Q-1: Explain ORM [Object Relational Model]

Object Relational Mapping (ORM) is a technique used in creating a "bridge" between object-oriented programs and, in most

cases, relational databases.

Put another way, you can see the ORM as the layer that connects object-oriented programming (OOP) to relational

databases.

When interacting with a database using OOP languages, you'll have to perform different operations like creating,

reading, updating, and deleting (CRUD) data from a database. By design, you use SQL for performing these operations in

relational databases.

While using SQL for this purpose isn't necessarily a bad idea, the ORM and ORM tools help simplify the interaction

between relational databases and different OOP languages.

Q-2: Do CRUD using Eloquent Query.

CREATE:

```
use App\Models\User;

// Create a new user

$user = User::create([

    'name' => 'John Doe',

    'email' => 'john@example.com',

    'password' => bcrypt('secret'),

]);
```

READ:

```
// Get all users

$users = User::all();

// Find a user by ID

$user = User::find(1);


// Get users with a specific condition

$activeUsers = User::where('active', 1)->get();
```

UPDATE:

```
// Find a user and update their information

$user = User::find(1);

$user->name = 'Jane Doe';

$user->save();
```

DELETE:

```
// Find a user and delete

$user = User::find(1);

$user->delete();
```

Or delete directly by ID:

```
User::destroy(1);
```

Q-3: Explain - Eloquent Relationships:

• Eloquent relationships are defined as methods on your Eloquent model classes. Since, like Eloquent models themselves,

relationships also serve as powerful query builders, defining relationships as methods provides powerful method chaining

and querying capabilities. For example, we may chain additional constraints on this posts relationship:

• One To One

• One To Many

• Many To Many

• Has Many Through

• Polymorphic Relations

Many To Many Polymorphic Relations

Q-4: What is Eager Loading and Lazy Loading?

Eager Loading:

When you are fetching any models from the database and then doing any type of processing on the model's relations, it's

important that you use eager loading. Eager loading is super simple using Laravel and basically prevents you from

encountering the N+1 problem with your data. This problem is caused by making N+1 queries to the database, where N is

the number of items being fetched from the database. To explain this better and give it some context, let's check out

the example below.

Imagine that you have two models (Comment and Author) with a one-to-one relationship between them. Now imagine that you

have 100 comments and you want to loop through each one of them and output the author's name.

Without eager loading, your code might look like this:

```
$comments = Comment::all();


foreach ($comments as $comment ) {

print_r($comment->author->name);

}
```

The code above would result in 101 database queries because it the results are "lazy loaded"! The first query would be

to fetch all of the comments. The other one hundred queries would come from getting the author’s name in each iteration

of the loop. Obviously, this can cause performance issues and slow down your application. So, how would we improve this?

By using eager loading, we could change the code to say:

```
$comments = Comment::with(‘authors’)->get();


foreach ($comments as $comment ) {

print_r($comment->author->name);

}
```

As you can see, this code looks almost the same and is still readable. By adding the ::with('authors') this will fetch

all of the comments and then make another query to fetch the authors at once. So, this means that we will have cut down

the query from 101 to 2!

Lazy Loading:

Lazy loading is the practice of delaying load or initialization of resources or objects until they’re actually needed to

improve performance and save system resources. For example, if a web page has an image that the user has to scroll down

to see, you can display a placeholder and lazy load the full image only when the user arrives to its location.

The benefits of lazy loading include:

Reduces initial load time – Lazy loading a webpage reduces page weight, allowing for a quicker page load time.

Bandwidth conservation – Lazy loading conserves bandwidth by delivering content to users only if it’s requested.

System resource conservation – Lazy loading conserves both server and client resources, because only some of the images,

JavaScript and other code actually needs to be rendered or executed