**LMS API - LINQ Queries Documentation**

---

**AssignmentsController**

1. Get Assignments by Course

```
var assignments = _context.Assignments
    .Where(a => a.CourseId == courseId)
    .OrderBy(a => a.Title.ToLower())
    .ToList();
```

**Purpose:** Retrieve all assignments for a specific course, sorted alphabetically by title (case-insensitive). **Used in:**

GetByCourse(int courseId) endpoint **Returns:** List of Assignment objects

---

2. Get Instructor's Assignments

```
var assignments = _context.Assignments
    .Where(a => _context.Courses.Any(c => c.Id == a.CourseId && c.InstructorId == instructorId))
    .OrderBy(a => a.Title)
    .ToList();
```

**Purpose:** Retrieve all assignments created by a specific instructor across all their courses. **Used in:**

GetMyAssignments() endpoint **Returns:** List of Assignment objects **Note:** Uses nested query to filter by instructor ownership

---

**SubmissionsController**

3. Get Student's Submissions

```
var submissions = _context.Submissions
    .Where(s => s.StudentId == studentId)
    .OrderByDescending(s => s.SubmittedOn)
    .ToList();
```

**Purpose:** Retrieve all submissions made by a specific student, sorted by most recent first. **Used in:**

MySubmissions() endpoint **Returns:** List of Submission objects

---

4. Get Submissions by Assignment (with Student Info)

```
var submissions = _context.Submissions
```

```
.Include(s => s.Student)
.Where(s => s.AssignmentId == assignmentId)
.OrderByDescending(s => s.SubmittedOn)
.Select(s => new
{
    s.Id,
    s.AssignmentId,
    s.StudentId,
    StudentName = s.Student.UserName,
    s.Content,
    s.SubmittedOn,
    s.Grade,
    s.Feedback,
    s.Status
})
.ToList();
```

**Purpose:** Retrieve all submissions for a specific assignment with student details for instructor grading. **Used in:**

GetSubmissionsByAssignment(int assignmentId) endpoint **Returns:** Anonymous object with submission and student information **Note:** Uses Include for eager loading and Select for projection

---

**CoursesController**

5. Get All Courses

```
var courses = _context.Courses
    .Include(c => c.Modules)
        .ThenInclude(m => m.Lessons)
    .ToList();
```

**Purpose:** Retrieve all courses with their complete module and lesson hierarchy. **Used in:** GetAll() endpoint **Returns:** List of Course objects with nested Modules and Lessons **Note:** Uses ThenInclude for multi-level eager loading

---

6. Get Course by ID

```
var course = _context.Courses
    .Include(c => c.Modules)
        .ThenInclude(m => m.Lessons)
    .FirstOrDefault(c => c.Id == id);
```

**Purpose:** Retrieve a single course with all its modules and lessons. **Used in:** GetById(int id) endpoint **Returns:** Single Course object or null **Note:** Loads complete course structure for display

---

7. Get Instructor's Courses

```
var courses = _context.Courses
    .Where(c => c.InstructorId == instructorId)
    .Include(c => c.Modules)
        .ThenInclude(m => m.Lessons)
    .ToList();
```

**Purpose:** Retrieve all courses created by a specific instructor with full content. **Used in:** GetMyCourses() endpoint **Returns:** List of Course objects **Note:** Filtered by instructor ownership

---

8. Get Published Courses

```
var courses = _context.Courses
    .Where(c => c.IsPublished && c.Status == CourseStatus.Approved)
    .Include(c => c.Modules)
        .ThenInclude(m => m.Lessons)
    .ToList();
```

**Purpose:** Retrieve all approved and published courses available for student enrollment. **Used in:** Browse/catalog functionality **Returns:** List of publicly available courses

---

**EnrollmentsController**

9. Get Student's Enrollments

```
var enrollments = _context.Enrollments
    .Include(e => e.Course)
    .Where(e => e.StudentId == studentId)
    .ToList();
```

**Purpose:** Retrieve all course enrollments for a specific student with course details. **Used in:** MyEnrollments() endpoint **Returns:** List of Enrollment objects with Course information **Note:** Used for student dashboard

---

10. Get Enrollments by Status

```
var enrollments = _context.Enrollments
    .Include(e => e.Course)
    .Where(e => e.Status == status)
    .OrderByDescending(e => e.EnrolledOn)
    .ToList();
```

**Purpose:** Retrieve enrollments filtered by approval status (Pending/Approved/Rejected). **Used in:** Admin enrollment management **Returns:** List of Enrollment objects sorted by enrollment date

---

11. Check Enrollment Status

```
var enrollment = _context.Enrollments
    .FirstOrDefault(e => e.StudentId == studentId && e.CourseId == courseId);
```

**Purpose:** Check if a student is enrolled in a specific course. **Used in:** Enrollment validation **Returns:** Single Enrollment object or null

---

**ProgressController**

12. Get Student Progress

```
var progress = _context.Progresses
    .Where(p => p.StudentId == studentId)
    .ToList();
```

**Purpose:** Retrieve all progress records for a student across all enrolled courses. **Used in:** Dashboard statistics calculation **Returns:** List of Progress objects

---

13. Get Course Progress for Student

```
var progress = _context.Progresses
    .FirstOrDefault(p => p.StudentId == studentId && p.CourseId == courseId);
```

**Purpose:** Get progress for a specific student in a specific course. **Used in:** Course detail page progress display **Returns:** Single Progress object or null

---

14. Calculate Total Lessons (Dynamic)

```
var totalLessons = _context.Modules
    .Where(m => m.CourseId == courseId)
    .SelectMany(m => m.Lessons)
    .Count();
```

**Purpose:** Dynamically count total lessons in a course from the database. **Used in:** Progress calculation **Returns:** Integer count of lessons **Note:** Uses SelectMany to flatten module-lesson hierarchy

---

**AuthController / UserManagementController**

15. Get Pending Users

```
var users = _context.Users
    .Where(u => u.Status == UserStatus.Pending)
    .OrderBy(u => u.Email)
    .ToList();
```

**Purpose:** Retrieve all users awaiting admin approval. **Used in:** Admin user approval workflow **Returns:** List of ApplicationUser objects

---

16. Find User by Email

```
var user = _context.Users
    .FirstOrDefault(u => u.Email == email);
```

**Purpose:** Look up a user by their email address. **Used in:** Login, password reset, user lookup **Returns:** Single ApplicationUser or null

---

**ReportsController**

17. Enrollment Report

```
var enrollments = _context.Enrollments
    .Include(e => e.Course)
    .GroupBy(e => e.CourseId)
    .Select(g => new
    {
        CourseId = g.Key,
```

```
    CourseName = g.First().Course.Title,
    TotalEnrollments = g.Count(),
    ApprovedEnrollments = g.Count(e => e.Status == "Approved"),
    PendingEnrollments = g.Count(e => e.Status == "Pending")
})
.ToList();
```

**Purpose:** Generate enrollment statistics grouped by course. **Used in:** Admin reports dashboard **Returns:** Anonymous objects with enrollment counts **Note:** Uses GroupBy and aggregate functions

---

18. Student Progress Report

```
var progressData = _context.Progresses
    .Include(p => p.Course)
    .Where(p => p.StudentId == studentId)
    .Select(p => new
    {
        p.CourseId,
        CourseName = p.Course.Title,
        p.CompletedLessons,
        p.TotalLessons,
        p.CompletionPercentage
    })
    .ToList();
```

**Purpose:** Generate progress report for a specific student across all courses. **Used in:** Student progress tracking **Returns:** Progress statistics per course

---

19. Submission Statistics

```
var stats = _context.Submissions
    .GroupBy(s => s.AssignmentId)
    .Select(g => new
    {
        AssignmentId = g.Key,
```

```
    TotalSubmissions = g.Count(),
    GradedSubmissions = g.Count(s => s.Status == "Graded"),
    AverageGrade = g.Where(s => s.Grade.HasValue).Average(s => s.Grade.Value)
})
.ToList();
```

**Purpose:** Calculate submission and grading statistics per assignment. **Used in:** Instructor analytics **Returns:** Submission metrics with average grades **Note:** Uses conditional aggregation with Where inside Average