Método	Descripción
`$table->id();`	Crea una columna de ID autoincremental (`BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY`).
`$table->bigIncrements('id');`	Lo mismo que `$table->id()`.
`$table->integer('edad');`	Entero normal (`INT`).
`$table->bigInteger('likes');`	Entero grande (`BIGINT`).
`$table->unsignedBigInteger('user_id');`	Entero sin signo (`BIGINT UNSIGNED`) usado para claves foráneas.
`$table->string('nombre', 100);`	Texto corto con un límite de caracteres (`VARCHAR`).
`$table->text('descripcion');`	Texto largo (`TEXT`).
`$table->boolean('activo');`	Booleano (`TINYINT(1)`).
`$table->decimal('precio', 8, 2);`	Número decimal (`DECIMAL(8,2)`).
`$table->float('altura', 8, 2);`	Número flotante (`FLOAT`).
`$table->date('fecha_nacimiento');`	Fecha (`DATE`).
`$table->time('hora');`	Hora (`TIME`).
`$table->dateTime('evento');`	Fecha y hora (`DATETIME`).
`$table->json('metadata');`	JSON (`JSON`).
`$table->binary('archivo');`	Almacenar archivos binarios (`BLOB`).
`$table->timestamps();`	Agrega `created_at` y `updated_at`.
`$table->softDeletes();`	Agrega `deleted_at` para "borrado lógico".

📌 Restricciones en Laravel

Las restricciones aseguran la integridad de los datos en la base de datos.

Método	Descripción
`$table->unique('email');`	Hace que la columna sea única.
`$table->nullable();`	Permite valores `NULL`.
`$table->default(0);`	Asigna un valor por defecto si no se proporciona.
`$table->index('nombre');`	Agrega un índice a la columna para mejorar la búsqueda.
`$table->foreign('user_id')->references('id')->on('users');`	Define una clave foránea.
`$table->foreignId('user_id')->constrained();`	Lo mismo que la línea anterior.
`$table->foreignId('user_id')->constrained()->onDelete('cascade');`	Si se borra el usuario, también se borra el registro relacionado.
`$table->check('edad > 18');`	Solo permite valores que cumplan la condición (puede que no funcione en todas las bases de datos).

📌 ¿Qué pasa si no se cumplen las restricciones?

🔴 **Si una restricción no se cumple, se genera un error en la base de datos.**

Restricción	Error si no se cumple
`unique('email')`	Error si intentas insertar un email repetido.
`nullable(false)` (Por defecto)	Error si intentas insertar un `NULL` en una columna que no lo permite.
`default(0)`	No hay error, simplemente se usa el valor predeterminado.
`foreign()->onDelete('restrict')` (Por defecto)	Error si intentas borrar un registro que tiene claves foráneas en otras tablas.
`foreign()->onDelete('cascade')`	Se borran automáticamente los registros relacionados.

`check('edad > 18')`	Error si intentas insertar un valor menor o igual a 18.

```
use Illuminate\Http\Request;
use Illuminate\Validation\ValidationException;

class UserController extends Controller
{
    public function store(Request $request)
    {
        try {
            $validatedData = $request->validate([
                'email' => 'required|email|unique:users,email',
                'edad' => 'required|integer|min:18|max:99',
            ]);

            return response()->json(['message' => 'Datos válidos'], 200);
        } catch (ValidationException $e) {
            return response()->json([
                'errors' => $e->errors(), // Devuelve los errores específicos
                'message' => 'Error de validación'
            ], 422);
        }
    }
}
```

1️⃣ Validaciones básicas de datos

Regla	Descripción	Ejemplo
`required`	Campo obligatorio	`'nombre' => 'required'`
`nullable`	Permite valores `null`	`'edad' => 'nullable`
`string`	Debe ser texto	`'nombre' => 'string'`

`integer`	Debe ser un número entero	`'edad' => 'integer'`
`numeric`	Debe ser un número (entero o decimal)	`'precio' => 'numeric'`
`boolean`	Debe ser `true` o `false`	`'activo' => 'boolean'`

2️⃣ Validaciones de longitud y tamaño

Regla	Descripción	Ejemplo
`min:X`	Longitud mínima o valor mínimo	`'nombre' => 'min:3'`
`max:X`	Longitud máxima o valor máximo	`'nombre' => 'max:50'`
`between:X,Y`	Debe estar entre un rango	`'edad' => 'between:18,99'`
`size:X`	Debe tener exactamente `X` caracteres o peso en archivos	`'codigo' => 'size:10'`

3️⃣ Validaciones de texto y formato

Regla	Descripción	Ejemplo
`email`	Debe ser un email válido	`'email' => 'email'`
`regex:/expresión_regular/`	Validar con expresión regular	`'telefono' => 'regex:/^[0-9]{10}$/'`
`alpha`	Solo letras	`'nombre' => 'alpha'`
`alpha_dash`	Letras, números, guiones y guiones bajos	`'username' => 'alpha_dash'`
`alpha_num`	Letras y números	`'codigo' => 'alpha_num'`

4️⃣ Validaciones de archivos e imágenes

Regla	Descripción	Ejemplo
`file`	Debe ser un archivo	`'archivo' => 'file'`
`image`	Debe ser una imagen (jpg, png, gif, etc.)	`'foto' => 'image'`
`mimes:x,y,z`	Extensiones permitidas	`'foto' => 'mimes:jpg,png'`
`mimetypes:x/y`	Tipo MIME permitido	`'documento' => 'mimetypes:application/pdf'`
`max:X`	Tamaño máximo en KB	`'archivo' => 'max:2048'`

5️⃣ Validaciones de fechas

Regla	Descripción	Ejemplo
`date`	Debe ser una fecha válida	`'fecha' => 'date'`
`date_format:Y-m-d`	Formato específico	`'fecha' => 'date_format:Y-m-d'`
`after:fecha`	Debe ser después de una fecha	`'fecha' => 'after:tomorrow'`
`before:fecha`	Debe ser antes de una fecha	`'fecha' => 'before:2025-01-01'`

6️⃣ Validaciones de relaciones en BD

Regla	Descripción	Ejemplo
`exists:tabla,columna`	Debe existir en la BD	`'user_id' => 'exists:users,id'`
`unique:tabla,columna`	Debe ser único en la BD	`'email' => 'unique:users,email'`

1️⃣ Obtener productos con precio menor a un valor

📌 Ejemplo con Eloquent

php
CopiarEditar
```
$productos = Product::where('precio', '<', 100)->get();
```

📌 Ejemplo con Query Builder

php
CopiarEditar
```
$productos = DB::table('products')->where('precio', '<',
100)->get();
```

2️⃣ Obtener productos con precio entre 50 y 200

📌 Eloquent

php
CopiarEditar
```
$productos = Product::whereBetween('precio', [50, 200])->get();
```

📌 Query Builder

php
CopiarEditar
```
$productos = DB::table('products')->whereBetween('precio', [50,
200])->get();
```

3️⃣ Obtener productos con precio mayor a 100 y en stock

📌 Eloquent

php
CopiarEditar
```
$productos = Product::where('precio', '>', 100)
    ->where('stock', '>', 0)
```

```
    ->get();
```

📌 **Query Builder**

php
CopiarEditar
```
$productos = DB::table('products')
    ->where('precio', '>', 100)
    ->where('stock', '>', 0)
    ->get();
```

4️⃣ Obtener productos donde el nombre contenga "laptop"

📌 **Eloquent**

php
CopiarEditar
```
$productos = Product::where('nombre', 'like', '%laptop%')->get();
```

📌 **Query Builder**

php
CopiarEditar
```
$productos = DB::table('products')->where('nombre', 'like',
'%laptop%')->get();
```

5️⃣ Obtener productos ordenados por precio descendente

📌 **Eloquent**

php
CopiarEditar
```
$productos = Product::orderBy('precio', 'desc')->get();
```

📌 **Query Builder**

php
CopiarEditar

```
$productos = DB::table('products')->orderBy('precio',
'desc')->get();
```

6️⃣ Obtener productos y limitar resultados (paginación)

📌 **Eloquent**

php
CopiarEditar

```
$productos = Product::limit(10)->get(); // Solo 10 productos
$productos = Product::paginate(5); // Paginación de 5 por página
```

📌 **Query Builder**

php
CopiarEditar

```
$productos = DB::table('products')->limit(10)->get();
```

7️⃣ Obtener un producto específico por ID

📌 **Eloquent**

php
CopiarEditar

```
$producto = Product::find(1);
```

📌 **Query Builder**

php
CopiarEditar

```
$producto = DB::table('products')->where('id', 1)->first();
```

8️⃣ Consultas condicionales (dependiendo de algo)

Si se pasa un precio, filtrar por precio; si no, traer todos

php
CopiarEditar

```php
$precio = request('precio'); // Supongamos que viene del request

$productos = Product::when($precio, function ($query, $precio) {
    return $query->where('precio', '<', $precio);
})->get();
```

9️⃣ Consultar con relaciones (productos de una categoría específica)

📌 Eloquent (usando `hasMany` en el modelo `Category`)

php
CopiarEditar

```php
$categoria = Category::where('nombre', 'Electrónica')->first();
$productos = $categoria->productos; // Relación definida en el modelo
```

📌 Query Builder (join)

php
CopiarEditar

```php
$productos = DB::table('products')
    ->join('categories', 'products.categoria_id', '=', 'categories.id')
    ->where('categories.nombre', 'Electrónica')
    ->select('products.*')
    ->get();
```

🔹 Resumen

Consulta	Eloquent	Query Builder
Menor que un precio	`Product::where('precio', '<', 100)->get();`	`DB::table('products')->where('precio', '<', 100)->get();`

Rango de precios	`whereBetween('precio', [50, 200])`	`whereBetween('precio', [50, 200])`
Filtrar por nombre	`where('nombre', 'like', '%laptop%')`	`where('nombre', 'like', '%laptop%')`
Ordenar por precio	`orderBy('precio', 'desc')`	`orderBy('precio', 'desc')`
Paginación	`paginate(5)`	❌ (No soportado)
Un solo producto	`find(1)`	`where('id', 1)->first();`
Consulta con relaciones	`$categoria->productos;`	`join()`