PromQL
you probably (maybe) don't use

Starring:

- without()
- group_left / group_right
- ignoring() / on()

Example query you're probably used to:

```
sum(rate(http_requests_total{code="500"}[2m]))
/ sum(rate(http_requests_total[2m]))
```

(summed rate of 500s per total requests)

Or maybe:

```
sum by(job) (rate(http_requests_total{code="200"}[2m]))
/ sum by(job) (rate(http_requests_total[2m]))
```

(per job summed rate of 200s per total requests)

without()

your_label{x="a",y="b",z="c", q="r", s="t"}

sum by(x,y,z,q) == sum without(s)

without(): remove the labels you don't need

vs.

by(): limiting labels in resultant query

ignoring() / on()

Default vector operations are **one to one**

> >Two entries match if they have the **exact same set of labels** and corresponding values.

The following label sets don't match:

- http_errors_total{code, handler, method}
- http_requests_total{code, instance, job, method}

ignoring() */ on()*

- http_errors_total{code, handler, method}
- http_requests_total{code, instance, job, method}

sum without(handler) (rate(http_errors_total[2m]))
/ **ignoring(job)**
sum without(instance) (rate(http_requests_total[2m]))

* There is a potential problem here -- will be addressed soon

*ignoring() / **on()***

- http_errors_total{code, handler, method}
- http_requests_total{code, instance, job, method}

sum without(handler) (rate(http_errors_total[2m]))
/ **on(code,method)**
sum without(instance, job) (rate(http_requests_total[2m]))

* There is a potential problem here -- will be addressed soon

ignoring() / on()

Voila -- we are able to manipulate which labels are used when matching vectors

> >Operations between vectors attempt to find a matching element in the right-hand-side vector for each entry in the left-hand side.

The previous operation *could work*, as long as both queries result in matched vectors. But if the two queries don't match, the query will fail.

```
Error executing query: multiple matches for labels: many-to-one matching must be explicit (group_left/group_right)
```

group_left() / group_right()

Handles the problem of 1:n or n:1

The direction indicates which side has higher cardinality:

Higher Cardinality

```
sum without(handler) (rate(http_errors_total[2m]))
/ on(code,method) group_left
sum without(instance, job) (rate(http_requests_total[2m]))
```