There are three message delivery semantics:

1. at most once, at least once and exactly once. At most once is messages may be lost but are never redelivered.
2. At least once is messages are never lost but may be redelivered.
3. Exactly once is each message is delivered once and only once. Exactly once is preferred but more expensive, and requires more bookkeeping for the producer and consumer.

To implement “at-most-once” consumer reads a message, then saves its offset in the partition by sending it to the broker, and finally process the message. The issue with “at-most-once” is a consumer could die after saving its position but before processing the message. Then the consumer that takes over or gets restarted would leave off at the last position and message in question is never processed.

To implement “at-least-once” the consumer reads a message, process messages, and finally saves offset to the broker. The issue with “at-least-once” is a consumer could crash after processing a message but before saving last offset position. Then if the consumer is restarted or another consumer takes over, the consumer could receive the message that was already processed. The “at-least-once” is the most common set up for messaging, and it is your responsibility to make the messages idempotent, which means getting the same message twice will not cause a problem (two debits).

To implement “exactly once” on the consumer side, the consumer would need a two-phase commit between storage for the consumer position, and storage of the consumer’s message process output. Or, the consumer could store the message process output in the same location as the last offset.

Kafka offers the first two, and it up to you to implement the third from the consumer perspective.