import os

import re

import json

from typing import List, Dict, Optional

from supabase import create\_client, Client

from dotenv import load\_dotenv

from datetime import datetime, timezone

def get\_filtered\_notices(

supabase: Client,

active: bool = True,

include\_naics: Optional[List[str]] = None,

exclude\_naics: Optional[List[str]] = None,

include\_solicitation\_types: Optional[List[str]] = None,

exclude\_solicitation\_types: Optional[List[str]] = None,

include\_psc: Optional[List[str]] = None,

exclude\_psc: Optional[List[str]] = None,

include\_set\_aside\_ids: Optional[List[str]] = None, # new filter: include by set\_aside\_id

exclude\_set\_aside\_ids: Optional[List[str]] = None, # new filter: exclude by set\_aside\_id

keyword\_query: Optional[str] = None

) -> List[Dict[str, any]]:

"""

Retrieve rows from the 'notices' table applying the provided filters and paginating through

all the available data.

"""

all\_notices = []

limit = 1000 # Batch size for pagination.

offset = 0

while True:

# Build the base query with embedded joins.

query = supabase.from\_("notices").select("""

\*,

naics\_details:naics!naics\_id(\*),

psc\_details:psc!psc\_id(\*),

setasides\_details:setasides!set\_aside\_id(\*),

solicitations\_details:solicitations!solicitation\_id(\*),

addresses\_details:addresses!organization\_address\_key(\*),

organization\_details:organizations!Notices\_organization\_key\_fkey(\*),

organization\_level\_1\_details:organizations!Notices\_organization\_level\_1\_key\_fkey(\*),

organization\_level\_2\_details:organizations!Notices\_organization\_level\_2\_key\_fkey(\*),

organization\_level\_3\_details:organizations!Notices\_organization\_level\_3\_key\_fkey(\*),

organization\_level\_4\_details:organizations!Notices\_organization\_level\_4\_key\_fkey(\*),

organization\_level\_5\_details:organizations!Notices\_organization\_level\_5\_key\_fkey(\*),

organization\_level\_6\_details:organizations!Notices\_organization\_level\_6\_key\_fkey(\*),

organization\_level\_7\_details:organizations!Notices\_organization\_level\_7\_key\_fkey(\*),

solicitation\_type\_details:solicitation\_types!type(\*)

""")

if active:

print("Applying active filter")

query = query.eq("latest", True)

current\_time = datetime.now(timezone.utc).isoformat()

print("Current time:", current\_time)

# Only include notices where the solicitation\_response\_deadline is in the future.

query = query.gt("solicitation\_response\_deadline", current\_time)

# Apply include filters for NAICS codes.

if include\_naics:

print("Including NAICS codes:", include\_naics)

query = query.in\_("naics", include\_naics)

# Apply exclude filters for NAICS codes.

if exclude\_naics:

for code in exclude\_naics:

query = query.neq("naics", code)

# Apply filters for solicitation types.

if include\_solicitation\_types:

query = query.in\_("type", include\_solicitation\_types)

if exclude\_solicitation\_types:

print("Excluding solicitation types: ", exclude\_solicitation\_types)

for s\_type in exclude\_solicitation\_types:

query = query.neq("type", s\_type)

# Apply filters for PSC codes.

if include\_psc:

query = query.in\_("psc", include\_psc)

if exclude\_psc:

for psc in exclude\_psc:

query = query.neq("psc", psc)

# Apply filters for set aside IDs from the joined setasides table.

if include\_set\_aside\_ids:

# Convert provided set\_aside\_ids to integer type to match int8 column in database.

include\_set\_aside\_ids = [int(x) for x in include\_set\_aside\_ids]

print("Including set\_aside\_ids:", include\_set\_aside\_ids)

query = query.in\_("setasides\_details.set\_aside\_id", include\_set\_aside\_ids)

if exclude\_set\_aside\_ids:

# Convert provided set\_aside\_ids to integer type to match int8 column in database.

exclude\_set\_aside\_ids = [int(x) for x in exclude\_set\_aside\_ids]

print("Excluding set\_aside\_ids:", exclude\_set\_aside\_ids)

for set\_aside\_id in exclude\_set\_aside\_ids:

query = query.neq("setasides\_details.set\_aside\_id", set\_aside\_id)

# Apply an advanced keyword search on text fields.

if keyword\_query:

print("Applying keyword search filter:", keyword\_query)

# Here we use a full-text search on the "description" column.

# Adjust the column name if your text content is stored in a different field.

query = query.text\_search("opportunity\_text", keyword\_query, {'config': 'english', 'type': 'websearch'})

# Apply pagination for the current batch using the Range header.

query = query.range(offset, offset + limit - 1)

result = query.execute()

# If no more data is returned, exit the loop.

if not result.data:

break

all\_notices.extend(result.data)

offset += limit

return all\_notices

def main():

load\_dotenv()

SUPABASE\_URL = os.getenv("SUPABASE\_URL")

SUPABASE\_KEY = os.getenv("SUPABASE\_KEY")

if not SUPABASE\_URL or not SUPABASE\_KEY:

raise ValueError("SUPABASE\_URL and SUPABASE\_KEY environment variables must be set")

supabase: Client = create\_client(SUPABASE\_URL, SUPABASE\_KEY)

# Define filter parameters.

active = True

#active = False

include\_naics = [

"237310", "488111", "493110", "518210", "541330", "541511", "541512", "541519",

"541611", "541614", "562910", "513210", "516210", "561210", "221330", "236210",

"236220", "237110", "237990", "238110", "238350", "238910", "238990", "333922",

"541219", "541350", "541360", "541370", "541380", "541430", "541513", "541612",

"541618", "541620", "541690", "541713", "541714", "541715", "541810", "541820",

"541830", "541840", "541850", "541860", "541870", "541890", "541990", "561410",

"561421", "561499", "561730", "561790", "611430"

]

exclude\_naics = []

include\_solicitation\_types = [] # types to include

#exclude\_solicitation\_types = []

exclude\_solicitation\_types = ["s", "a", "u", "j", "l", "m", "g", "f"] # types to exclude

exclude\_psc = []

include\_psc = []

# Filters for set aside codes and IDs (from the joined setasides table).

include\_set\_aside\_ids = []

#exclude\_set\_aside\_ids = []

exclude\_set\_aside\_ids = [16, 24, 17, 18, 19, 20, 22, 21, 23 ] # example set\_aside\_id values to exclude

keyword\_query = "ITAD or media destruction or digital media destruction or LTO tape destruction or hard drive destruction or mobile device destruction or obsolete digital media disposal or asset disposition or secure data sanitization or IT asset disposal or data destruction or shredding or document destruction or destruction service"

# Retrieve notices with the new filtering options.

notices = get\_filtered\_notices(

supabase=supabase,

active=active,

include\_naics=include\_naics,

exclude\_naics=exclude\_naics,

include\_solicitation\_types=include\_solicitation\_types,

exclude\_solicitation\_types=exclude\_solicitation\_types,

include\_psc=include\_psc,

exclude\_psc=exclude\_psc,

include\_set\_aside\_ids=include\_set\_aside\_ids,

exclude\_set\_aside\_ids=exclude\_set\_aside\_ids,

keyword\_query=keyword\_query

)

#print(json.dumps(notices, indent=2))

print("Number of results: ", len(notices))

with open("wnzTPS5NfNK5vVqRhbQ9i\_results.json", "w") as f:

json.dump(notices, f, indent=2, default=str)

print("Saved results to wnzTPS5NfNK5vVqRhbQ9i\_result.json file")

if \_\_name\_\_ == "\_\_main\_\_":

main()