get('login', 'users@login');

Using reverse routing we can create a link to it and pass in any parameters that we have defined. Optional parameters, if not supplied, are removed from the generated link.

{{ HTML::link\_to\_action('users@login') }}

It will create a link like http://mysite.com/login in view

## Laravel rollback all migrations

In order to rollback/ remove your all migrations in Laravel, use below artisan command.

php artisan migrate:reset

## How to use "IN" Query in Laravel ?

In Laravel you can use "IN" query as

**Using Eloquent**

$data = YourModel::whereIn('id', array(1, 2, 3))->get();

**Using DB Query builder**

$data = DB::table('your\_table')->whereIn('id', array(1, 2, 3))->get();

## How to use where not in clause in Laravel?

WhereNotIn method of laraveldb builder and Eloquent is used to select records that are not in given array.

**Where not in Example**

**whereNotIn Query Builder Example:**

DB::table('yourtable')->select(..)->whereNotIn('id', [100,200])->get();

**whereNotIn Eloquent Example:**

YourModel::select(..)->whereNotIn('id', [100,200])->get();

Redirect all Laravel routes to Https from Http

Https help us to secure our web application and increase your increase users trust on our website. HTTPS: **HyperText Transfer Protocol Secure** is the secure version of HTTP. Http and Https are protocols over which our data is sent from browser to the server or website we are connected or browsing. When we Https all communication between browser and server is done in an encrypted way. Https protect highly confidential online transactions like online banking and online shopping order forms.

In this article, I am going in detail about Https, here we are going to see different ways to force our Laravel routes or URLs to HTTPs from Http.

**Different ways or methods to force LaravelUrl to HTTPs**

* Using environment variables.
* Using Middleware
* Using Apache Virtual host
* Using .htaccess File

## Laravel eloquent between dates

We can get records between two dates laravel 5 with using Eloquent "whereBetween" method. Here sample function to get posts between two dates from the database.

publicfunctiongetPosts(){

$fromDate = date('Y-m-d' . ' 00:00:00', time());

$toDate = date('Y-m-d' . ' 22:00:40', time());

$posts = POST:: where('user\_id',$this->user\_id)->where('status',1)->whereBetween('posted\_on',[$fromDate, $toDate])->get();

return $posts;

}

How can you tell if a number is even or odd without using any condition or loop?

<?php

$arr=array("0"=>"Even","1"=>"Odd");

$check=13;

echo "Your number is: ".$arr[$check%2];

?>

# Write a program to reverse digits of a number

<?php

// Iterative function to

// reverse digits of num

functionreversDigits($num)

{

    $rev\_num= 0;

    while($num> 1)

    {

        $rev\_num= $rev\_num\* 10 +

                        $num% 10;

        $num= (int)$num/ 10;

    }

    return$rev\_num;

}

// Driver Code

$num= 4562;

echo"Reverse of no. is ",

       reversDigits($num);

// This code is contributed by aj\_36

?>

**Write a Program for finding the biggest number in an array without using any array functions.**

<?php

$numbers = array(12,23,45,20,5,6,34,17,9,56,999);  
$length      = count($numbers);  
$max         = $numbers[0];  
for($i=1;$i<$length;$i++)  
  {  
      if($numbers[$i]>$max)  
        {  
          $max=$numbers[$i];  
        }  
  }  
echo "The biggest number is ".$max;  
?>

**Write a Program for finding the smallest number in an array**

<?php

$numbers=array(12,23,45,20,5,6,34,17,9,56);  
$length=count($numbers);  
$min=$numbers[0];  
for($i=1;$i<$length;$i++)  
  {  
      if($numbers[$i]<$min)  
        {  
          $min=$numbers[$i];  
        }  
  }  
echo "The smallest number is ".$min;

?>

**PHP var\_dump() vsprint\_r()**

The var\_dump() function displays structured information (type and value) about one or more variables.

The print\_r() function displays human-readable information about a variable.

See the following two examples :

<?php

$fruits=array("Banana","Apple","Mango","Coconut");

var\_dump($fruits);

?>

Copy

Output :

array(4) { [0]=> string(6) "Banana" [1]=> string(5) "Apple" [2]=> string(5) "Mango" [3]=> string(7) "Coconut" }

<?php

$fruits=array("Banana","Apple","Mango","Coconut");

print\_r($fruits);

?>

Copy

Output :

Array ( [0] => Banana [1] => Apple [2] => Mango [3] => Coconut )

Notice the handle function which accepts two parameters $request and $next . The $request parameter holds the incoming request URI in your application while $next parameter is used to pass the request deeper into the application. The logic needed is written within the handle function and that brings us to types of middleware-before middleware and after middleware.

Before middleware as the name suggests handles some set of logic before forwarding the request deeper into the application. On the other hand after middleware runs after the request has been handled by the application and the response built.

Before middleware:

<?php

namespaceApp\Http\Middleware;

useClosure;

classRedirectIfSuperAdmin

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @return mixed

\*/

publicfunctionhandle($request,Closure$next)

{

//Your logic goes here

return$next($request);

}

}

After middleware:

<?php

namespaceApp\Http\Middleware;

useClosure;

classRedirectIfSuperAdmin

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @return mixed

\*/

publicfunctionhandle($request,Closure$next)

{

$response=$next($request);

//Your logic goes here e.g return redirect('/)

return$response;

}

}

Categories of Middleware

* Global middleware
* Route middleware

Global middlewares run for every single request that hits the application. Laravel comes with most of these middlewares such as ValidatePostSize, TrimStrings,CheckForMaintenanceMode etc.

Route middlewares run only on routes they are attached to e.g redirectIfAuthenticated.

Registering a Middleware

Any middleware created has to be registered as that is the only way Laravel knows that such exists. To register a middleware simply open the file named kernel.php which is located inside Http folder like so:

This file contains list of all registered middlewares that come with Laravel by default. it contains three major arrays which include $middleware , $middlewareGroups and $routeMiddleware

<?php

namespaceApp\Http;

useIlluminate\Foundation\Http\KernelasHttpKernel;

classKernelextendsHttpKernel

{

/\*\*

\* The application's global HTTP middleware stack.

\*

\* These middleware are run during every request to your application.

\*

\* @var array

\*/

protected$middleware=[

\Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode::class,

\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,

\App\Http\Middleware\TrimStrings::class,

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

\App\Http\Middleware\TrustProxies::class,

];

/\*\*

\* The application's route middleware groups.

\*

\* @var array

\*/

protected$middlewareGroups=[

'web'=>[

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

// \Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

'api'=>[

'throttle:60,1',

'bindings',

],

];

/\*\*

\* The application's route middleware.

\*

\* These middleware may be assigned to groups or used individually.

\*

\* @var array

\*/

protected$routeMiddleware=[

'auth'=>\Illuminate\Auth\Middleware\Authenticate::class,

'auth.basic'=>\Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings'=>\Illuminate\Routing\Middleware\SubstituteBindings::class,

'can'=>\Illuminate\Auth\Middleware\Authorize::class,

'guest'=>\App\Http\Middleware\RedirectIfAuthenticated::class,

'throttle'=>\Illuminate\Routing\Middleware\ThrottleRequests::class,

//the just created middlware

'superadmin'=>\App\Http\Middleware\RedirectIfSuperAdmin::class,

];

}

The $middleware array holds global middlewares which runs for every single HTTP request to the application, so if you want a middleware to run for every request you should register it here. The $middlewareGroups makes it possible to register middlewares in groups making it easier to attach lots of middlewares to a route by using the group name. The $routeMiddleware array holds individual registered middlewares.

Assigning a Middleware

Once a middleware is registered it can be attached to a route in two major ways

* Through the constructor method in a controller
* Through the route

Middleware assignment through constructor

Middleware assignment via a constructor on a controller gives more flexibility as it offers two important functions except($parameters) and only($parameters) which can be used to prevent or allow the middleware to apply to some helper functions in that controller. Without using the helper functions the middleware applies to every single function on that controller.

<?php

useIlluminate\Http\Request;

classForumControllerextendsController

{

publicfunction\_\_construct(){

/\*\*in this case the middleware named auth is applied

to every single function within this controller

\*/

$this->middleware('auth');

}

publicfunctionviewForum(){

returnview('index');

}

publicfunctionedit($id){

}

publicfunctiondelete($id){

}

}

With the except and only functions we can select which functions the middleware will apply to as shown below:

<?php

useIlluminate\Http\Request;

classForumControllerextendsController

{

publicfunction\_\_construct(){

/\*\*the authentication middleware here applies to all functions but

viewForums() and viewForumDetails() and the opposite of this happens

when you use only()

\*/

$this->middleware('auth')->except(['viewForums','viewForumDetails']);

}

publicfunctionviewForums(){

returnview('index');

}

publicfunctionedit($id){

}

publicfunctiondelete($id){

}

publicfunctionviewForumDetails(){

}

}

Middleware assignment through routes

Provided a middleware has been registered in can be attached to the route directly as show below:

<?php

//method 1

Route::get('admin/profile',function(){

//action

})->middleware('auth');

/\*\*method 2

or using the fully qualified class name like so:

\*/

useApp\Http\Middleware\CheckAge;

Route::get('admin/profile',function(){

// action

})->middleware(CheckAge::class);

//method 3

Route::group(['middleware'=>['web']],function(){

//action

});

N:B Middleware groups can be assigned to a route the same way as a single middleware

Middleware Parameters

Additional parameters can be passed to a middleware. A typical example is where each user id is assigned to a role and the middleware checks the role of a user to determine if he/she has access to the requested URI. Parameters can be passed to a middleware as shown below:

<?php

//First method (Through route)

Route::get('admin/profile',function(){

//action

})->middleware('auth:<role>');//<role> here should be replaced by whatever parameter the user intends to pass.

//Second method (Through a controller)

useIlluminate\Http\Request;

classForumControllerextendsController

{

publicfunction\_\_construct(){

$this->middleware('auth:<role>');

}

}

Multiple parameters can be passed to a middleware by separating each parameter by a comma.

<?php

Route::get('admin/profile',function(){

//action

})->middleware('auth:<role>,<age>,<country>');//<role>, <age>, <country> here should be replaced by whatever parameters the user intends to pass.

These parameters are passed to the handle function of the middleware after the $next variable

<?php

classRedirectIfSuperAdmin

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @return mixed

\*/

publicfunctionhandle($request,Closure$next,$role,$age,$country)

{

//Logic for the middleware using the parsed parameters

return$next($request);

}

}

Summary

To create a middleware you go through the following processes

* Create the middleware with artisan command php artisan make:middleware<Middleware Name>.
* Register the middleware in kernel.php located in the app→Http folder
* Write your logic in the created middleware
* Assign middleware to a route or controller

Write a query to get the difference between the highest and lowest salaries

SELECTMAX(salary)-MIN(salary) DIFFERENCE

FROM employees;

# Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database

BY CHAITANYA SINGH | FILED UNDER: [DBMS](https://beginnersbook.com/category/dbms/)

**Normalization** is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let’s discuss about anomalies first then we will discuss normal forms with examples.

## Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let’s take an example to understand this.

**Example**: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp\_id for storing employee’s id, emp\_name for storing employee’s name, emp\_address for storing employee’s address and emp\_dept for storing the department details in which the employee works. At some point of time the table looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_dept |
| 101 | Rick | Delhi | D001 |
| 101 | Rick | Delhi | D002 |
| 123 | Maggie | Agra | D890 |
| 166 | Glenn | Chennai | D900 |
| 166 | Glenn | Chennai | D004 |

The above table is not normalized. We will see the problems that we face when a table is not normalized.

**Update anomaly**: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

**Insert anomaly**: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp\_dept field doesn’t allow nulls.

**Delete anomaly**: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp\_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

## Normalization

Here are the most commonly used normal forms:

* First normal form(1NF)
* Second normal form(2NF)
* Third normal form(3NF)
* Boyce &Codd normal form (BCNF)

## First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

**Example**: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_mobile |
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212  9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123  8123450987 |

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF**as the rule says “each attribute of a table must have atomic (single) values”, the emp\_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_mobile |
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212 |
| 102 | Jon | Kanpur | 9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123 |
| 104 | Lester | Bangalore | 8123450987 |

## Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

**Example**: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

|  |  |  |
| --- | --- | --- |
| teacher\_id | subject | teacher\_age |
| 111 | Maths | 38 |
| 111 | Physics | 38 |
| 222 | Biology | 38 |
| 333 | Physics | 40 |
| 333 | Chemistry | 40 |

**Candidate Keys**: {teacher\_id, subject}  
**Non prime attribute**: teacher\_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher\_age is dependent on teacher\_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:  
**teacher\_details table:**

|  |  |
| --- | --- |
| teacher\_id | teacher\_age |
| 111 | 38 |
| 222 | 38 |
| 333 | 40 |

**teacher\_subject table:**

|  |  |
| --- | --- |
| teacher\_id | subject |
| 111 | Maths |
| 111 | Physics |
| 222 | Biology |
| 333 | Physics |
| 333 | Chemistry |

Now the tables comply with Second normal form (2NF).

## Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

* Table must be in 2NF
* [Transitive functional dependency](https://beginnersbook.com/2015/04/transitive-dependency-in-dbms/) of non-prime attribute on any super key should be removed.

An attribute that is not part of any [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/) is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

* X is a [super key](https://beginnersbook.com/2015/04/super-key-in-dbms/) of table
* Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

**Example**: Suppose a company wants to store the complete address of each employee, they create a table named employee\_details that looks like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_zip | emp\_state | emp\_city | emp\_district |
| 1001 | John | 282005 | UP | Agra | DayalBagh |
| 1002 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Lora | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Lilly | 292008 | UK | Pauri | Bhagwan |
| 1201 | Steve | 222999 | MP | Gwalior | Ratan |

**Super keys**: {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip}…so on  
**Candidate Keys**: {emp\_id}  
**Non-prime attributes**: all attributes except emp\_id are non-prime as they are not part of any candidate keys.

Here, emp\_state, emp\_city&emp\_district dependent on emp\_zip. And, emp\_zip is dependent on emp\_id that makes non-prime attributes (emp\_state, emp\_city&emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

**employee table:**

|  |  |  |
| --- | --- | --- |
| emp\_id | emp\_name | emp\_zip |
| 1001 | John | 282005 |
| 1002 | Ajeet | 222008 |
| 1006 | Lora | 282007 |
| 1101 | Lilly | 292008 |
| 1201 | Steve | 222999 |

**employee\_zip table:**

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_zip | emp\_state | emp\_city | emp\_district |
| 282005 | UP | Agra | DayalBagh |
| 222008 | TN | Chennai | M-City |
| 282007 | TN | Chennai | Urrapakkam |
| 292008 | UK | Pauri | Bhagwan |
| 222999 | MP | Gwalior | Ratan |

## Boyce Codd normal form (BCNF)

It is an advance version of 3NF that’s why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every [functional dependency](https://beginnersbook.com/2015/04/functional-dependency-in-dbms/) X->Y, X should be the super key of the table.

**Example**: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| emp\_id | emp\_nationality | emp\_dept | dept\_type | dept\_no\_of\_emp |
| 1001 | Austrian | Production and planning | D001 | 200 |
| 1001 | Austrian | stores | D001 | 250 |
| 1002 | American | design and technical support | D134 | 100 |
| 1002 | American | Purchasing department | D134 | 600 |

**Functional dependencies in the table above**:  
emp\_id ->emp\_nationality  
emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate key**: {emp\_id, emp\_dept}

The table is not in BCNF as neither emp\_id nor emp\_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:  
**emp\_nationality table:**

|  |  |
| --- | --- |
| emp\_id | emp\_nationality |
| 1001 | Austrian |
| 1002 | American |

**emp\_dept table:**

|  |  |  |
| --- | --- | --- |
| emp\_dept | dept\_type | dept\_no\_of\_emp |
| Production and planning | D001 | 200 |
| stores | D001 | 250 |
| design and technical support | D134 | 100 |
| Purchasing department | D134 | 600 |

**emp\_dept\_mapping table:**

|  |  |
| --- | --- |
| emp\_id | emp\_dept |
| 1001 | Production and planning |
| 1001 | stores |
| 1002 | design and technical support |
| 1002 | Purchasing department |

**Functional dependencies**:  
emp\_id ->emp\_nationality  
emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate keys**:  
For first table: emp\_id  
For second table: emp\_dept  
For third table: {emp\_id, emp\_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

**Show Last Query Run (Laravel)**

1. **\**DB**::**enableQueryLog**();**
2. $list **=** **\**DB**::**table**(**"categories"**)->**get**();**
3. $query **=** **\**DB**::**getQueryLog**();**
4. print\_r**(**end**(**$query**));**

extracting integer/floating point from a string

(float) filter\_var($extracted\_price, FILTER\_SANITIZE\_NUMBER\_FLOAT, FILTER\_FLAG\_ALLOW\_FRACTION)