Test Questions

1) Bu kodun SQL karşılığıyla ilgili doğru ifade nedir? A

```
{
            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 is A because this is a LINQ operation groupBy operation takes place in SQL not C# therefore its the correct answer. If we look at B, its completely wrong. For C, average and sum are the aggregate functions sql which takes in SQL as well. For D, it's same again an aggregate function.

2) Aşağıdaki kodun çıktısı nedir? B

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

Answer is B because it repeates "Hi" for 3 times which appends with an "-" so its Hi-Hi-Hi.

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

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

Answer is B because it needs to check the database for the totalAmount > 1000 and then uses a C# method IsPrime() which is in the memory.

4) Kod çalıştırıldığında hangi durum/sonuç gerçekleşir? B

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

Answer is B because the code splits the query into two separate queries using AsSplitQuery(), then uses AsNoTracking() which means EF Core not tracks the changes.

5) Aşağıdaki kodun çıktısı nedir? C

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

Answer is C because the string.Format method changes the values of placeholder depending on the order that you give. Here its 1 and 0 so the answer would be World Hello.

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

Answer is B because Enumerable methods don't work on IQueryable instead Inumerable. Also IQueryable methods works on any collection, not string only.

7) Aşağıdaki kodun çıktısı nedir? B

```
{
          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));
}
```

Answer is B because when we check the LINQ, it filters for the ages higher than 30 and sorts them descendingly which results in Mehmet,Ali

8) Aşağıdaki kodun çıktısı nedir? A

```
{
            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 is A because first it filters the numbers to even. And then multiplies them by themselves, finishes with appending them each other with “-“ resulting in 4-16-36. In order its like 1,2,3,4,5,6 becomes 2,4,6 and then 4,16,36 finally 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? A

Answer is A again because to convert a string to Json first you need to serialize it. To be able to deserialise means to convert Json into a List<string>.

10) Aşağıdaki kodda trackedEntitites değeri kaç olur? A

```
{
            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();}
```

Answer is A because when you use AsNoTracking() you turn off the EF Core ability to keep track of the changes. Eventually if you check the count of the changed entries you won't find ant so its 0.

11) Hangisi doğrudur? B

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

Answer is B because Skip() and Take() only applies to the main table which is Departments. It doesn't matter if they used AsSplitQuery() since that deals with EF Core related entities.

12) Bu kodun sonucu ile ilgili doğru ifade hangisidir? B

```
{
        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 is B because when you check the query especially this part (`OrderId = order != null ? order.Id : (int?)null`) you can see that it shows the customers with no order so their orderid's are null.

13) Bu kodun SQL karşılığı ile ilgili hangisi doğrudur? A

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

}
```

Answer is A because all of the Like,Distint,Count are aggregate functions, special functions in SQL that makes our lives easier, therefore they work in SQL not in C#.

14) Hangisi doğrudur? A

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

We should need better explanation for this question however if we assume that the order table includes the name of the customers, we can say that the answer is A. The reason being unnecessary is that you can still get the name using only Select().

15) Hangisi doğrudur? B

```
{
        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 is B because join is a sql operation and the name.length will be checked after you have the x.e.name in the memory. In this example length is a attribute of the data that we are receiving therefore its not checked in SQL in contrast of C#.