All questions are directly aligned with the Ecommerce end-to-end demo and cover **C# Basics → OOP → Collections → LINQ → Delegates → Async → Events → Exceptions**.

📝 **LAB PRACTICE QUESTIONS – ECOMMERCE MODULE**

Level: Freshers / Beginner–Intermediate

Total Questions: 35

✅ **SECTION 1 — C# BASICS (6 Questions)**

1. Write a program to take **product price and quantity** from the user and print the **total bill**.
2. Write a program using **if/else** to check whether a user is **Admin** or **Customer** based on role input.
3. Write a **switch-case** program to perform operations: Add Product, View Product, Exit.
4. Demonstrate **implicit and explicit casting** using price to discount conversion.
5. Write a program using a **for-loop** to print product IDs from 1 to 10.
6. Show the difference between **var**, **dynamic**, and **object** with examples.

✅ **SECTION 2 — OOP & INTERMEDIATE C# (8 Questions)**

7. Create a Product class with fields: Id, Name, Price, Stock, and methods: ReduceStock().
8. Create a base class User and two derived classes (Customer, Admin) and implement method overriding.
9. Create an interface IShopper with method AddToCart() and implement it in Customer.
10. Demonstrate method overloading with 2 versions of CalculateTotal() method.
11. Create a **constructor** that initializes product details and print all values.
12. Create an example of **sealed class** or **sealed method** and explain why we use it.
13. Write a program using **base keyword** to call parent constructor from child.
14. Create a **partial class** Customer1 and Customer2 and combine in runtime.

✅ SECTION 3 — COLLECTIONS & GENERICS (5 Questions)

15. Create a List<Product> and add 5 products manually.
16. Store product-wise stock using a **Dictionary** (key: productId, value: stock).
17. Write a generic method GetFirst<T>(List<T> list) that returns the first item.
18. Create a HashSet<string> of unique categories and display them.
19. Implement a Stack<string> for storing recently viewed products.

✅ SECTION 4 — EXCEPTION HANDLING (4 Questions)

20. Create a custom exception InvalidOrderException thrown when order amount is 0.
21. Write try–catch–finally where finally prints: “Process Completed”.
22. Demonstrate the difference between **throw** and **throw ex**.
23. Catch multiple exceptions (FormatException, NullReferenceException).

✅ SECTION 5 — DELEGATES, EVENTS & LAMBDA (6 Questions)(Optional)

24. Create a delegate ShippingCostDelegate and pass lambda to calculate shipping cost.
25. Use built-in **Func** delegate to calculate total price including GST.
26. Use **Predicate** to filter only in-stock products.
27. Write an event LowStockReached and raise it when stock < 5.
28. Create a custom event OrderCancelled and subscribe to display a message.
29. Write a lambda expression to return all product names in uppercase.

✅ SECTION 6 — LINQ (4 Questions)

30. Write a LINQ query to return all products where price > 1000.
31. Write a LINQ query to group products by category.
32. Using LINQ, sort products by price in descending order.
33. Use LINQ to calculate total stock value: (Price × Stock) for all products.

✅ **SECTION 7 — ASYNC PROGRAMMING (2 Questions)**

34. Write an async method SimulatePayment() that waits 3 seconds and returns “Success”.

35. Call the above async method from Main() using await and print the success message.

🎯 **BONUS QUESTIONS (Optional)**

36. Implement a background task that checks stock every 5 seconds using async delay.

37. Use multiple tasks to process 3 payments in parallel using Task.WhenAll.

38. Create a LINQ query returning the **second highest priced product**.

39. Implement a delegate chain (multicast delegate) for logging.

40. Add a JSON file with product data and load it into List<Product> using System.Text.Json.