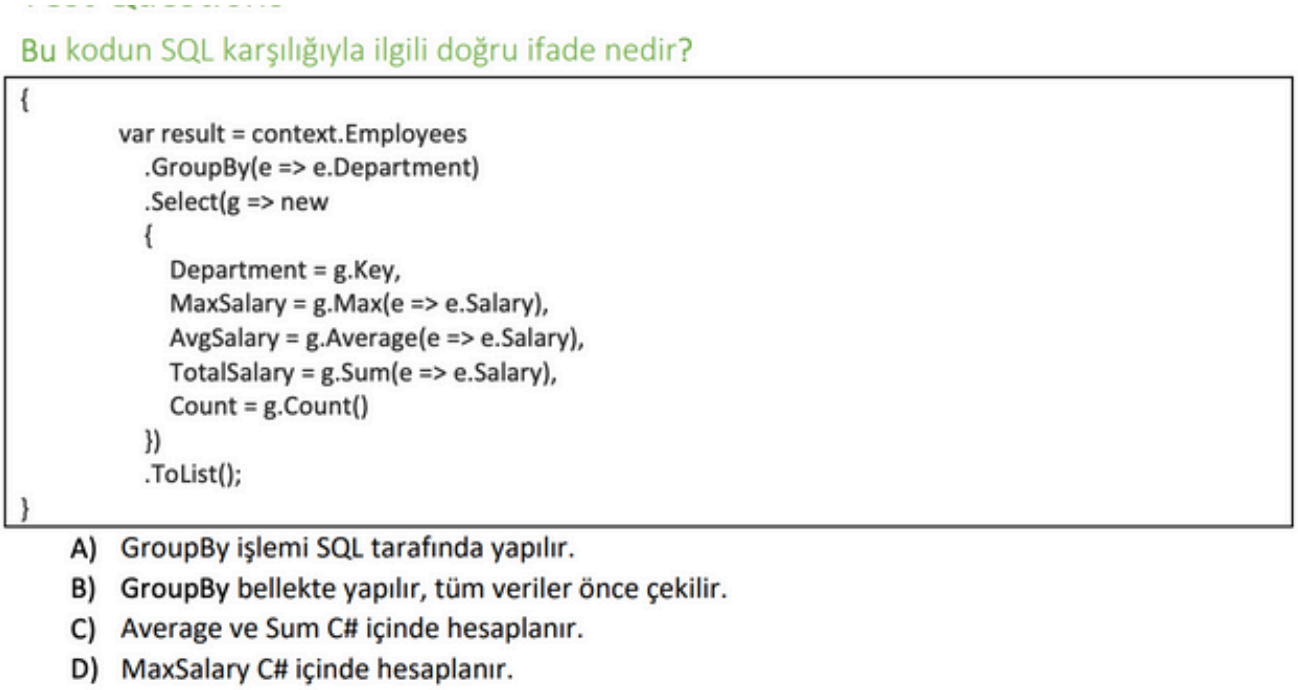
Test Questions and Answers

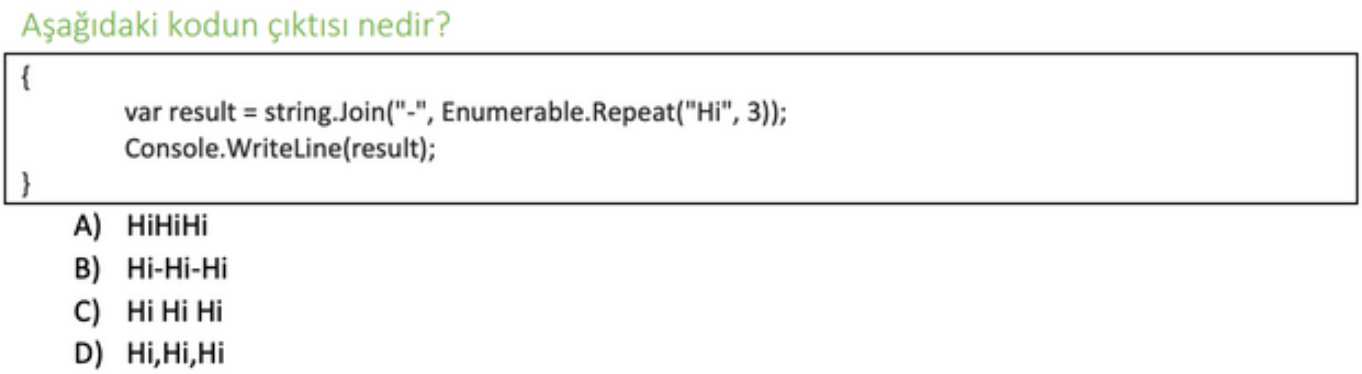
SORU 1)



Cevap: A) GroupBy işlemi SQL tarafında yapılır.

Açıklama: EF Core bu sorguyu tamamen SQL'e çevirir. GROUP BY, MAX, AVG, SUM ve COUNT fonksiyonları veritabanı tarafında çalıştırılır. Bu, performans açısından en verimli yaklaşımdır çünkü büyük veri setlerinde gruplama ve hesaplama işlemleri veritabanında yapılır.

SORU 2)



Cevap: B) Hi-Hi-Hi

Açıklama: Enumerable.Repeat("Hi", 3) "Hi" stringini 3 kez tekrar eden bir koleksiyon oluşturur. string.Join("-", ...) bu elemanları "-" ayırıcısıyla birleştirir.

SORU 3)

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();
}
```

A) Tüm filtreler SQL tarafında çalışır, performans çok yüksektir.
B) İlk Where SQL'de, ikinci Where belleğe alındıktan sonra çalışır.
C) Tüm Where filtreleri bellekte çalışır.
D) AsEnumerable sorguyu hızlandırır, hepsi SQL tarafında çalış

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

Açıklama: AsEnumerable() çağrısı, sorgunun geri kalanının bellekte (client-side) çalışmasını sağlar. İlk Where filtresi SQL'e çevrilir ve veritabanında çalışır. AsEnumerable()'dan sonraki ikinci Where filtresi ve IsPrime metodu bellekte çalışır.

SORU 4)

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();
        }
}
```

A) Tüm Department kayıtları tek bir SQL sorgusu ile, JOIN kullanılarak getirilir. EF Core değişiklik izleme yapar.
B) Department ve Employee verileri iki ayrı SQL sorgusu ile getirilir, EF Core değişiklik izleme yapmaz.
C) Department ve Employee verileri ayrı sorgularla getirilir, ancak EF Core değişiklik izleme yapar.
D) Tüm veriler tek sorguda getirilir ve değişiklik izleme yapılmaz.

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

Açıklama:

- AsSplitQuery(): Include edilen tablolar için ayrı sorgular oluşturur

- AsNoTracking(): Change tracking'i devre dışı bırakır

- İki ayrı sorgu çalıştırılır: biri Departments, diğeri Employees için.

SORU 5)

Aşağıdaki kodun çıktısı nedir?

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

A) "{0} {1} "
B) "Hello World"
C) "World Hello"
D) "HelloWorld"

Cevap: C) "World Hello"

Açıklama:

{1} ikinci parametreyi (World), {0} birinci parametreyi (Hello) referans alır.

Sonuç: "World Hello"

SORU 6)

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

A) Enumerable metodları yalnızca IQueryable üzerinde çalışır
B) Enumerable metodları IEnumerable üzerinde çalışır, Queryable metodları Expression Tree ile sorgu üretir
C) Enumerable metodları SQL veritabanına sorgu gönderir
D) Queryable metodları yalnızca string koleksiyonları üzerinde çalışır

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

Açıklama:

- System.Linq.Enumerable: LINQ to Objects, bellekteki koleksiyonlar üzerinde çalışır

- System.Linq.Queryable: LINQ to SQL/EF, Expression Tree kullanarak sorgu sağlayıcılarına çevrilir.

SORU 7)

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));
}
```

A) Ali,Mehmet
B) Mehmet,Ali
C) Ayşe,Ali,Mehmet
D) Ali

Cevap: B) Mehmet,Ali

Açıklama:

1. Where: Yaşı 30'dan büyük olanları filtreler (Ali:35, Mehmet:40)

2. Select: Sadece isimleri alır (Ali, Mehmet)

3. OrderByDescending: Alfabetik olarak ters sıralar (Mehmet, Ali)

SORU 8)

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
B) 2-4-6
C) 1-4-9-16-25-36
D) 4-16-36-

Cevap: A) 4-16-36

Açıklama:

1. Where: Çift sayıları filtreler (2,4,6)

2. Select: Karelerini alır (4,16,36)

3. ForEach: StringBuilder'a "-" ile ekler (4-16-36-)

4. TrimEnd('-'): Son "-" karakterini kaldırır (4-16-36)

SORU 9)

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

A) Listeyi serialize et → JSON string oluştur → Deserialize → liste
B) Listeyi deserialize et → JSON string oluştur → liste
C) JSON string oluştur → liste → serialize
D) JSON string parse → ToString()

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

Açıklama: Standart işlem sırası:

1. C# nesnesini JSON string'e serialize et

2. JSON string'i tekrar C# nesnesine deserialize et

SORU 10)

Aşağıdaki kodda trackedEntitites 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
B) 1
C) Ürün sayısı kadar
D) EF Core hata fırlatır

Cevap: A) 0

Açıklama:

- AsNoTracking() ile getirilen nesneler izlenmez

- Select ile anonymous type oluşturulur (immutable)

- products[0].Name = "Updated Name" satırı compile hatası verir çünkü anonymous type read-only'dir

- ChangeTracker hiçbir entity izlemez, sonuç 0.

SORU 11)

Hangisi doğrudur?

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

A) Her include ilişkisi ayrı sorgu olarak çalışır, Skip/Take her sorguya uygulanır.
B) Skip/Take sadece ana tabloya uygulanır, ilişkilerde tüm kayıtlar gelir.
C) Skip/Take hem ana tablo hem ilişkili tablolara uygulanır.
D) AsSplitQuery performansı düşürür, tek sorgu ile çalışır

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

Açıklama: AsSplitQuery() ile ayrı sorgular oluşturulur, ancak Skip/Take sadece ana sorguya (Departments) uygulanır. İlişkili tablolar (Employees, Projects) için filtrelenen department'ların tüm kayıtları gelir.

SORU 12)

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();
}
```

A) Sadece siparişi olan müşteriler listelenir.
B) Siparişi olmayan müşteriler de listelenir, OrderId null olur.
C) Sadece siparişi olmayan müşteriler listelenir.
D) GroupJoin SQL tarafında çalışmaz, tüm veriler belleğe alınır

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

Açıklama: GroupJoin + SelectMany + DefaultIfEmpty() kombinasyonu LEFT JOIN oluşturur. Siparişi olmayan müşteriler de sonuçta yer alır, OrderId değerleri null olur.

SORU 13)

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.
B) EF.Functions.Like SQL tarafında çalışır, Distinct ve Count bellekte yapılır.
C) Tüm işlemler bellekte yapılır.
D) EF.Functions.Like sadece C# tarafında çalışır

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

Açıklama: Tüm işlemler SQL'e çevrilir:

- EF.Functions.Like → SQL LIKE

- Distinct() → SQL DISTINCT

- Count() → SQL COUNT

Sonuç olarak tek bir SQL sorgusu çalıştırılır.

SORU 14)

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.
B) Include gereklidir, yoksa Customer.Name gelmez.
C) Include ile Customer tüm kolonları gelir, Select bunu filtreler.
D) Select Include'dan önce çalışır.

Cevap: A) Include bu senaryoda gereksizdir, EF Core sadece Select ile ilgili alanları çeker.

Açıklama: Select ile projection yapılırken EF Core otomatik olarak gerekli JOIN'leri oluşturur. Include bu durumda gereksizdir çünkü sadece belirtilen alanlar (Id Customer.Name) çekilir.

SORU 15)

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();
}
```

A) Join ve Length kontrolü SQL tarafında yapılır.
B) Join SQL'de yapılır, Name.Length kontrolü belleğe alındıktan sonra yapılır.
C) Tüm işlemler SQL tarafında yapılır.
D) Join bellekte yapılır

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

Açıklama: AsEnumerable() çağrısından önce Join işlemi SQL'e çevrilir. AsEnumerable()'dan sonraki Where filtresi ve Name.Length kontrolü bellekte (client-side) çalıştırılır.