Iceberg V4 Aggregate Column Stats

Contact: Anoop Johnson <anoop@apache.org>
Last update: Apr 24, 2026

Background

In Iceberg v3, manifest-level statistics are partition-field summaries. Since the stats are computed from partition values, it is always present for every file. Aggregates like min-of-min/max-of-max etc are always correct because there are never missing values.

In v4, we are making two changes that break this assumption:

1. Partition tuples are replaced by general column statistics: partition values are modeled as stats on derived columns (expressions), and stored in the same `column_stats` struct as non-partition column stats.
2. Root manifest has aggregate column stats over leaf manifests: this enables pruning entire leaf manifests during scan planning.

The problem: column stats at the file level are optional. A data file may be missing stats for any column (e.g. the writer did not collect them, column was added after the file was written etc). If even one data file in the leaf is missing stats, a naive stats aggregation can produce incorrect query results due to false pruning when applying predicates.

Example:

Consider a leaf manifest with 3 data files and a query doing `SELECT * from t where x > 10`

	True max(x) in data	**max(x) in stats**
Data File 1	11	Stats not collected
Data File 2	5	5
Data File 3	4	4
Manifest aggregate (naive max)		5 (incorrect)

The reader evaluating `x > 10` against the manifest aggregate sees `max(x) = 5`, concluding 5 > 10 is false, and prunes the entire leaf manifest. Data file 1, which contains rows where x = 11 that match the query is silently skipped.

Proposed Spec Changes

Define null_count as the sentinel for stats collected

A leaf entry is considered to have collected statistics for a column if `null_count` for that column is non-null. This is a safe sentinel value because `null_count` is always computable by any writer that collects stats (since it's just counting nulls), and a writer that didn't collect stats can emit `null` as null_count.

This is an important distinction because `lower_bound` being `null` is ambiguous: it could mean either that a) that all values are null or NaN (this is a case of valid stats that are safe to aggregate) or b) stats were not collected (unknown values, so unsafe to aggregate).

Define aggregation rules

When all child entries (i.e. entries inside leaf manifests) have collected stats for a column (i.e. have a non-null `null_count`), aggregate statistics are computed as follows:

Stat field	**Aggregation**	**Notes**
lower_bound	Min of non-null child values	Null only if every child's lower_bound is null or NaN
upper_bound	Max of non-null child values	Same as above
null_count	Sum of child values	
value_count	Sum of child values	
nan_count	Sum of child values	
max_uncompressed_length	Max of non-null child values	Must be null if any child is missing it.
avg_uncompressed_length	Average is not aggregatable	Must be null at the manifest level (ie root manifest). Weighted average is an alternative.

If any child entry did not collect statistics for a column (ie `null_count` is `null`), all aggregate statistics for that column must be null.

With this rule, the above example produces `max(x) = null` at the manifest level in the root. The reader evaluates `null > 10` as `null`, and does not prune the manifest.

Partition column pruning

Partition tuples should be retained as required fields on every entry. Manifest-level partition pruning aggregates directly from partition tuples (min/max of transformed values), which guarantees completeness: this is the same as v3 partition field summaries.

Pruning rule

A reader may prune a leaf manifest if evaluating a filter predicate against the manifest's aggregate statistics returns `false`. If the required statistic for evaluation is null, the predicate must evaluate to `null` (ie the manifest must not be pruned)

Considered Alternatives

Track stats completeness with an explicit field

Have an explicit boolean per-column `stats_collected` field at the entry level that lets readers know whether the aggregate can be trusted for pruning.

But this requires readers to check a separate field before trusting the aggregate and adds a new field to the schema.

Require stats for all columns in V4

Mandate that all V4 data file entries must have stats for all columns in the table schema. This eliminates the missing-stats problem entirely since aggregates are always correct.

Rejected because it is too strong a requirement. Requiring stats would make V4 incompatible with common real-world scenarios and complicate the upgrade path from V3.