**1.**

CREATE EXTENSION citus;

CREATE TABLE json\_table (

id VARCHAR(50) PRIMARY KEY,

type VARCHAR(50),

name VARCHAR(50),

ppu DECIMAL,

batters JSONB,

topping JSONB

);

INSERT INTO json\_table

SELECT \* FROM json\_populate\_recordset (NULL::json\_table,

'[

{

"id": "0001",

"type": "donut",

"name": "Cake",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" },

{ "id": "1002", "type": "Chocolate" },

{ "id": "1003", "type": "Blueberry" },

{ "id": "1004", "type": "Devil''s Food" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5005", "type": "Sugar" },

{ "id": "5007", "type": "Powdered Sugar" },

{ "id": "5006", "type": "Chocolate with Sprinkles" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

},

{

"id": "0002",

"type": "donut",

"name": "Raised",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5005", "type": "Sugar" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

},

{

"id": "0003",

"type": "donut",

"name": "Old Fashioned",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" },

{ "id": "1002", "type": "Chocolate" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

}

]');

CREATE TABLE product (

id VARCHAR(50),

type VARCHAR(50),

name VARCHAR(50),

ppu DECIMAL,

batter\_id VARCHAR(50),

batter\_type VARCHAR(100),

topping\_id VARCHAR(50),

topping\_type VARCHAR(100)

)

USING columnar;

INSERT INTO product

SELECT jt.id,jt.type,jt.name,jt.ppu

,j.batter::jsonb -> 'id' as batter\_id

,j.batter::jsonb -> 'type' as batter\_type

,k.topping::jsonb -> 'id' as topping\_id

,k.topping::jsonb -> 'type' as topping\_type

FROM json\_table jt

CROSS JOIN LATERAL

jsonb\_array\_elements\_text(

batters->'batter'

) AS j(batter)

CROSS JOIN LATERAL

jsonb\_array\_elements\_text(

topping

) AS k(topping)

**2.**

create\_json\_table

insert\_data\_json\_table

create\_columnar\_table

insert\_data\_columnar\_table

import airflow

from datetime import timedelta

from airflow import DAG

from airflow.operators.postgres\_operator import PostgresOperator

from airflow.utils.dates import days\_ago

dag\_psql = DAG(

dag\_id = "jsontocolumnar",

default\_args=args,

schedule\_interval='@once',

dagrun\_timeout=timedelta(minutes=60),

start\_date = airflow.utils.dates.days\_ago(1)

)

create\_json\_table\_sql\_query = """

CREATE TABLE json\_table (

id VARCHAR(50) PRIMARY KEY,

type VARCHAR(50),

name VARCHAR(50),

ppu DECIMAL,

batters JSONB,

topping JSONB

);

"""

insert\_data\_json\_sql\_query = """

INSERT INTO json\_table

SELECT \* FROM json\_populate\_recordset (NULL::json\_table,

'[

{

"id": "0001",

"type": "donut",

"name": "Cake",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" },

{ "id": "1002", "type": "Chocolate" },

{ "id": "1003", "type": "Blueberry" },

{ "id": "1004", "type": "Devil''s Food" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5005", "type": "Sugar" },

{ "id": "5007", "type": "Powdered Sugar" },

{ "id": "5006", "type": "Chocolate with Sprinkles" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

},

{

"id": "0002",

"type": "donut",

"name": "Raised",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5005", "type": "Sugar" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

},

{

"id": "0003",

"type": "donut",

"name": "Old Fashioned",

"ppu": 0.55,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" },

{ "id": "1002", "type": "Chocolate" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

}

]');

"""

create\_columnar\_table\_sql\_query = """

CREATE TABLE product (

id VARCHAR(50),

type VARCHAR(50),

name VARCHAR(50),

ppu DECIMAL,

batter\_id VARCHAR(50),

batter\_type VARCHAR(100),

topping\_id VARCHAR(50),

topping\_type VARCHAR(100)

)

USING columnar;

"""

insert\_data\_columnar\_sql\_query = """

INSERT INTO product

SELECT jt.id,jt.type,jt.name,jt.ppu

,j.batter::jsonb -> 'id' as batter\_id

,j.batter::jsonb -> 'type' as batter\_type

,k.topping::jsonb -> 'id' as topping\_id

,k.topping::jsonb -> 'type' as topping\_type

FROM json\_table jt

CROSS JOIN LATERAL

jsonb\_array\_elements\_text(

batters->'batter'

) AS j(batter)

CROSS JOIN LATERAL

jsonb\_array\_elements\_text(

topping

) AS k(topping)

"""

create\_json\_table = PostgresOperator(

sql = create\_json\_table\_sql\_query,

task\_id = "create\_json\_table",

postgres\_conn\_id = "postgres\_local",

dag = dag\_psql )

insert\_data\_json\_table = PostgresOperator(

sql = insert\_data\_json\_sql\_query,

task\_id = "insert\_data\_json\_table",

postgres\_conn\_id = "postgres\_local",

dag = dag\_psql )

create\_columnar\_table = PostgresOperator(

sql = create\_columnar\_table\_sql\_query,

task\_id = "create\_columnar\_table",

postgres\_conn\_id = "postgres\_local",

dag = dag\_psql )

insert\_data\_columnar\_table = PostgresOperator(

sql = insert\_data\_columnar\_sql\_query,

task\_id = "insert\_data\_columnar\_table",

postgres\_conn\_id = "postgres\_local",

dag = dag\_psql )

create\_json\_table >> insert\_data\_json\_table >> create\_columnar\_table >> insert\_data\_columnar\_table

**3.**

import request

user = 'your\_email','your\_password'

url = 'https://your\_zendesk\_url/api/v2/groups.json'

response = requests.get(url, auth=user)

if response.status\_code != 200:

print('Status:', response.status\_code, '.Error')

exit()

data = response.json()

group\_list = data['groups']

for group in group\_list:

print(group['name'])



Example source : Zendesk

**4.**

|  |
| --- |
|  |
| #4.1 |
|  |  |
|  | #Input jumlah N |
|  | j = int(input('N: ')) |
|  | count = 0 |
|  | a = 1 |
|  |  |
|  | #define prime number |
|  | def is\_prime\_number (x): |
|  | for i in range(2, x): |
|  | if x % i == 0: |
|  | return False |
|  | return True |
|  |  |
|  | prime\_number\_list = [] |
|  | while count < j: |
|  | a = a + 1 |
|  | if is\_prime\_number(a): |
|  | prime\_number\_list.append(a) |
|  | count = count + 1 |
|  |  |
|  |  |
|  | print(prime\_number\_list) |
|  |  |
| #4.2 |  |
|  |  |
|  | # Input Kedua List |
|  | j1 = input('list\_1: ') |
|  | k1 = input('list\_2: ') |
|  |  |
|  | #Sort Kedua List |
|  | j2 = sorted(j1.split(",")) |
|  | k2 = sorted(k1.split(",")) |
|  |  |
|  | #Cek hitung jumlah List |
|  | count\_j = len(j2) |
|  | count\_k = len(k2) |
|  |  |
|  | #Add null ke list yg jumlahnya lebih sedikit |
|  | while count\_j < count\_k: |
|  | j2.append("NULL") |
|  | count\_j = count\_j+1 |
|  |  |
|  | while count\_k < count\_j: |
|  | k2.append("NULL") |
|  | count\_k = count\_k+1 |
|  |  |
|  | #Zip 2 list |
|  | new\_lst = [list(x) for x in zip(j2, k2)] |
|  | print(new\_lst) |
|  |  |