This document provides help on how to capture the solution\_treatment/post\_processing/solution\_processing order information

1. **Create the table first**

-- Create the SOLUTION\_TREATMENT\_SEQUENCE table

CREATE TABLE IF NOT EXISTS SOLUTION\_TREATMENT\_SEQUENCE (

solution\_treatment\_id SERIAL PRIMARY KEY,

solution\_treatment\_sequence integer[],

FOREIGN KEY (solution\_treatment\_id) REFERENCES SOLUTION\_TREATMENT (solution\_treatment\_id)

);

1. **Populate the sequence table**

-- Insert data into the SOLUTION\_TREATMENT\_SEQUENCE table

INSERT INTO SOLUTION\_TREATMENT\_SEQUENCE (solution\_treatment\_id, solution\_treatment\_sequence)

SELECT

solution\_treatment\_id,

ARRAY\_AGG(solution\_treatment\_step\_id) AS solution\_treatment\_sequence

FROM

SOLUTION\_TREATMENT\_ORDER

GROUP BY

solution\_treatment\_id;

1. **Fix code :**

import json

import psycopg2

# Function to insert data into SOLUTION\_TREATMENT\_STEP table

def insert\_into\_solution\_treatment\_step(cur, treatment\_type, params, meta):

# Check if the record already exists

cur.execute(

"SELECT solution\_treatment\_step\_id FROM SOLUTION\_TREATMENT\_STEP WHERE treatment\_type = %s AND params = %s::jsonb AND meta = %s::jsonb",

(treatment\_type, params, meta)

)

existing\_id = cur.fetchone()

if existing\_id:

solution\_treatment\_step\_id = existing\_id[0]

else:

#Insert data into SOLUTION\_TREATMENT\_STEP table

cur.execute(

"INSERT INTO SOLUTION\_TREATMENT\_STEP (treatment\_type, params, meta) VALUES (%s, %s::jsonb, %s::jsonb) RETURNING solution\_treatment\_step\_id",

(treatment\_type, params, meta)

)

solution\_treatment\_step\_id = cur.fetchone()[0]

return solution\_treatment\_step\_id

# Function to insert data into SOLUTION\_TREATMENT\_ORDER and SOLUTION\_TREATMENT\_SEQUENCE tables

def insert\_into\_solution\_treatment\_order\_and\_sequence(cur, step\_ids, process\_order\_list):

# Check if the combination of step IDs already exists in the SOLUTION\_TREATMENT\_SEQUENCE table

cur.execute(

"""

SELECT solution\_treatment\_id

FROM SOLUTION\_TREATMENT\_SEQUENCE

WHERE solution\_treatment\_sequence = %s

""",

(step\_ids,)

)

existing\_id = cur.fetchone()

if existing\_id:

# Use the existing solution\_treatment\_id

solution\_treatment\_id = existing\_id[0]

else:

# Insert a new solution\_treatment\_id into SOLUTION\_TREATMENT table

cur.execute(

"""

INSERT INTO SOLUTION\_TREATMENT (solution\_treatment\_id)

VALUES (nextval('solution\_treatment\_solution\_treatment\_id\_seq'))

RETURNING solution\_treatment\_id;

"""

)

solution\_treatment\_id = cur.fetchone()[0]

# Insert data into SOLUTION\_TREATMENT\_ORDER table

data = [(solution\_treatment\_id, step\_id, process\_order) for step\_id, process\_order in zip(step\_ids, process\_order\_list)]

cur.executemany(

"""

INSERT INTO SOLUTION\_TREATMENT\_ORDER (solution\_treatment\_id, solution\_treatment\_step\_id, process\_order)

VALUES (%s, %s, %s);

""",

data

)

# Insert the sequence into SOLUTION\_TREATMENT\_SEQUENCE table

cur.execute(

"""

INSERT INTO SOLUTION\_TREATMENT\_SEQUENCE (solution\_treatment\_id, solution\_treatment\_sequence)

VALUES (%s, %s);

""",

(solution\_treatment\_id, step\_ids)

)

return solution\_treatment\_id # Return the solution\_treatment\_id, whether existing or newly generated

# Establish a connection to the PostgreSQL database

conn = psycopg2.connect(\*\*param\_dict)

# Create a cursor object to interact with the database

cur = conn.cursor()

# Initialize solution\_treatment\_id

solution\_treatment\_id = None

# Initialize lists to store step IDs and process orders for this combination

step\_ids = []

process\_order\_list = []

# Check if solution\_processing is empty

if solution\_processing:

for treatment in solution\_processing.values():

# Convert params and meta to JSON format

params\_json = json.dumps(treatment.get('params', {}))

meta\_json = json.dumps(treatment.get('meta', {}))

treatment\_type = treatment['treatment\_type']

# Insert data into SOLUTION\_TREATMENT\_STEP table

solution\_treatment\_step\_id = insert\_into\_solution\_treatment\_step(cur, treatment\_type, params\_json, meta\_json)

# Append step ID and process order to the lists

step\_ids.append(solution\_treatment\_step\_id)

process\_order\_list.append(treatment['process\_step'])

# Insert data into SOLUTION\_TREATMENT, SOLUTION\_TREATMENT\_ORDER and SOLUTION\_TREATMENT\_SEQUENCE table

solution\_treatment\_id = insert\_into\_solution\_treatment\_order\_and\_sequence(cur, step\_ids, process\_order\_list)

# Commit the changes to the database

print("Solution treatment saved successfully with id:", solution\_treatment\_id)

conn.commit()

else:

print("solution\_processing is empty. No database operations performed.")

# Close the cursor and connection

cur.close()

conn.close()