aiven

PostgreSQL® JSONB Functions Cheatsheet

https://www.postgresql.org/docs/current/functions-json.html

@ftisiot

Dataset

```sql
create table test(id serial, json_data jsonb);
insert into test(json_data) values (
'{
    "id": 778,
    "shop": "Luigis Pizza",
    "name": "Edward Olson",
    "phoneNumbers":
        ["(935)503-3765x4154","(935)12345"],
    "address": "Unit 9398 Box 2056\nDPO AP 24022",
    "image": null,
    "pizzas": [
        {
            "pizzaName": "Salami",
            "additionalToppings": ["🥓", "🌶️"]
        },
        {
            "pizzaName": "Margherita",
            "additionalToppings": ["🍌", "🌶️", "🍍"]
        }
    ]
}');
```

Extract Fields

Extract JSON field

```sql
select
    json_data -> 'id' id,
    json_data -> 'name' name
    from test;
```

```
 id  |      name
-----+----------------
 778 | "Edward Olson"
```

Extract JSON field as text

```sql
select
    json_data ->> 'id' id,
    json_data ->> 'name' name
from test;
```

```
 id  |      name
-----+--------------
 778 | Edward Olson
```

Extract JSON subpath

```sql
select
    json_data #>
      '{pizzas,1,additionalToppings}'
        as additional_toppings_2nd_pizza
from test;
```

```
 additional_toppings_2nd_pizza
-----------------------------------
 ["🍌", "🌶️", "🍍"]
```

Extract item from array

```sql
select
    json_data
      -> 'phoneNumbers' -> 1
    as second_phonenumber
from test;
```

```
 second_phonenumber
--------------------
 "(935)12345"
```

Extract path

```sql
select
    jsonb_extract_path(json_data,
      'pizzas','1','pizzaName') second_pizza_name
from test;
```

```
 second_pizza_name
-------------------
 "Margherita"
```

Jsonpath query

Does JSON path return any item

```sql
select
    json_data
        @? '$.pizzas[1].pizzaName == "Salami"'
        as pizzaName_salami
from test;
```

```
 pizzaname_salami
------------------
 t
```

Does JSON path return any item

```sql
select
    jsonb_path_exists(json_data,
        '$.pizzas[*].pizzaName == "Salami"')
        as is_there_salami_pizza
from test;
```

```
 salami_pizza
--------------
 t
```

Return JSON path itemsany item

```sql
select
    jsonb_path_query(json_data,
      '$.pizzas[*] ? (@.pizzaName == "Salami")')
      as is_there_salami_pizza
from test;
```

```
                  is_there_salami_pizza
-----------------------------------------------------------
 {"pizzaName": "Salami", "additionalToppings": ["🥓", "🌶️"]}
```

Contains query

JSON A contains JSON B

```sql
select
    json_data
        @> '{"id": 778}'::jsonb
        as contains_id
from test;
```

```
 contains_id
-------------
 t
```

JSON A contains Item B

```sql
select
    json_data ? 'shop'
        as contains_shop
from test;
```

```
 contains_shop
---------------
 t
```

JSON A contains any item in B

```sql
select
    json_data
        ?| ARRAY['shop','cookies']
        as contains_shop_or_cookies
from test;
```

```
 contains_shop_or_cookies
--------------------------
 t
```

JSON A contains all items in B in B

```sql
select
    json_data
        ?& ARRAY['shop','cookies']
        as contains_shop_and_cookies
from test;
```

```
 contains_shop_and_cookies
---------------------------
 f
```

Create JSON Objects

Convert item/row to JSON

```sql
select
    to_jsonb(
      row(33,
        'the pizza is in the oven'::text));
```

```
                to_jsonb
------------------------------------------------
 {"f1": 33, "f2": "the pizza is in the oven"}
```

Create heterogeneously-typed JSON array

```sql
select
    jsonb_build_array(1,4,'🍌', 'pizza');
```

```
   jsonb_build_array
-----------------------
 [1, 4, "🍌", "pizza"]
```

Build JSON object from list of items

```sql
select
    jsonb_build_object(
        'name', 'francesco',
        'pizzas', ARRAY['Margherita','Diavola']);
```

```
                     jsonb_build_object
--------------------------------------------------------------
 {"name" : "francesco", "pizzas" : ["Margherita","Diavola"]}
```

Convert array to JSON array

```sql
select array_to_json(
        ARRAY['🍌', '🌶️', '🍍']);
```

```
 array_to_json
----------------
 ["🍌","🌶️","🍍"]
```

Build JSON object from text array

```sql
select
    jsonb_object(
      '{{name, francesco}, {pizza, "Margherita"}}');
```

```
                  jsonb_object
------------------------------------------------
 {"name" : "francesco", "pizza" : "Margherita"}
```

Build JSON object from key and value arrays

```sql
select
    jsonb_object(
        '{name, pizza}',
        '{francesco, Margherita}');
```

```
                  jsonb_object
------------------------------------------------
 {"name" : "francesco", "pizza" : "Margherita"}
```

Parse Arrays

Convert array elements to JSON rows

```sql
select p.*
from test
cross join lateral jsonb_array_elements(
    json_data -> 'phoneNumbers') p;
```

```
        value
----------------------
 "(935)503-3765x4154"
 "(935)12345"
```

Convert array elements to text rows

```sql
select p.*
from test
cross join lateral jsonb_array_elements_text(
    json_data -> 'phoneNumbers') p;
```

```
       value
--------------------
 (935)503-3765x4154
 (935)12345
```

Return array length

```sql
select
    jsonb_array_length(
                   json_data ->
'phoneNumbers')
from test;
```

```
 jsonb_array_length
--------------------
                  2
```

Types

Get item type

```sql
select
    jsonb_typeof(json_data->'id') type_id,
    jsonb_typeof(json_data->'name') type_name
from
    test;
```

```
 type_id | type_name
---------+-----------
 number  | string
```

Parse Keys

Return key/values as JSON rows

```sql
select p.*
from test
cross join lateral jsonb_each(json_data) p;
```

```
     key     |            value
-------------+----------------------------------
 id          | 778
 name        | "Edward Olson"
 shop        | "Luigis Pizza"
 pizzas      | [{"pizzaName": "Salami", "additionalToppings": ["🥓", "🌶️"]}, ...]
 address     | "Unit 9398 Box 2056\nDPO AP 24022"
 phoneNumbers | ["(935)503-3765x4154", "(935)12345"]
```

Return key/values as text rows

```sql
select p.*
from test
cross join lateral jsonb_each_text(json_data) p;
```

```
     key     |            value
-------------+----------------------------------
 id          | 778
 name        | Edward Olson
 shop        | Luigis Pizza
 pizzas      | [{"pizzaName": "Salami", "additionalToppings": ["🥓", "🌶️"]}, ...]
 address     | Unit 9398 Box 2056
             | DPO AP 24022
 phoneNumbers | ["(935)503-3765x4154", "(935)12345"]
```

Extract all keys

```sql
select
    jsonb_object_keys(json_data)
from test;
```

```
 jsonb_object_keys
-------------------
 id
 name
 shop
 pizzas
 address
 phoneNumbers
```

Prettify

Prettify JSON output

```sql
select
    jsonb_pretty(json_data->'pizzas')
from test;
```

```
            jsonb_pretty
-----------------------------------
 [                                +
     {                            +
         "pizzaName": "Salami",   +
         "additionalToppings": [  +
             "🥓",                 +
             "🌶️"                  +
         ]                        +
     },                           +
     ...                          +
 ]
```

Tabulate

Extract to record based on type

```sql
create type pizzaOrder
as (id int, name text, address text);
select p.*
from test cross join lateral
    jsonb_populate_record(
        null::pizzaOrder, json_data) p;
```

```
 id  |     name     |      address
-----+--------------+--------------------
 778 | Edward Olson | Unit 9398 Box 2056+
     |              | DPO AP 24022
```

Extract to recordset based on type

```sql
create type pizza as ("pizzaName" text,
"additionalToppings" text[]);
select p.* from test
cross join lateral
    jsonb_populate_recordset(
        null::pizza, json_data -> 'pizzas') p;
```

```
 pizzaName  | additionalToppings
------------+--------------------
 Salami     | {🥓,🌶️}
 Margherita | {🍌,🌶️,🍍}
```

Extract to record declaring columns

```sql
select p.* from test
cross join lateral
    jsonb_to_record(json_data)
    as p(id int, "phoneNumbers" text[]);
```

```
 id  |           phoneNumbers
-----+----------------------------------
 778 | {(935)503-3765x4154,(935)12345}
```

Extract to recordset declaring columns

```sql
select p.* from test
cross join lateral
    jsonb_to_recordset(json_data -> 'pizzas')
    as p("pizzaName" text,
              "additionalToppings" text[]);
```

```
 pizzaName  | additionalToppings
------------+--------------------
 Salami     | {🥓,🌶️}
 Margherita | {🍌,🌶️,🍍}
```

Concat JSON A and JSON B

```sql
select
    json_data
        || '{"note":"leave outside"}'
        as add_note
from test;
```

```
           add_note
--------------------------------
 {"id": 778,
  "name": "Edward Olson",
  "note": "leave outside",
  "shop": "Luigis Pizza",...}
```

Edit

Remove items in A

```sql
select
    json_data
        - ARRAY['pizzas','id']
        as no_pizzas_and_id
from test;
```

```
          no_pizzas_and_id
----------------------------------
{"name": "Edward Olson",
 "shop": "Luigis Pizza",
 "address": "Unit 9398 Box 2056\nDPO AP 24022",
 "phoneNumbers": ["(935)503-3765x4154", "(935)12345"]}
```

Remove specified path

```sql
select
    json_data
        #- '{pizzas,1,additionalToppings}'
        as no_2nd_pizza_additionalToppings
from test;
```

```
      no_2nd_pizza_additionaltoppings
-------------------------------------------
 {"id": 778,
  "name": "Edward Olson",
  "shop": "Luigis Pizza",
  "pizzas": [
    {"pizzaName": "Salami", additionalToppings": ["🥓", "🌶️"]},
    {pizzaName": "Margherita"}], ...}
```

Remove item # from array

```sql
select
  (json_data -> 'phoneNumbers') - 1
        as no_first_phone_number
from test;
```

```
  no_first_phone_number
-------------------------
 ["(935)503-3765x4154"]
```

Add/modify items

```sql
select
    jsonb_set(json_data -> 'pizzas',
        '{0,"pizzaName"}',
        to_jsonb('4 Stagioni'::text), false)
        as change_first_pizza_name
from test;
```

```
                         change_first_pizza_name
--------------------------------------------------------------------------
[{"pizzaName": "4 Stagioni", "additionalToppings": ["🥓", "🌶️"]},
 {"pizzaName": "Margherita", "additionalToppings": ["🍌", "🌶️", "🍍"]}]
```

Remove nulls

```sql
select
    jsonb_strip_nulls(json_data) no_nulls
from test;
```

```
The JSON document doesn't contain the image
field which was null
```

Insert items

```sql
select
    jsonb_insert(json_data,
        '{"phoneNumbers",0}',
        to_jsonb('12345'::text), false)
        as add_first_phone_number
from test;
```

```
                        add_first_phone_number
----------------------------------------------------------------------
 {"id": 778, "name": "Edward Olson", "shop": "Luigis Pizza",...,
"phoneNumbers": ["12345", "(935)503-3765x4154", "(935)12345"]}
```