Q1. What are the types of errors that you usually see in VBA?

Ans: Types of errors:

1. **Syntax errors** – A specific line of code is not written correctly

2. **Compile errors** – Issues that happen when putting together lines of code, though the individual lines of code seem to make sense

3. **Runtime errors** – When the code is usually correct in principle, but an action taken by the user or the data being used leads to unexpected errors.

What are Syntax Errors?

Syntax errors are VBA's way of telling you if your code makes sense, at the most basic of levels. We can make a simple comparison to the rules of writing a sentence in English:

- A subject is required.
- A verb is required.
- A capital letter is required at the beginning.
- Punctuation is required at the end.
- Objects, adjectives, and other grammatical features are optional.

In a similar manner, VBA imposes certain rules over what is required in certain situations. For example, when initiating a basic conditional IF statement in your code, you must replicate the following syntax rules for the first line of an IF statement;

- It must begin with If
- Next must be a logical test, such as x>10
- Finally, it must end with the word then

What are Compile Errors?

Compile errors refer to a wider group of VBA errors, which include syntax errors. Compile errors also identify problems with your code when considered as a whole. The syntax of each individual line may be correct, but when put together, the lines of your code don't make sense. Compile errors are highlighted when you compile or run your code.

What are Runtime Errors?

Once you've fixed syntax errors and compile errors, you're ready to start testing or running your code properly to see how well it executes your desired actions or calculations. We call the period of testing, or live execution of code, "runtime."

Runtime errors cannot be detected by simply looking at the code; they are a result of your code interacting with the specific inputs or data at that time.

Runtime errors are often caused by unexpected data being passed to the VBA code, mismatching data types, dividing by unexpected zeros, and defined cell ranges not being available. Runtime errors are also the most varied and complex to track down and fix.

Q2. How do you handle Runtime errors in VBA?

Ans: Once you've fixed syntax errors and compile errors, you're ready to start testing or running your code properly to see how well it executes your desired actions or calculations. We call the period of testing, or live execution of code, "runtime."

Runtime errors cannot be detected by simply looking at the code; they are a result of your code interacting with the specific inputs or data at that time.

Runtime errors are often caused by unexpected data being passed to the VBA code, mismatching data types, dividing by unexpected zeros, and defined cell ranges not being available. Runtime errors are also the most varied and complex to track down and fix.

Q3. Write some good practices to be followed by VBA users for handling errors?

Ans: VBA Error Handling refers to the process of anticipating, detecting, and resolving VBA Runtime Errors. The VBA Error Handling process occurs when writing code, before any errors actually occur.

VBA Runtime Errors are errors that occur during code execution. Examples of runtime errors include:

- Referencing a non-existent workbook, worksheet, or other object (Run-time Error 1004)
- Invalid data ex. referencing an Excel cell containing an error (Type Mismatch – Run-time Error 13)
- Attempting to divide by zero

Most VBA error handling is done with the On Error Statement. The On Error statement tells VBA what to do if it encounters an error. There are three On Error Statements:

- On Error GoTo 0
- On Error Resume Next
- On Error GoTo Line

On Error GoTo 0

On Error GoTo **0** is VBA's default setting. You can restore this default setting by adding the following line of code:

When an error occurs with On Error GoTo 0, VBA will stop executing code and display its standard error message box.

On Error Resume Next

On Error Resume Next tells VBA to skip any lines of code containing errors and proceed to the next line.

A great time to use On Error Resume Next is when working with objects that may or may not exist. For example, you want to write some code that will delete a shape, but if you run the code when the shape is already deleted, VBA will throw an error. Instead you can use On Error Resume Next to tell VBA to delete the shape if it exists.

On Error GoTo Line

On Error GoTo Line tells VBA to “go to” a labeled line of code when an error is encountered. You declare the Go To statement like this (where errHandler is the line label to go to).

Q4. What is UDF? Why are UDF’s used? Create a UDF to multiply 2 numbers in VBA?

Ans: A user-defined function (UDF) is a common fixture in programming languages, and the main tool of programmers for creating applications with reusable code. Since programs are mostly composed of code that comes from the programmer, or in this case the user, most of it is composed of user-defined functions occasionally punctuated by built-in functions.

User-defined functions allow programmers to create their own routines and procedures that the computer can follow; it is the basic building block of any program and also very important for modularity and code reuse since a programmer could create a user-defined function which does a specific process and simply call it every time it is needed. Their syntax depend entirely on the programming language or application where they are created.

Though part of any programming language, user-defined functions more commonly refer to the special functions that a user creates as scripts or programs in large systems such as databases or spreadsheets like Microsoft Excel. This is because most of the functions being used in a database system or spreadsheet are built-in functions that the user simply has to call and provide parameters, and most of what the application can do is already being done by one or more built-in functions. In this case, user-defined functions are special custom functions that are meant to do something not normally done by the built-in functions. In programming languages such as C, C++ and Java almost every part of the program is user-defined, hence, these functions are no longer referred to as "user-defined" to separate them from the built-in functions, they are simply called functions.

The below code creates a function that will extract the numeric parts from an alphanumeric string.

```
Function GetNumeric(CellRef As String) as Long

Dim StringLength As Integer

StringLength = Len(CellRef)

For i = 1 To StringLength

If IsNumeric(Mid(CellRef, i, 1)) Then Result = Result & Mid(CellRef, i, 1)

Next i

GetNumeric = Result

End Function
```