

# KIP-941: Range queries to accept null lower and upper bounds


- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Test Plan](#)
- [Rejected Alternatives](#)

## Status

**Current state:** *["Adopted"]*

**Discussion thread:** <https://lists.apache.org/thread/wnn2qrb7vglw11bdm2cdlkm1430b75zc>

JIRA:

 Unable to render Jira issues macro, execution error.

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

Within the RangeQuery class in the Interactive Query API (which allows you to leverage the state of your application from outside your application), there are methods for getting the upper and lower bounds. When web client requests come in with query params (which become those bounds), it's common for those params to be null. We want developers to just be able to pass in the upper/lower bounds if they want instead of implementing their own logic to avoid getting the whole range.

An example of the logic they can avoid after this KIP is implemented is below:

```
private RangeQuery<String, ValueAndTimestamp<StockTransactionAggregation>> createRangeQuery(String lower,
String upper) {
    if (isBlank(lower) && isBlank(upper)) {
        return RangeQuery.withNoBounds();
    } else if (!isBlank(lower) && isBlank(upper)) {
        return RangeQuery.withLowerBound(lower);
    } else if (isBlank(lower) && !isBlank(upper)) {
        return RangeQuery.withUpperBound(upper);
    } else {
        return RangeQuery.withRange(lower, upper);
    }
}
```

## Public Interfaces

We are not explicitly changing the public interface, but a KIP is appropriate because there's a behavior change. As it stands on 6/26/23, if you pass in a null value to the RangeQuery without a specified upper and lower bound, it will perform a full range scan. After this proposed change, you will need to specify an upper and lower bound of null in order to perform a full range scan. This change in behavior will need to be documented.

## Proposed Changes

The proposed change takes [line 57](#) of the RangeQuery java class from:

```
public static <K, V> RangeQuery<K, V> withRange(final K lower, final K upper) {  
    return new RangeQuery<>(Optional.of(lower), Optional.of(upper));  
}
```

to:

```
public static <K, V> RangeQuery<K, V> withRange(final K lower, final K upper) {  
    return new RangeQuery<>(Optional.ofNullable(lower), Optional.ofNullable(upper));  
}
```

## Compatibility, Deprecation, and Migration Plan

This will not affect users who are implementing a check of their nulls in their own code.

Users who send a web request with no parameters will still receive a full scan. This is mostly an improvement for new adopters of range queries.

New users will have the convenience of using the upper and lower bounds, and the docs will need to be updated to let users know they get the benefit of a full range scan when they insert null values.

## Test Plan

*Relevant unit tests will be added to demonstrate the functionality.*

## Rejected Alternatives

N/A.