1. What is your favourite design pattern, and why?
2. My favourite design pattern is the MVC design pattern. The reasons are as follows

* Since MVC handles multiple views using the same enterprise model it is easier to maintain test and upgrade the multiple system.
* It is easier to add new clients by adding their views and controllers.
* The Model is completely decoupled from the view it allows lot of flexibilities to design and implement the model considering reusability and modularity. This model also can be extended for further distributed application.
* It is possible to have development process in parallel for model, view and controller.
* This makes the application extensible and scalable.

1. For your favourite programming language, tell me about a new (or upcoming) language feature that has you excited. Why is it exciting for you?
2. My favourite programming language is c#. The latest version 8.0 introduced the

**Using declaration**

The using statement is a great way to ensure that the Dispose method will be called on a type implementing the IDisposable interface when an instance gets out of scope.

using (var reader = new StreamReader(filename))

{

var contents = reader.ReadToEnd();

Console.WriteLine($"Read {contents.Length} characters from file.");

}

In C# 8.0, the using declaration is available as an alternative:

using var reader = new StreamReader(filename);

var contents = reader.ReadToEnd();

Console.WriteLine($"Read {contents.Length} characters from file.");

The using keyword can now be placed in front of a variable declaration. When such a variable falls out of scope (i.e. the containing block of code is exited) the Dispose method will automatically be called.

This can be especially useful when multiple instances of types implementing the IDisposable interface are used in the same block of code:

using var reader1 = new StreamReader(filename1);

using var reader2 = XmlReader.Create(filename2);

// process the files

The above code is much more readable and less error-prone than the equivalent code written with the using statement:

using (var reader1 = new StreamReader(filename1))

using (var reader2 = XmlReader.Create(filename2))

{

// process the files

}

3). What do you not like to see when you're reviewing your own of another colleague's code?

a.) Code should not be reviewed unless it already passed its unit tests, static and dynamic analysis and coding standards compliancy checks.

This leaves the reviewer to concentrate on the code quality.

Does it fulfil its requirements?

Does it match the design?

Does it implement the most efficient algorithm for the task?

Is it maintainable?

Is it scalable?

4.) Tell me about a time you fixed a performance issue.

a.) Visual Studio profiling tools help analyze performance issues in your application. SQL Server Profiler is an interface to create and manage traces and analyze and replay trace results.

There was this instance wherein we were working on the checkout basket procedure at Bidvest for the implementation checkout feature. This feature is complex and involves millions of lines of code. During the checkout phase the Basket was taking a long time to load and this had to be optimised.

Using the Visual Studio Profiler and SQL Server Profiler the queries were carefully analysed and the queries which took a long time to execute were moved to the database and exposed as stored procedures. In other words I played a pivotal role in migrating the complex business logic involved in the checkout phase to the database layer thus improving the performance of the checkout process by nearly 50% and keeping the functionality intact.