`std::iterator` Is Deprecated

In C++17 you must not derive from 'std::iterator'. Instead it requires you to write the trait 'typedefs' explicitly.

The Standard Library API requires that each iterator type has to expose five typedef s:

- iterator_category the type of the iterator
- value_type type stored in the iterator
- difference_type the type that is the result of subtracting two iterators
- pointer pointer type of the stored type
- reference the reference type of the stored type

```
iterator_category must be one of input_iterator_tag, forward_iterator_tag,
bidirectional_iterator_tag or random_access_iterator_tag.
```

Before C++17, if you wanted to define a custom iterator, you could use std::iterator as a base class. In C++14 it's defined as:

```
template<
   class Category,
   class T,
   class Distance = std::ptrdiff_t,
   class Pointer = T*,
   class Reference = T&
> struct iterator;
```

This helper class made defining the typedefs in a short way:

```
class ColumnIterator
    : public std::iterator<std::random_access_iterator_tag, Column>
{
         // ...
};
```

In C++17 you must not derive from std::iterator and you need to write the

trait typedefs explicitly:

```
class ColumnIterator {
public:
    using iterator_category = std::random_iterator_tag;
    using value_type = Column;
    using difference_type = std::ptrdiff_t;
    using pointer = Column*;
    using reference = Column&;

// ...
};
```

While you have to write more code, it's much easier to read, and it's less errorprone.

For example, the referencing paper mentions the following example from The Standard Library:

```
template <class T, class charT = char, class traits = char_traits<charT> >
class ostream_iterator:
   public iterator<output_iterator_tag, void, void, void, void>;
```

In the above code, it's not clear what all of those four void types mean in the definition.

Additionally, std::iterator could lead to complicated errors in the name lookup, especially if you happen to use the same name in the derived iterator as in the base class.

Extra Info: You can read more information in the paper P0174R2 - Deprecating Vestigial Library Parts in C++17.

Let's have a look at some smaller elements that were altered in the Standard Library.