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

This error occurs when an attempt is made to move a borrowed variable into a closure.

Erroneous code example:

```
struct FancyNum {
    num: u8,
}

fn main() {
    let fancy_num = FancyNum { num: 5 };
    let fancy_ref = &fancy_num;

    let x = move || {
        println!("child function: {}", fancy_num.num);
        // error: cannot move `fancy_num` into closure because it is borrowed
    };

    x();
    println!("main function: {}", fancy_ref.num);
}
```

Here, `fancy_num` is borrowed by `fancy_ref` and so cannot be moved into the closure `x`. There is no way to move a value into a closure while it is borrowed, as that would invalidate the borrow.

If the closure can't outlive the value being moved, try using a reference rather than moving:

```
struct FancyNum {
    num: u8,
}

fn main() {
    let fancy_num = FancyNum { num: 5 };
    let fancy_ref = &fancy_num;

    let x = move || {
        // fancy_ref is usable here because it doesn't move `fancy_num`
        println!("child function: {}", fancy_ref.num);
    };

    x();

    println!("main function: {}", fancy_num.num);
}
```

If the value has to be borrowed and then moved, try limiting the lifetime of the borrow using a scoped block:

```
struct FancyNum {
    num: u8,
}
```

```
fn main() {
    let fancy_num = FancyNum { num: 5 };

    {
        let fancy_ref = &fancy_num;
        println!("main function: {}", fancy_ref.num);
        // `fancy_ref` goes out of scope here
    }

    let x = move || {
        // `fancy_num` can be moved now (no more references exist)
        println!("child function: {}", fancy_num.num);
    };

    x();
}
```

If the lifetime of a reference isn't enough, such as in the case of threading, consider using an `Arc` to create a reference-counted value:

```
use std::sync::Arc;
use std::thread;

struct FancyNum {
    num: u8,
}

fn main() {
    let fancy_ref1 = Arc::new(FancyNum { num: 5 });
    let fancy_ref2 = fancy_ref1.clone();

    let x = thread::spawn(move || {
        // `fancy_ref1` can be moved and has a `'static` lifetime
        println!("child thread: {}", fancy_ref1.num);
    });

    x.join().expect("child thread should finish");
    println!("main thread: {}", fancy_ref2.num);
}
```