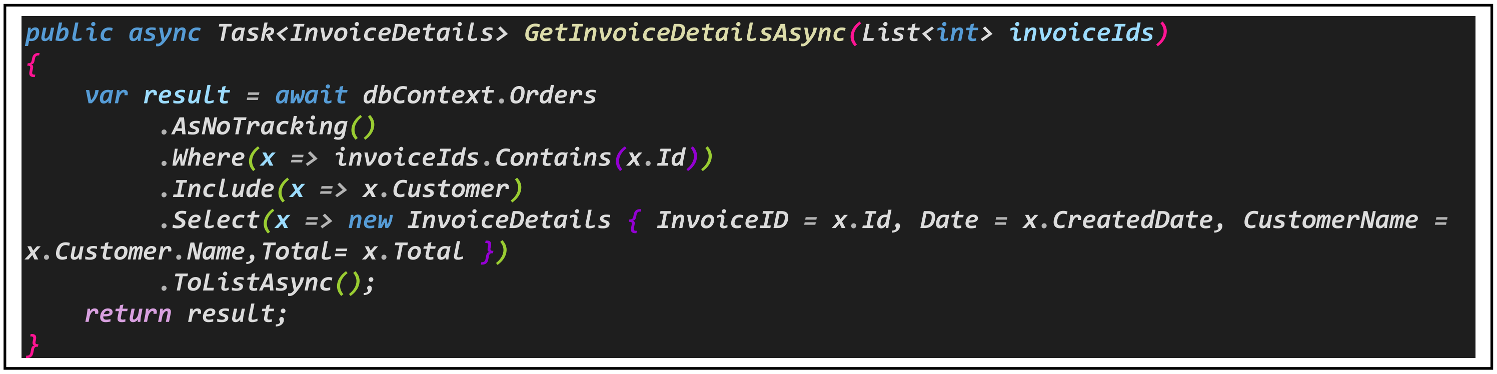
1) **Why do we need a framework like Angular? When should we use jQuery instead?**
   - Use Angular to build interactive SPAs with features such as two-way binding, declarative templates, object oriented, REST API handling, dependency injection, testable and dynamic, its suitable for large applications.
   - use jQuery when building simple app that require basic front-end functionality.
2) **An offer has a date, current status and an optional status date, list of previous statuses, current buyer and offer amount and list of possible offer amounts. An offer amount has a buyer and an amount. Buyer has a name and zip code.**
   c-) What indexes would you define to make the previous query efficient?
   - Non-Clustered index on OfferDate.
3) **In which cases you would use Entity Framework and in which cases you would use a Stored Procedure?**
   - Use **Entity Framework** in straightforward simple CRUD application.
   - Use **SP** in complex applications and complex JOINs to encapsulate business.
4) **What would you do if you had data that doesn't change often but it's used pretty much all the time?**
   - Cache the data
5) **How would you implement a method that needs to return the invoice id, date, customer name, and total based on a list of invoice ids? Invoices are saved in the database so you should use Entity Framework to retrieve them. Should this method be sync or async? What are the advantages or disadvantages of each option?**



```
public async Task<InvoiceDetails> GetInvoiceDetailsAsync(List<int> invoiceIds)
{
    var result = await dbContext.Orders
         .AsNoTracking()
         .Where(x => invoiceIds.Contains(x.Id))
         .Include(x => x.Customer)
         .Select(x => new InvoiceDetails { InvoiceID = x.Id, Date = x.CreatedDate, CustomerName =
x.Customer.Name,Total= x.Total })
         .ToListAsync();
    return result;
}
```

- **Should be Async cause it call IO operation and may take time to response so async here free the thread to other operations to execute till this method finish execution.**

	Sync	**Async**
Advantages	- **Useful with business operations with no IO calling** - **Useful when write in a shared object** - **Easy to debug** -	- **Increase Performance** - **Useful with IO operations**
Disadvantages	- **Can lead to UI thread blocking** - **Block the thread tell operation finish processing**	- **Complexity** - **Debugging** - **Potential Performance Issues Can lead to Thread Starvation** - **Can cause wrong data when write in a shared object.**