ORACLE

Analyzing 1 billion rows in SQL

And how fast databases can really be

Gerald Venzl
Lead Product Manager of Developer Initiatives
Oracle Database Development

What is this talk about?

We will analyze 1 billion rows of synthetical weather station data in SQL → **LIVE**

Gerald Venzl

- Oracle Database Product Manager
- CNCF Ambassador
  - CNCF.io – *make cloud native computing ubiquitous*
- ISO SQL Standard member
- SQL & Performance enthusiast

Gerald Venzl

github.com/gvenzl

geraldonit.com

linkedin.com/in/geraldvenzl

tobsky.com/gvenzl

The data:

- 1 billion lines in a text file
- 13 GB of data
- The content:

```
$ head measurements_413.txt
Tamale;34.1
San Juan;34.0
St. John's;18.6
Omaha;3.9
Lahore;28.4
Libreville;11.2
Abéché;38.6
Naha;31.6
Phnom Penh;24.3
Gangtok;21.1
```


What we want

Retrieve temperature measurement values and calculate the min, average, and max temperature per weather station in alphabetical order

What that implies

- Scan 13 GB of data
- Group rows per station
- Aggregate min, avg, max values per station
- Sort aggregate

```
SELECT station_name, MIN(val), AVG(val), MAX(val)
 FROM <data>
 GROUP BY station_name
 ORDER BY station_name;
```

In Java

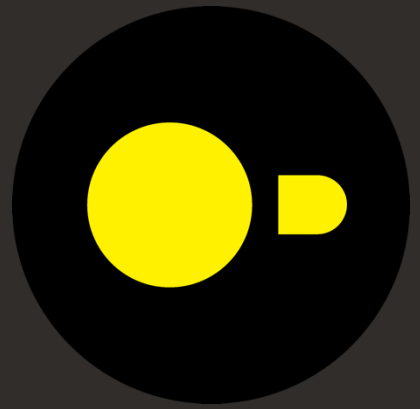
```
public static void main(String[] args) throws IOException {
Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(
            MeasurementAggregator::new,
            (a, m) -> {
               a.min = Math.min(a.min, m.value);
               a.max = Math.max(a.max, m.value);
               a.sum += m.value;
               a.count++;
            },
            (agg1, agg2) -> {
               var res = new MeasurementAggregator();
               res.min = Math.min(agg1.min, agg2.min);
               res.max = Math.max(agg1.max, agg2.max);
               res.sum = agg1.sum + agg2.sum;
               res.count = agg1.count + agg2.count;

               return res;
            },
            agg -> {
               return new ResultRow(agg.min, (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);
            });

      Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE))
            .map(l -> new Measurement(l.split(";")))
            .collect(groupingBy(m -> m.station(), collector)));
```

What we will be using

- 128 cores of *Intel Xeon Platinum 8358 CPU @ 2.60GHz (max turbo 3.4GHz)*
  - Oracle Cloud Infrastructure BM.Standard3.64
- 1 TB of RAM
- 2 TB of disk space

What we will be testing

DuckDB 1.1.3



Postgres 17

Oracle 19c

Let’s get started

github.com/gvenzl/one-billion-rows-database


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.