1){
var result = context.Employees .GroupBy(e => e.Department) .Select(g => new
{

Department = g.Key,
MaxSalary = g.Max(e => e.Salary), AvgSalary = g.Average(e => e.Salary), TotalSalary = g.Sum(e => e.Salary), Count = g.Count()

}) .ToList();

} answer: GroupBy işlemi ile SQL tarafından yapılır.

2){

var result = string.Join("-", Enumerable.Repeat("Hi", 3));

Console.WriteLine(result);

}answer: Hi-Hi-Hi

3){

var query = context.Orders
.Where(o => o.TotalAmount > 1000) .AsEnumerable()
.Where(o => IsPrime(o.Id))
.ToList();

}answer: İlk Where SQL'de, ikinci Where belleğe alındıktan sonra çalışır.

4){

using (var context = new AppDbContext())

{
var departments = context.Departments

.Include(d => d.Employees) .AsSplitQuery()
.AsNoTracking()
.Where(d => d.Employees.Count > 5) .ToList();

}

}answer: Department ve Employee verileri iki ayrı SQL sorgusu ile getirilir, EF Core değişiklik izleme yapmaz.

5){

var result = string.Format("{1} {0}", "Hello", "World");

Console.WriteLine(result);

}answer: "World Hello"

6)Aşağıdakilerden hangisi System.Linq.Enumerable ve System.Linq.Queryable arasındaki farktır?

Answer: Enumerable metodları IEnumerable üzerinde çalışır, Queryable metodları Expression Tree ile sorgu üretir.

7){var people = new List<Person>{ new Person("Ali", 35),
new Person("Ayşe", 25),
new Person("Mehmet", 40)

};
var names = people.Where(p => p.Age > 30)

.Select(p => p.Name) .OrderByDescending(n => n);

Console.WriteLine(string.Join(",", names));

}asnwer: Mehmet,Ali

8){var numbers = new List<int>{1,2,3,4,5,6}; var sb = new StringBuilder();
numbers.Where(n => n % 2 == 0)

.Select(n => n * n)
.ToList()
.ForEach(n => sb.Append(n + "-"));

Console.WriteLine(sb.ToString().TrimEnd('-'));

}answer: 4-16-36

9)System.Text.Json ve System.Collections.Generic kullanılarak bir listeyi JSON'a dönüştürmek ve ardından deseralize etmek için doğru işlem sırası nedir?

Answer: Listeyi serialize et → JSON string oluştur → Deserialize → liste

10) {var products = context.Products .AsNoTracking()
.Where(p => p.Price > 100)
.Select(p => new { p.Id, p.Name, p.Price }) .ToList();

products[0].Name = "Updated Name";
var trackedEntities = context.ChangeTracker.Entries().Count();

}asnwer: 0

11){var departments = context.Departments .Include(d => d.Employees)

.ThenInclude(e => e.Projects) .AsSplitQuery()
.OrderBy(d => d.Name)
.Skip(2)

.Take(3) .ToList();

}answer: Skip/Take sadece ana tabloya uygulanır, ilişkilerde tüm kayıtlar gelir.

12){var query = context.Customers .GroupJoin(

context.Orders,
c => c.Id,
o => o.CustomerId,
(c, orders) => new { Customer = c, Orders = orders }

)
.SelectMany(co => co.Orders.DefaultIfEmpty(),

(co, order) => new {

CustomerName = co.Customer.Name,

OrderId = order != null ? order.Id : (int?)null })

.ToList();

}

}answer: Siparişi olmayan müşteriler de listelenir, OrderId null olur.

13){var names = context.Employees
.Where(e => EF.Functions.Like(e.Name, "A%")) .Select(e => e.Name)
.Distinct()
.Count();

}answer: EF.Functions.Like SQL tarafında çalışır, Distinct ve Count SQL tarafında yapılır.

14){var result = context.Orders
.Include(o => o.Customer)
.Select(o => new { o.Id, o.Customer.Name }) .ToList();

}answer: Include bu senaryoda gereksizdir, EF Core sadece Select ile ilgili alanları çeker.

15){var query = context.Employees .Join(context.Departments,

e => e.DepartmentId, d => d.Id,
(e, d) => new { e, d })

.AsEnumerable()
.Where(x => x.e.Name.Length > 5) .ToList();

}answer: Join SQL'de yapılır, Name.Length kontrolü belleğe alındıktan sonra yapılır.