This error indicates that the struct, enum or enum variant must be matched non-exhaustively as it has been marked as `non_exhaustive`.

When applied within a crate, downstream users of the crate will need to use the `_` pattern when matching enums and use the `..` pattern when matching structs. Downstream crates cannot match against non-exhaustive enum variants.

For example, in the below example, since the enum is marked as `non_exhaustive`, it is required that downstream crates match non-exhaustively on it.

“‘rust,ignore (pseudo-Rust) #[non_exhaustive] pub enum Error { Message(String), Other, }

impl Display for Error { fn fmt(&self, formatter: &mut fmt::Formatter) -> fmt::Result { // This will not error, despite being marked as non_exhaustive, as this // enum is defined within the current crate, it can be matched // exhaustively. let display = match self { Message(s) => s, Other => “other or unknown error”, }; formatter.write_str(display) } }

~~~
An example of matching non-exhaustively on the above enum is provided below:

```rust,ignore (pseudo-Rust)
use mycrate::Error;

// This will not error as the non_exhaustive Error enum has been matched with a
// wildcard.
match error {
    Message(s) => ...,
    Other => ...,
    _ => ...,
}
~~~

Similarly, for structs, match with `..` to avoid this error.