Module 7

What to learn

- LINQ
  - Select
  - All/Any
  - Contains
  - Aggregate
  - Average
  - Count
  - Single/SingleorDefault
  - DefaultIfEmpty
- Intersect/Union/Take,TakeWhile
- Deferred Execution
- Immediate Execution
- Let Keyword
- Into Keyword

Practice Exercise

Practice 1

Select

Use Select to extract only the names of employees from a list of Employee objects and display them.
Write a LINQ query to create a new list of Student objects with only Name and Age properties from the original collection.
Use Select to convert a list of product prices from USD to EUR using a conversion rate.
Write a LINQ query to transform a list of dates into their corresponding day names (e.g., "Monday", "Tuesday").
Use Select to create a list of email addresses from a collection of user objects.

Practice 2

All/Any

Use All to check if all products in a list are in stock.
Use Any to verify if there is at least one student with a grade above 90 in a list of scores.
Write a LINQ query to check if all employees in a department have salaries above a certain threshold.
Use Any to determine if there are pending orders in an order list with a status of "Pending".
Write a query to check if all courses have at least one student enrolled.

Practice 3

Contains

Use Contains to check if a specific product ID exists in a list of product IDs.

Write a LINQ query to determine if a list of names contains a specific name entered by the user.
Use Contains to filter orders that match specific order IDs from a predefined list.
Write a query to check if a list of books includes a specific title using Contains.
Use Contains to find if a list of countries includes a specific country entered by the user

Practice 4

Aggregate

Use Aggregate to concatenate all product names from a list into a single comma-separated string.
Write a query to calculate the product of all numbers in a list using Aggregate.
Use Aggregate to find the longest word in a list of strings.
Implement a LINQ query to compute the cumulative total of sales from a list of transactions using Aggregate.
Use Aggregate to reverse a string by iterating through its characters.

Practice 5

Average

Use Average to calculate the average age of employees in a department.
Write a LINQ query to find the average grade of students in a class.
Calculate the average price of all products in a catalog using LINQ.
Write a query to compute the average transaction amount from a list of sales.
Use LINQ to find the average word length in a collection of sentences.

Practice 6

Count

Use Count to find the number of products in a list with a price greater than $50.
Write a LINQ query to count the number of students enrolled in a specific course.
Use Count to find how many numbers in a list are divisible by 3.
Implement a query to count the number of orders placed in the last 7 days.
Use Count to determine how many employees in a company belong to a specific department.

Practice 7

Single/SingleOrDefault

Use Single to find the product with a unique ID from a product list. Throw an exception if not found.
Use SingleOrDefault to retrieve a user from a list by their unique username. Return null if no match is found.
Write a query to find the student with a unique roll number from a list of students.
Use SingleOrDefault to retrieve a transaction from a list with a specific transaction ID. Ensure the code handles cases with no match.
Use Single to find the book with a unique ISBN from a catalog.

Practice 8

DefaultIfEmpty

Use DefaultIfEmpty to handle cases where a query to find high-priority tasks returns no results.

Write a LINQ query to find products in a specific category. Return a default message if the category is empty.
Use DefaultIfEmpty to ensure a collection of employees in a department always returns at least one placeholder.
Write a LINQ query to retrieve all orders for a specific date and return "No Orders Found" if none exist.
Use DefaultIfEmpty to provide a default value when filtering a list of customers.

Practice 9

Intersect/Union/Take/TakeWhile

Use Union to combine two lists of integers, removing duplicates.
Use Intersect to find common employees between two departments.
Write a query using Take to retrieve the first 5 students from a list of grades.
Use TakeWhile to retrieve scores from a list until a score below 50 is encountered.
Use Union to merge two lists of courses and eliminate duplicates.

Practice 10

Deferred Execution

Write a LINQ query to demonstrate deferred execution by creating a query and executing it only after modifying the source collection.
Implement a program that demonstrates deferred execution using a list of integers and a Where clause.
Show how deferred execution works by iterating through a collection after adding new elements to it.
Write a query that retrieves all even numbers from a list and demonstrate deferred execution by modifying the list after the query is defined.
Use deferred execution to filter a collection and only execute the filter when iterated.

Practice 11

Immediate Execution

Use ToList to force immediate execution of a query that filters products by price.
Write a LINQ query that calculates the sum of a list of numbers using immediate execution.
Demonstrate immediate execution by converting a query result to an array using ToArray.
Use ToDictionary to immediately execute a query and store the result as a dictionary.
Compare deferred and immediate execution using a query that uses ToList.

Practice 12

Let Keyword

Use the let keyword to store the result of a calculation (e.g., square of a number) in a LINQ query.
Write a query that uses let to define an intermediate variable for the length of strings and filter strings by their lengths.
Use let to calculate the discounted price of products in a catalog and filter products with discounts above $10.
Write a LINQ query to split names into first and last names using let and filter by last name length.
Use let to simplify a LINQ query that filters a collection based on a complex condition.

Practice 13

Into Keyword

Use the into keyword to create a continuation query that filters employees by department and then sorts them by age.
Write a query that groups students by their grades and continues with an additional filter using into.
Use into to chain queries that first select products by category and then sort them by price.
Write a LINQ query that uses into to filter words by length and then group them by their first letter.
Implement a program that uses into to refine a group of customers by country and filter those with more than 5 orders.

Assignment Exercise

Assignment 1
Objective:
This assignment focuses on utilizing LINQ concepts such as **Select**, **All/Any**, **Contains**, **Aggregate**, **Average**, **Count**, **Single/SingleOrDefault**, **DefaultIfEmpty**, **Intersect/Union/Take/TakeWhile**, **Deferred Execution**, **Immediate Execution**, **Let Keyword**, and **Into Keyword** to solve real-world scenarios effectively.

Scenario

You are tasked with creating a **Business Analytics Tool** for a retail company. The tool will analyze customer data, sales data, and product inventory using LINQ to provide actionable insights.

Requirements

Data Models

Create the following classes:

Customer

- CustomerId (int)
- Name (string)
- City (string)
- IsPremium (bool)

Product

- ProductId (int)
- ProductName (string)
- Category (string)
- Price (decimal)
- Stock (int)

Order

- OrderId (int)
- CustomerId (int)

ProductId (int)
Quantity (int)
OrderDate (DateTime)

Tasks

Task 1: Basic LINQ Queries

Use Select to create a list of product names and their prices.
Use All to check if all customers are from the same city.
Use Any to verify if there is at least one premium customer in the dataset.
Use Contains to check if a specific product exists in the product list by name.
Use Aggregate to concatenate all customer names into a single string, separated by commas.

Task 2: Statistical Analysis

Use Average to calculate the average price of products in a specific category.
Use Count to find how many orders were placed in the last 30 days.
Use SingleOrDefault to retrieve a product with a specific product ID, and handle cases where the product is not found.
Use DefaultIfEmpty to handle a scenario where there are no orders for a specific date. Return a default message if the list is empty.
Use Take to retrieve the first 5 customers from the customer list.

Task 3: Combining and Filtering Data

Use Intersect to find common products sold between two specific categories.
Use Union to merge two lists of products from different warehouses, eliminating duplicates.
Use TakeWhile to retrieve orders from a list until an order with quantity less than 5 is encountered.
Demonstrate **Deferred Execution** by modifying the product list after defining a LINQ query and observe the impact on the query results.
Demonstrate **Immediate Execution** by converting a query result to a list using ToList.

Task 4: Complex Queries

Use the let keyword to calculate the total price (price × quantity) for each order and filter orders with a total price above $100.
Use the into keyword to group customers by their city and filter cities with more than 5 customers.
Write a LINQ query to group products by category, calculate the average price per category, and display the result.
Use let to define an intermediate variable for stock status (e.g., "Low Stock" if stock < 10) and filter products based on this status.
Use into to create a continuation query that first filters premium customers and then sorts them by their names.

Online Reference

https://www.tutorialsteacher.com/linq/linq-lambda-expression

.NET Core Web API

WEB API (old)

Authentication And Authorization (WEBAPI)(old)

FullStackDevelopment_WIth_Dotnet_AND_Angular