

Test Questions

Bu kodun SQL karşılığıyla ilgili doğru ifade nedir?

```
{
    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();
}
```

A) GroupBy işlemi SQL tarafında yapılır.

-Bu kodda gruptama ve hesaplamaları SQL yapıyor, yani veriler bana gelmeden önce veritabanında hazırlanıyor.

Aşağıdaki kodun çıktısı nedir?

```
{
    var result = string.Join("-", Enumerable.Repeat("Hi", 3));
    Console.WriteLine(result);
}
```

B) Hi-Hi-Hi

-Repeat("Hi", 3) bana üç tane "Hi" verir, string.Join("-", ...) bunları araya - koyarak birleştirir, sonuç Hi-Hi-Hi olur.

Bu kodda IsPrime metodu C# içinde yazılmış özel bir metot. Kodun çalışmasıyla ilgili doğru ifade nedir?

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

B) İlk Where SQL'de, ikinci Where belleğe alındıktan sonra çalışır.

- TotalAmount > 1000 filtresini SQL yapıyor, ama IsPrime SQL'in bilmediği bir metot olduğu için ancak veriler belleğe geldikten sonra çalışıyor.

Kod çalıştırıldığında hangi durum/sonuç gerçekleşir?

```
{
    using (var context = new AppDbContext())
    {
        var departments = context.Departments
            .Include(d => d.Employees)
            .AsSplitQuery()
            .AsNoTracking()
            .Where(d => d.Employees.Count > 5)
            .ToList();
    }
}
```

B) Department ve Employee verileri iki ayrı SQL sorgusu ile getirilir, EF Core değişiklik izleme yapmaz.

- Kod iki sorgu çalıştırır ve verileri getirir ama değişiklikleri takip etmez.

Aşağıdaki kodun çıktısı nedir?

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

C) "World Hello"

- {1} ikinci parametreyi yani "World", {0} ise ilk parametreyi yani "Hello" alır, bu yüzden çıktı World Hello olur.

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

B) Enumerable metodları IEnumerable üzerinde çalışır, Queryable metodları Expression Tree ile sorgu üretir

-Enumerable bellekte çalışır, Queryable ise veritabanına çevrilecek sorgu oluşturur.

Aşağıdaki kodun çıktısı nedir?

```
{
    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));
}
```

B) Mehmet,Ali

-30'dan büyük yaşlılar Ali (35) ve Mehmet (40), isimleri ters sırayla sıralanınca Mehmet,Ali çıkar.

Aşağıdaki kodun çıktısı nedir?

```
{
    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('-'));
}
```

A) 4-16-36

-Çift sayılar (2,4,6) seçilir, karesi alınır (4,16,36) ve tire ile yazdırılır.

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

A) Listeyi serialize et → JSON string oluştur → Deserialize → liste

-Önce listeyi JSON string'e çeviririm (serialize), sonra JSON'dan tekrar listeye dönüştürürüm (deserialize).

Aşağıdaki kodda trackedEntities değeri kaç olur?

```
{  
    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();  
}
```

A) 0

-AsNoTracking() yüzünden EF Core ürünleri takip etmez

Hangisi doğrudur?

```
{  
    var departments = context.Departments  
        .Include(d => d.Employees)  
        .ThenInclude(e => e.Projects)  
        .AsSplitQuery()  
        .OrderBy(d => d.Name)  
        .Skip(2)  
        .Take(3)  
        .ToList();  
}
```

B) Skip/Take sadece ana tabloya uygulanır, ilişkilerde tüm kayıtlar gelir.

-Skip/Take sadece Departments tablosuna etki eder, ilişkili Employees ve Projects verileri tam gelir.

Bu kodun sonucu ile ilgili doğru ifade hangisidir?

```
{
    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();
}
```

B) Siparişi olmayan müşteriler de listelenir, OrderId null olur.

-DefaultIfEmpty() sayesinde siparişi olmayan müşteriler de gelir.

Bu kodun SQL karşılığı ile ilgili hangisi doğrudur?

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

A) EF.Functions.Like SQL tarafında çalışır, Distinct ve Count SQL tarafında yapılır.

-Like, Distinct ve Count hepsi SQL'e çevrilir, sonuç doğrudan veritabanında hesaplanır.

Hangisi doğrudur?

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

A) Include bu senaryoda gereksizdir, EF Core sadece Select ile ilgili alanları çeker.
-Select zaten Customer.Name'i alıyor, Include eklemeye gerek yok.

Hangisi doğrudur?

```
{
    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();
}
```

B) Join SQL'de yapılır, Name.Length kontrolü belleğe alındıktan sonra yapılır.

-Join SQL tarafında çalışır ama AsEnumerable() sonrası Name.Length filtresi bellekte yapılır.