Siddhesh Prabhugaonkar
Microsoft Certified Trainer
siddheshpg@azureauthority.in

In NestJS, if the handler returns (even an exception object), the request is considered successful and the default POST status `201` is sent. You must **throw** the exception (or let your service throw it) so Nest's global exception filter sets HTTP `409`.

Option A — Let the service throw (recommended)

Controller

```
// employees.controller.ts
import { Controller, Post, Body } from '@nestjs/common';
import { EmployeesService } from './employees.service';
import { CreateEmployeeDto } from './dto/create-employee.dto';

@Controller('employees')
export class EmployeesController {
  constructor(private readonly employeesService: EmployeesService) {}

  @Post()
  async create(@Body() dto: CreateEmployeeDto) {
    // No try/catch needed if the service throws proper Nest exceptions
    return await this.employeesService.create(dto);
  }
}
```

Service (Sequelize + Oracle example)

```
// employees.service.ts
import { Injectable, ConflictException } from '@nestjs/common';
// import { UniqueConstraintError } from 'sequelize'; // if you want the
class
import { InjectModel } from '@nestjs/sequelize';
import { Employee } from './entities/employee.entity.js';

@Injectable()
export class EmployeesService {
  constructor(@InjectModel(Employee) private model: typeof Employee) {}

  async create(dto: { email: string; name: string; departmentId?: number })
{
    try {
      // optional pre-check (fewer DB errors, clearer message)
      const existing = await this.model.findOne({ where: { email: dto.email
} });
      if (existing) {
        throw new ConflictException('Employee already exists');
```

```
      }
      const employee = await this.model.create(dto);
      return employee; // Nest will send 201 Created
    } catch (err) {
      // Map DB unique violations to 409
      const isSequelizeUnique =
        err?.name === 'SequelizeUniqueConstraintError' ||
        err?.original?.code === 'ORA-00001'; // Oracle unique violation

      if (isSequelizeUnique) {
        throw new ConflictException('Employee already exists');
      }
      throw err; // let other errors bubble up as 500 or as thrown
    }
  }
}
```

Option B — Catch in controller and throw

If you must handle it in the controller:

```
import { ConflictException, Post, Body, Controller } from '@nestjs/common';

@Post()
async create(@Body() dto: CreateEmployeeDto) {
  try {
    return await this.employeesService.create(dto);
  } catch (error) {
    if (error?.message === 'Employee already exists') {
      throw new ConflictException(error.message); // <-- THROW, not return
    }
    throw error;
  }
}
```

Why your code sent 201

- `@Post()` defaults to 201 on success.
- `return new ConflictException(...)` returns a plain object; Nest thinks the handler succeeded and sends 201 with that object as the body.