# The Tallying Schema

This is a schema as passed when intializing a cluster/table for a particular dataset. You may modify this schema as necessary. Queries are generated automatically based on the flags that are present.

The `total count` and `sum` of values is tracked by default, unless flagged for `Distinct` values, where each value is maintained.

Keep in mind, that the totals change every day. You may use our `save state` option to take snapshots of your aggregates over time, or you may read your set everyday, and keep track of state on your own.

```
[
    {
        "attrName":"City",
        "type":"string",
        "isDistinct":true
    },
    {
        "attrName":"Temperature",
        "type":"integer",
        "isDistinct":false
    },
    {
        "attrName":"Forecast",
        "type":"string",
        "isDistinct":true,
        "relate":["City"]
    }

]
```

Breaking it Down:

| Attribute Name | Meaning |
|---|---|
| `attrName` | This identifies the "column" or attribute that is present in the schema |
| `type` | This will let us know the type for the actual data value |
| `isDisinct` | The isDistinct Flag let's us keep track of individual elements when necessary. If elements repeat, how many times they repeat is also kept. Read below for more details |
| `relate` | An array with attribute names to keep a running aggregate of the `count` of one attribute in relation to another. Useful to see metrics such as Number of Sunny Days in Seattle |

Note:

```
In the future, we will expand our relate functions such that they may hold more
aggregate functions more than just counting. For example, this might mean
relating such that a set of integers, or averages might be held instead of a
count. Eg. The average temperatures in a relation over time.
```

## How Disinction Works

When the Distinct flag is used, we keep track in array like so:

```
[
    {
     "Value": "Seattle",
     "Count": 11
    },
    {
     "Value": "Dallas",
     "Count": 11
    }
]
```

Disctinct values are indexed to minimize runtime and keep track of each Distinct Value and their count.

## How Relation Works

Relating an attribute to another, would mean that an attribute will be kept in reference to the other. Of course, the more relations, the higher the costs. Remember, since data is not permanent, once data leaves your table, it will no longer be possible to relate data points after data is no longer being stored. This is how a relation will be stored for your retrieval later:

```
[
    {
     "Value": "Sunny",
     "Related": {
        "Seattle": 3,
        "Dallas": 9
     }
    },
    {
     "Value": "Cloudy",
     "Related": {
        "Seattle": 8,
        "Dallas": 2
     }
    }
]
```

Relations are useful when comparing datasets.