- How to do config database in Laravel
  - ✓ Database configuration for database.php in config folder directory or .env file for database configuration.
- Call MySQLi Store Procedure from Laravel
  - ✓ Call stored_procedure(:data); COMMIT;
  - ✓ $bind = [
  - ✓ 'data' => $ data
  - ✓];
  - ✓ DB::statement($query, $bind);
- Explain ORM
  - ✓ ORM stands for Object Relational Mapper. The Eloquent ORM included Laravel provides a beautiful, simple Active Record implementation for working with database. Each database table has a corresponding "Model" which is used to interact with a that table.
- Explain - Eloquent Relationships?
  - ✓ Eloquent relationship are defined as method on eloquent model classes.
    - Types of Relationship
      - One to One
      - One to Many
      - Many to Many
      - Has One of Many
      - Has One Through
      - Has Many Through
- What is Eager Loading and lazy loading?
  - ✓ Eager loading and lazy loading are two strategies for managing data retrieval in the context of database systems, particularly in Object-Relational Mapping frameworks. They help control when and how related data is fetched from the database, which can impact performance and resource usage.
  - ✓ Eager Loading:
    - Eager loading retrieves all related data in a single query includes joins or other mechanisms to fetch associated entities along with the primary entity. As a result, all the necessary data is available immediately, avoiding the need for additional queries later.
    - ❖ Advantages:
      - Reduces the number of database queries, which can improve performance by reducing round-trip time.
      - Ensures that all required data is available when needed, which can be useful for displaying complete information at once.
    - ❖ Disadvantages:
      - Can lead to fetching more data then necessary, which might impact performance, especially if the related data is large or complex.
      - May increase memory usage since all data is loaded into memory at once.
    - ❖ Example:
      - SELECT authors.*, books.* from authors left join books ON authors.id = books.author_id WHERE authors.id = 1

- ✓ Lazy Loading:
  - Lazy loading delays the retrieval of related data until it is specifically requested. Initially, only the main entity is loaded and related data is fetched as needed. This approach is particularly useful when the related data is large or might not be required immediately.
  - ❖ Advantages:
    - Can save memory and reduce initial load time since only the necessary data is fetched.
    - Precents loading unnecessary data, which can ben beneficial if not all related data is needed.
  - ❖ Disadvantages:
    - Can lead to the N+1 query problem, where multiple database queries are issued, potentially one for each related entity, which can degrade performance.
  - ❖ Example:
    - SELECT * FROM authors WHERE id = 1;
    - SELECT * FROM books WHERE author_id = 1;

- How to pass Multiple Variable in route?
  - ✓ Route::get('employee/{variable1}/{{variable2}}/show',[EmployeeController::class,'show']);
- How to pass variable which can be null in Route?
  - ✓ Route::get('user/{userId?}',function(?string $userId =null){
    - Return $userId;
  - ✓ });
- Create custom auto using middleware?
  - ✓ Php artisan make:middleware MiddlewareName
- Generate Resource Controller for employee
  - ✓ Route::resource('employee',EmployeeController::class);
- In Employee Controllers action Call Middleware?
  - ✓ Public function __construct(){
    - $this->middleware('auth');
  - ✓ }
- How to remove route caching?
  - ✓ Using Artisan Command on Terminal
    - Php artisan route:cache

- Create Custom Macro For search User
  - ✓ User::macro('search', function ($keyword) { return $this->where('name', 'like', "%$keyword%") ->orWhere('email', 'like', "%$keyword%") ->get(); });
  - ✓ $users = User::search('John')->get();