• ***<u>PHP-LARAVEL-USING FORMS AND GATHERING INPUT-INDUSTRY:-</u>***

1) Explain ORM.

<u>Ans:-</u>

- ***The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database. Each database table has a corresponding "Model" which is used to interact with that table.***
- ***Before getting started, be sure to configure a database connection in***
- ***config/database.php.***
- ***ORM stands for Object-Relational Mapping, which is a technique for mapping object-oriented systems to relational databases1. Laravel provides an ORM technique called Eloquent, which helps to interact with the database and perform database operations21. Eloquent maps the objects or models in the code to the database entities or tables.***

2) Do Crud using Eloquent Query.

<u>Ans:-</u>

- **Create (Insert):**

<u>Routs -> web.php :-</u>

```
Route::get('/signup',[customerController::class,'create']);
```

```
Route::post('/signup',[customerController::class,'store']);
```

<u>Then, customerController.php :-</u>

```
 public function create()
    {
        return view('viewdata/signup');
    }

public function store(Request $request)
    {
                        /**
                        $validated = $request->validate([
                        'name' => 'required|string|max:255',
                        'email' => 'required|email',
                        'contact_no' => 'required|digit|min:10|max:10',
                        'file' => 'required|image'
                        ]);
                        **/

                         $data=new customer;
                        $namemail=$data->name=$request->name;
                        $email=$data->email=$request->email;
                        $data->contact_no=$request->contact_no;
                        $data->gender=$request->gender;

                        //img upload
                        $file=$request->file('file');
                        $filename=time().'_img.'.$request->file('file')->getClientOriginalExtension();
                        $file->move('upload/customer/',$filename);   // use move for move image in
public/images
                        $data->file=$filename;

                        $data->save();
                        Alert::success('Congrats', 'You\'ve Successfully Registered');
                        return redirect()->back();
    }
```

- ## View (Show):

<u>Routs -> web.php :-</u>

```
Route::get('/view_cust',[customerController::class,'show']);
```

Then, customerController.php :-

```
public function show(customer $customer)
   {
      $data=customer::all();
      return view('viewdata/view_cust',['data_customer'=>$data]);
   }
```

- **Delete (Remove):**

Routs -> web.php :-

```
Route::get('/view_cust/{id}',[customerController::class,'destroy']);
```

Then, customerController.php :-

```
public function destroy(customer $customer,$id)
   {
      // get id data img
                  $data=customer::find($id);   //get only one data in string
                  $filename=$data->file;

                  // data delete with unlink image
                  customer::find($id)->delete();
                  if($filename!="")
                  {
                           unlink('upload/customer/'.$filename);
                  }
                  Alert::success('Congrats', 'You\'ve Successfully Deleted');
```

```
        return redirect()->back();
    }
```

- ## Update (Edit):

<u>Routs -> web.php :-</u>

```
Route::get('/editdata/{id}',[customerController::class,'edit']);
Route::post('/update/{id}',[customerController::class,'update']);
```

<u>Then, customerController.php :-</u>

```
 public function edit(customer $customer,$id)
    {
                        $data=customer::find($id);
                        return view('viewdata/editdata',['data'=>$data]);
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, customer $customer,$id)
    {
        $data=customer::find($id);
                        $data->name=$request->name;
                        $data->email=$request->email;
                        $data->gender=$request->gender;

                        //img upload
```

```
            if($request->hasFile('file'))
            {
                  $old_img=$data->file;
                  unlink('upload/customer/'.$old_img);

            $file=$request->file('file');
              $filename=time().'_img.'.$request->file('file')-
>getClientOriginalExtension();
              $file->move('upload/customer/',$filename);   // use move for move image in
public/images
              $data->file=$filename;
            }

            $data->update();
            Alert::success('Congrats', 'You\'ve Successfully Updated');
            return redirect('/view_cust');
  }
```

3) Explain - Eloquent Relationships.

<u>Ans:-</u>

- ***Database tables are often related to one another. For example, a blog post may have many comments or an order could be related to the user who placed it. Eloquent makes managing and working with these relationships easy, and supports a variety of common relationships:***
- *One To One*
- *One To Many*
- *Many To Many*

- *Has One Through*
- *Has Many Through*
- *One To One (Polymorphic)*
- *One To Many (Polymorphic)*
- *Many To Many (Polymorphic)*

• ***<u>One-to-One Relationship</u>:-***

A one-to-one relationship is a very basic type of database relationship. In a one-to-one relationship, each record in the table can be associated with only one record in another table.

Example: A user has one profile.

```
// User Model
public function profile()
{
   return $this->hasOne(Profile::class);
}
// Profile Model
public function user()
{
   return $this->belongsTo(User::class);
}
```

• ***<u>One-to-Many Relationship</u>:-***

In a one-to-many relationship, a single record in one table can be related to multiple records in another table.

Example: A user has many posts.

```
// User Model
public function posts()
{
   return $this->hasMany(Post::class);
}
// Post Model
public function user()
{
   return $this->belongsTo(User::class);
}
```

- ***Many-to-Many Relationship*:-**

In a many-to-many relationship, each record in one table can be related to multiple records in another table, and vice versa.

Example: Users can belong to many roles, and roles can have many users.

```
// User Model
public function roles()
{
   return $this->belongsToMany(Role::class);
}


// Role Model
public function users()
{
    return $this->belongsToMany(User::class);
}
```

- ***Has-Many-Through Relationship:***

This relationship is used to model distant relationships where a model has a relationship through another intermediate model.

Example: Country has many posts through users.

```
// Country Model
public function posts()
{
    return $this->hasManyThrough(Post::class, User::class);
}
```

- ***Polymorphic Relationships:-***

Polymorphic relationships allow a model to belong to more than one other type of model on a single association.

Example: Comments can belong to either posts or videos.

```
// Comment Model
public function commentable()
{
    return $this->morphTo();
}
// Post Model
public function comments()
{
    return $this->morphMany(Comment::class, 'commentable');
}
// Video Model
public function comments()
{
   return $this->morphMany(Comment::class, 'commentable');
}
```

4) *What is Eager Loading and lazy loading?*

Ans:-

- ***Eager Loading:-***

- **Eager loading is a technique used to retrieve a parent model with its related child models in a more efficient way, reducing the number of database queries. It is particularly useful when you need to load related data for multiple parent models in a single query.**
- **In Eager Loading, you specify which relationships you want to load when querying the parent model. Laravel then retrieves all the required data in a single query, improving performance.**
- **Example of eager loading in Laravel:**

```
// Load users with their associated posts
$users = User::with('posts')->get();
```

- ***Lazy Loading:-***

- **Lazy loading, on the other hand, is the default behavior in Eloquent. It loads related data only when you actually access** the related models. This means that when you fetch a parent model, Eloquent won't retrieve the related data until you explicitly request it. Lazy loading can lead to the N+1 query problem, where multiple queries are executed for related records.

- Example of lazy loading in Laravel:

```
// Load a user and then access their posts
$user = User::find(1);
$posts = $user->posts; // Posts are loaded when accessed
```

- Use Eager Loading when you know in advance that you'll need related data and want to minimize database queries. It's ideal for loading data for multiple records with their relationships.
- Use Lazy Loading when you want to load related data on a case-by-case basis, and you don't always need the related data, which can save database queries when accessing unrelated records.

5) Do Session for Employee Management System.

Ans:-

```
// Create session and store data

$request->session()->put('session_name',$data['empid']);

          session()->put('session_name',$value);

// Retrieving session

$request->session()->get('session_name')

   session()->get('session_name')
```

session('session_name')

Exa:

```
echo session('session_name')
<h1>Hello : {{session('session_name')}}</h1>
// Retrieving all session
$request->session()->all()
        session()->all()
//Determining session available or not
if($request->session()->has('session_name'))
{
}
or
if(session('session_name'))
{
}
```