**SQl / Database**

|  |
| --- |
| **Regarding notifications** |

|  |
| --- |
| **Regarding notifications** |

|  |
| --- |
| **Regarding notifications** |

|  |
| --- |
| **Regarding Relationships** |

**1. Define Relationships in Models**

**In AssignmentTeamMapping.php**

php

CopyEdit

class AssignmentTeamMapping extends Model

{

protected $table = 'assignmentteammappings';

public function assignmentMapping()

{

return $this->belongsTo(AssignmentMapping::class, 'assignmentmapping\_id');

}

public function teamMember()

{

return $this->belongsTo(TeamMember::class, 'teammember\_id');

}

}

**In AssignmentMapping.php**

php

CopyEdit

class AssignmentMapping extends Model

{

protected $table = 'assignmentmappings';

public function assignmentTeamMappings()

{

return $this->hasMany(AssignmentTeamMapping::class, 'assignmentmapping\_id');

}

}

**In TeamMember.php**

php

CopyEdit

class TeamMember extends Model

{

protected $table = 'teammembers';

public function title()

{

return $this->belongsTo(Title::class, 'title\_id');

}

public function role()

{

return $this->belongsTo(Role::class, 'role\_id');

}

}

**2. Use Relationships in the Query**

php

CopyEdit

$teammemberDatas = AssignmentTeamMapping::with([

'assignmentMapping' => function ($query) use ($id) {

$query->where('assignmentgenerate\_id', $id);

},

'teamMember.title',

'teamMember.role'

])

->whereNotNull('id')

->where('type', '!=', 1)

->orderByDesc('id')

->get()

->map(function ($teamMapping) {

return [

'teammember' => $teamMapping->teamMember,

'rolename' => $teamMapping->teamMember->role->rolename ?? null,

'title' => $teamMapping->teamMember->title->title ?? null,

'assignmentteammappingsId' => $teamMapping->id,

'assignmentteammappingsStatus' => $teamMapping->status,

'type' => $teamMapping->type,

'teamhour' => $teamMapping->teamhour,

'viewerteam' => $teamMapping->viewerteam,

'assignmentgenerateid' => $teamMapping->assignmentMapping->assignmentgenerate\_id ?? null,

'leadpartner' => $teamMapping->assignmentMapping->leadpartner ?? null,

];

});

dd($teammemberDatas);

**Benefits of Using Relationships**

✅ **More Readable & Maintainable**  
✅ **Easier to Modify** (No need to change multiple queries)  
✅ **Faster Development** with Laravel Eloquent  
✅ **Better Performance** (Laravel optimizes queries with eager loading)

Now, instead of manually writing joins, Laravel **automatically manages relationships**, making your code **cleaner and optimized**. 🚀

|  |
| --- |
| **Difference Between JOIN and LEFT JOIN in SQL / regarding join / regarding left join** |

**Difference Between JOIN and LEFT JOIN in SQL**

JOIN and LEFT JOIN are used to combine records from multiple tables, but they behave differently when there are no matching records.

**1. JOIN (INNER JOIN)**

* Returns **only matching records** from both tables.
* If there is **no match**, the row is **excluded** from the result.

🔹 **Example:**

sql

CopyEdit

SELECT employees.id, employees.name, departments.department\_name

FROM employees

JOIN departments ON employees.department\_id = departments.id;

✅ **Result:**

* Only employees who belong to a department will be included.
* Employees without a department will be excluded.

**2. LEFT JOIN (LEFT OUTER JOIN)**

* Returns **all records from the left table** and **matching records from the right table**.
* If there is **no match**, NULL values are returned for the right table's columns.

🔹 **Example:**

sql

CopyEdit

SELECT employees.id, employees.name, departments.department\_name

FROM employees

LEFT JOIN departments ON employees.department\_id = departments.id;

✅ **Result:**

* All employees will be shown.
* If an employee has **no department**, department\_name will be NULL.

**Key Differences:**

| **Feature** | **JOIN (INNER JOIN)** | **LEFT JOIN** |
| --- | --- | --- |
| Matching Rows Required? | ✅ Yes | ❌ No |
| Unmatched Rows Included? | ❌ No | ✅ Yes (with NULL values) |
| Performance | 🚀 Faster (Less data) | 🐢 Slower (More data) |

**When to Use?**

* **Use JOIN** when you only need records that have a match in both tables.
* **Use LEFT JOIN** when you want **all records from the left table**, even if there’s no match.

**Example Scenario in Your Query:**

**Before Optimization**

php

CopyEdit

DB::table('assignmentmappings')

->leftJoin('assignmentteammappings', 'assignmentteammappings.assignmentmapping\_id', '=', 'assignmentmappings.id')

🔹 **Why LEFT JOIN?**

* Because we want all assignmentmappings, even if there’s **no matching assignmentteammappings**.

**After Optimization**

php

CopyEdit

DB::table('assignmentteammappings')

->join('assignmentmappings', 'assignmentteammappings.assignmentmapping\_id', '=', 'assignmentmappings.id')

🔹 **Why JOIN?**

* Because we only need assignmentteammappings **that have a matching assignmentmapping\_id**.

🚀 **Conclusion:**

* JOIN is used when we only need **matching records**.
* LEFT JOIN is used when we need **all records from the left table, even without matches**.