nucamp

JSON in SQL

Recall: Schemaless databases

Informally described as "NoSQL"

Example: MongoDB (uses JSON format)

Provide flexibility for unstructured data

Postgres provides JSON as a schemaless data type

JSON data types

Type	Efficient Storage & Processing	Easily Portable	Validates JSON Rules	JSON Functions Available
TEXT		Yes		
JSON		Yes	Yes	Yes
JSONB	Yes		Yes	Yes

JSON path

Describes location of value in JSON

JSON arrays use indices

JSON objects use keys

Together, indices and keys form paths to all values in JSON

```
{
    "Title": "Sunflower Seeds",
    "Artist": "Weiwei",
    "Exhibits": [6, 1, 5]
}
```

'{Title}' $\longrightarrow$ 'Sunflower Seeds'

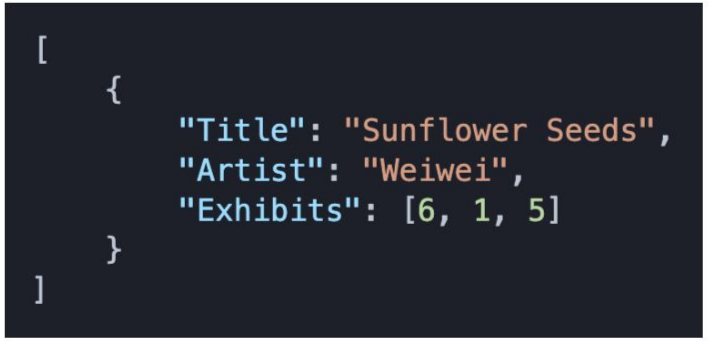
JSON path

Describes location of value in JSON

JSON arrays use indices

JSON objects use keys

Together, indices and keys form paths to all values in JSON



```
[
    {
        "Title": "Sunflower Seeds",
        "Artist": "Weiwei",
        "Exhibits": [6, 1, 5]
    }
]
```

$$\text{'\{0,Title\}'} \longrightarrow \text{'Sunflower Seeds'}$$

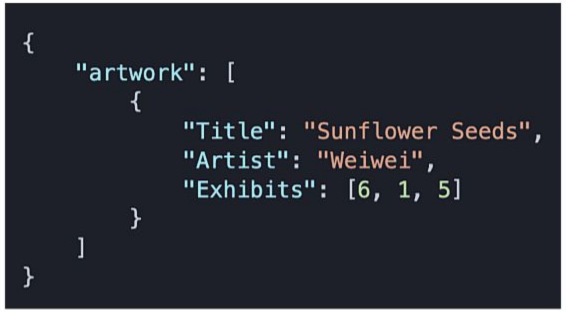
JSON path

Describes location of value in JSON

JSON arrays use indices

JSON objects use keys

Together, indices and keys form paths to all values in JSON



```
{
    "artwork": [
        {
            "Title": "Sunflower Seeds",
            "Artist": "Weiwei",
            "Exhibits": [6, 1, 5]
        }
    ]
}
```

$$\text{'\{artwork,0,Title\}'} \longrightarrow \text{'Sunflower Seeds'}$$

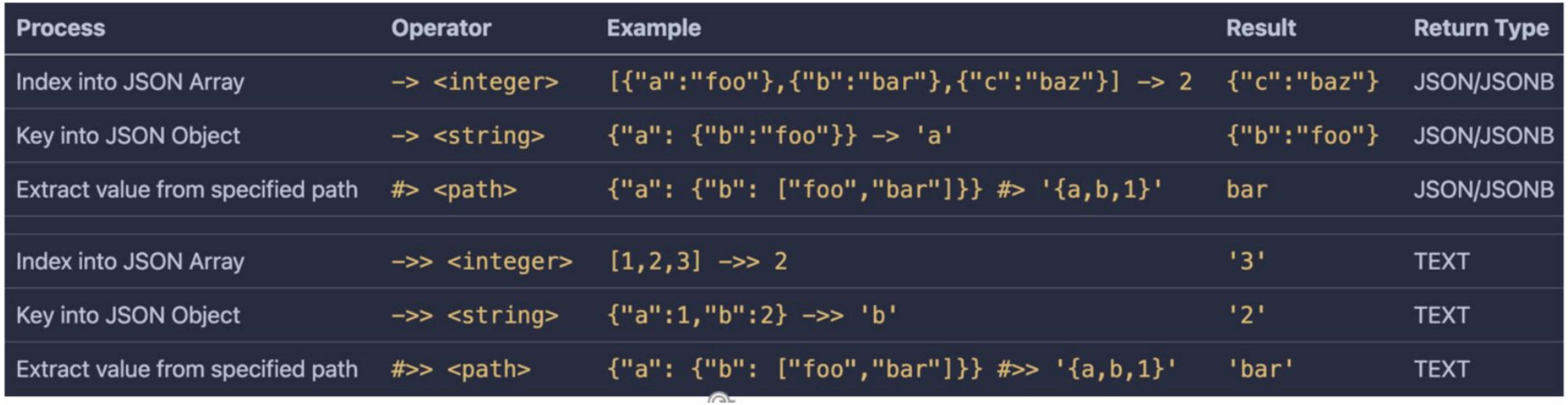
SQL with Python

JSON operators

Only work on JSON and JSONB data types

Use two greater-than signs to get value as TEXT

Other operators for advanced use cases are described in documentation



Process	Operator	Example	Result	Return Type
Index into JSON Array	`-> <integer>`	`[{"a":"foo"},{"b":"bar"},{"c":"baz"}] -> 2`	`{"c":"baz"}`	JSON/JSONB
Key into JSON Object	`-> <string>`	`{"a": {"b":"foo"}} -> 'a'`	`{"b":"foo"}`	JSON/JSONB
Extract value from specified path	`#> <path>`	`{"a": {"b": ["foo","bar"]}} #> '{a,b,1}'`	`bar`	JSON/JSONB
Index into JSON Array	`->> <integer>`	`[1,2,3] ->> 2`	`'3'`	TEXT
Key into JSON Object	`->> <string>`	`{"a":1,"b":2} ->> 'b'`	`'2'`	TEXT
Extract value from specified path	`#>> <path>`	`{"a": {"b": ["foo","bar"]}} #>> '{a,b,1}'`	`'bar'`	TEXT

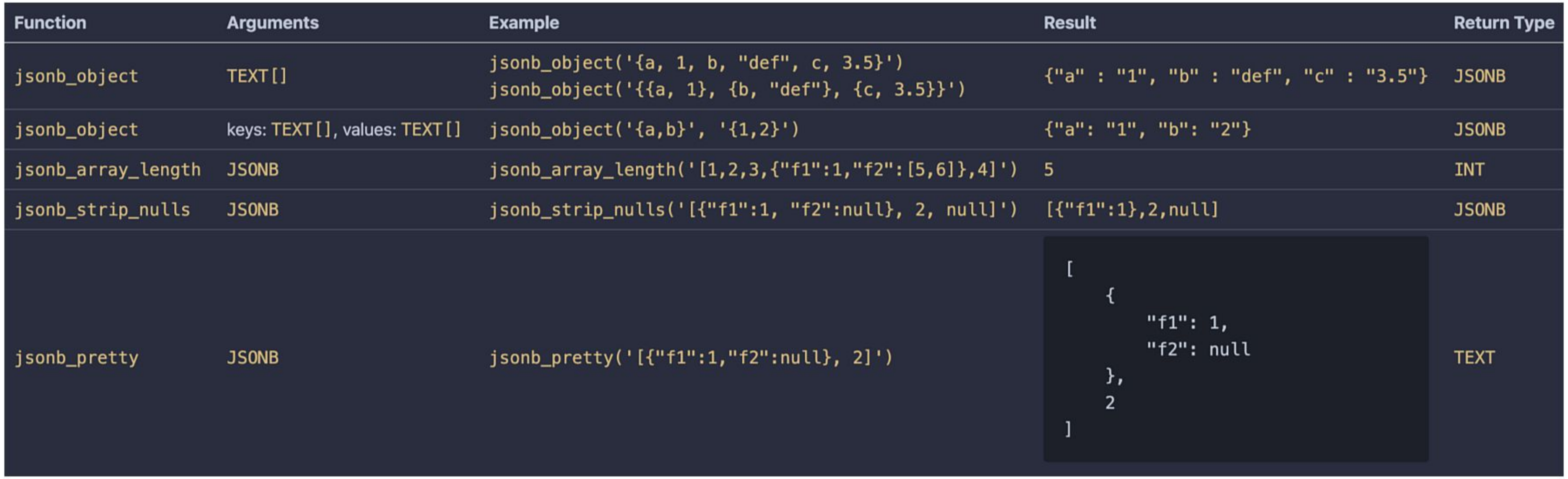
JSON functions

nucamp

Equivalent functions exist for regular JSON type

3 ways to build a JSONB object using jsonb_object()

jsonb_strip_nulls() recursively removes null entries from object



Function	Arguments	Example	Result	Return Type

jsonb_object	TEXT[]	jsonb_object('{a, 1, b, "def", c, 3.5}')
		jsonb_object('{{a, 1}, {b, "def"}, {c, 3.5}}')
			{"a" : "1", "b" : "def", "c" : "3.5"}	JSONB

jsonb_object	keys: TEXT[], values: TEXT[]	jsonb_object('{a,b}', '{1,2}')	{"a": "1", "b": "2"}	JSONB
jsonb_array_length	JSONB	jsonb_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4]')	5	INT
jsonb_strip_nulls	JSONB	jsonb_strip_nulls('[{"f1":1, "f2":null}, 2, null]')	[{"f1":1},2,null]	JSONB

jsonb_pretty	JSONB	jsonb_pretty('[{"f1":1,"f2":null}, 2]')	[
			    {
			        "f1": 1,
			        "f2": null
			    },
			    2
			]
				TEXT

When to use JSON?

Generally, only in special cases

Store unstructured, generic data in field

Integrate with other systems which use JSON

Co-locate data otherwise spread across many tables