SELECT Dim1, Dim2,
    SUM(Measure1) AS MSum,
    COUNT(*) AS RecordCount,
    AVG(Measure2) AS MAvg,
    MIN(Measure1) AS MMin
    MAX(CASE WHEN Measure2 < 100 THEN Measure2 END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A', 'B'))
    AND (Filter2 = 'C')
    AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8

db.runCommand({
    mapreduce: "DenormAggCollection",
    query: {
        filter1: { '$in': ['A', 'B'] },
        filter2: 'C',
        filter3: { '$gt': 123 }
    },
    map: function() { emit(
        { d1: this.Dim1, d2: this.Dim2 },
        { msum: this.measure1, recs: 1, mmin: this.measure1,
        mmax: this.measure2 < 100 ? this.measure2 : 0 }
    });,
    reduce: function(key, vals) {
        var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
        for(var i = 0; i < vals.length; i++) {
            ret.msum += vals[i].msum;
            ret.recs += vals[i].recs;
            if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
            if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
                ret.mmax = vals[i].mmax;
        }
        return ret;
    },
    finalize: function(key, val) {
        val.mavg = val.msum / val.recs;
        return val;
    },
    out: 'result1',
    verbose: true
});

db.result1.find({ mmin: { '$gt': 0 } }).
    sort({ recs: -1 }).
    skip(4).
    limit(8);

Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.

Measures must be manually aggregated.

Aggregates depending on record counts must wait until finalization.

Measures can use procedural logic.

Filters have an ORM/ActiveRecord-looking style.

Aggregate filtering must be applied to the result set, not in the map/reduce.

Ascending: 1; Descending: -1