Note: this error code is no longer emitted by the compiler.

An attempt was made to mutate data using a non-mutable reference. This commonly occurs when attempting to assign to a non-mutable reference of a mutable reference (`&(&mut T)`).

Erroneous code example:

```
struct FancyNum {
    num: u8,
}

fn main() {
    let mut fancy = FancyNum{ num: 5 };
    let fancy_ref = &(&mut fancy);
    fancy_ref.num = 6; // error: cannot assign to data in a `&` reference
    println!("{}", fancy_ref.num);
}
```

Here, `&mut fancy` is mutable, but `&(&mut fancy)` is not. Creating an immutable reference to a value borrows it immutably. There can be multiple references of type `&(&mut T)` that point to the same value, so they must be immutable to prevent multiple mutable references to the same value.

To fix this, either remove the outer reference:

```
struct FancyNum {
    num: u8,
}

fn main() {
    let mut fancy = FancyNum{ num: 5 };

    let fancy_ref = &mut fancy;
    // `fancy_ref` is now &mut FancyNum, rather than &(&mut FancyNum)

    fancy_ref.num = 6; // No error!

    println!("{}", fancy_ref.num);
}
```

Or make the outer reference mutable:

```
struct FancyNum {
    num: u8
}

fn main() {
    let mut fancy = FancyNum{ num: 5 };

    let fancy_ref = &mut (&mut fancy);
    // `fancy_ref` is now &mut(&mut FancyNum), rather than &(&mut FancyNum)

    fancy_ref.num = 6; // No error!
```

```
    println!("{}", fancy_ref.num);
}
```