

## CrateDB Office Hour #3



- Opportunity to talk to CrateDB engineers ©
- Open Zoom call
- No recording!
- As a starting point, we will show some cool features
- Ask questions, share your use case



## Agenda

Window functions in CrateDB

User defined functions

Geospatial data and search in CrateDB

Your questions



## Window functions

- Window functions operate on a subset of rows defined by the window specification
- Help perform calculations that depend on the rows' order or involve comparisons with other rows in the same result set
- To declare window function, use the OVER clause to define the window

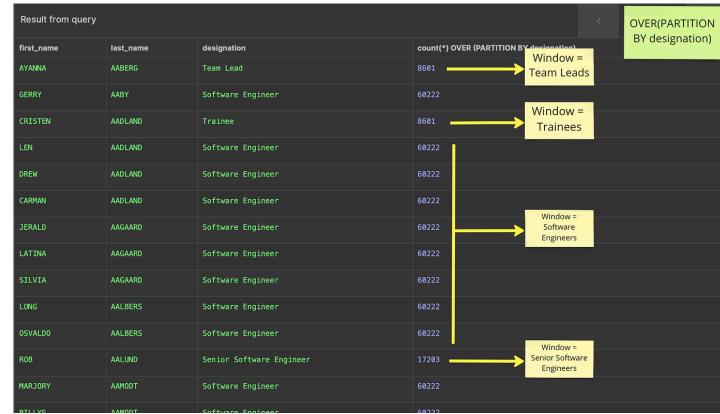
```
SELECT col1, col2, COUNT(*) OVER(PARTITION BY col1)
FROM table1
ORDER BY col2;
Computation window as a partition by col1
```



## Window functions: example



- Count the number of employees for each designation value (e.g., trainee, team lead, software engineer)
- Each designation is a window on which count will be calculated





## WINDOW clause

- Defining window using WINDOW clause can simplify the code and allow you to reuse window specification across multiple expressions
- The ORDER and PARTITION define what is referred to as the window—the ordered subset of data over which calculations are made

```
SELECT first_name,
    last_name,
    designation,
    salary,
    MAX(salary) OVER w max_salary
FROM demo.employees
WINDOW w AS (PARTITION BY designation ORDER BY salary DESC)
ORDER BY last_name
LIMIT 100;
```

 For each employee, show the name, designation, employee's salary and the \*max salary for her/his designation group

<sup>\*</sup> The list of aggreagate and general-purpose functions on the defined window: https://community.crate.io/t/window-functions-in-cratedb/1398



## User-Defined Functions (UDFs)

- Allows you to expand the capabilities of CrateDB with JavaScript code, creating reusable functions.
- For example, get\_postal\_code function can be used to extract postal code from the address
- A dedicated scalar function: regexp matches

And then, use it as follows:

```
SELECT get_postal_code(address) FROM demo.employees
```

```
CREATE FUNCTION get_postal_code(address STRING)
RETURNS STRING
LANGUAGE JAVASCRIPT
AS '
  function get_postal_code(address) {
    const postalCodeRegex = /\d{5}\$/;
    const match = address.match(postalCodeRegex);
    if (match) {
       return match[0];
    } else {
       return null;
    }
  }
}
';
```

#### Check other UDF examples here

https://community.crate.io/t/user-defined-function-collection/773 https://community.crate.io/t/advanced-downsampling-with-the-lttb-algorithm/1287



## Working with Geospatial data

- GEO\_POINT: geographic data type used to store latitude and longitude coordinates.
- GEO\_SHAPE: <u>geographic data type</u> used to store 2D shapes defined as <u>GeoJSON geometry</u> <u>objects</u>.
- GEO\_SHAPE can store different types of GeoJSON geometry objects:

## Geospatial data: Functions

- distance (geo\_point1, geo\_point2) calculate the distance between two points on Earth
- intersection (geo\_shape1, geo\_shape2) returns true if both shapes share some points or area
- within (shape1, shape2) returns true if shape1 is within shape 2
- geohash (geo point) returns a GeoHash representation based on full precision
- area (geo\_shape) calculates the area of the input shape in square-degrees

For each city, find distance from north pole:

```
SELECT city, distance(location, 'POINT(0.0 90.0)') AS dist_north_pole
FROM "demo"."world_cities"
LIMIT 100;
```



## GEO\_SHAPE

- GEO\_SHAPE is represented as an object containing a valid GeoJSON geometry object:
- Alternatively, a 'well-known text representation' string can be used to represent GEO\_SHAPE:

```
'POLYGON ((5 5, 10 5, 10 10, 5 10, 5 5))'
```

Find if two shapes intersect:



# Your questions



Contact: marija@crate.io



# Thank you 😲