import itertools

import multiprocessing

import random

import time

def generate\_query\_plans():

    tables = ['tableA', 'tableB', 'tableC', 'tableD', 'tableE', 'tableF']

    columns = ['columnA', 'columnB', 'columnC', 'columnD', 'columnE', 'columnF']

    join\_types = ['INNER JOIN', 'LEFT JOIN', 'RIGHT JOIN', 'FULL OUTER JOIN']

    where\_operators = ['=', '>', '<', '>=', '<=', '<>', 'LIKE']

    query\_plans = []

    for i in range(500):

        tables\_subset = random.sample(tables, random.randint(2, 5))

        columns\_subset = random.sample(columns, random.randint(2, 5))

        join\_subset = random.sample(join\_types, len(tables\_subset) - 1)

        where\_subset = []

        for column in columns\_subset:

            where\_subset.append((column, random.choice(where\_operators), random.randint(1, 100)))

        # Convert the query execution plan to a SQL statement

        query = 'SELECT ' + ', '.join(columns\_subset) + ' FROM ' + tables\_subset[0]

        for i in range(len(join\_subset)):

            query += ' ' + join\_subset[i] + ' ' + tables\_subset[i + 1] + ' ON ' + random.choice(columns\_subset) + ' = ' + random.choice(columns\_subset)

        if where\_subset:

            query += ' WHERE ' + ' AND '.join([f"{x[0]} {x[1]} {x[2]}" for x in where\_subset])

        query\_plans.append(query)

    return query\_plans

def optimize\_query(query):

    time.sleep(random.random())

    return (query, random.random())

def optimize\_queries\_parallel(query\_plans):

    num\_processes = multiprocessing.cpu\_count()

    query\_subsets = [query\_plans[i::num\_processes] for i in range(num\_processes)]

    pool = multiprocessing.Pool(processes=num\_processes)

    optimized\_query\_plans = []

    for i in range(num\_processes):

        optimized\_query\_subsets = pool.map(optimize\_query, query\_subsets[i])

        optimized\_query\_plans += optimized\_query\_subsets

    pool.close()

    pool.join()

    optimized\_query\_plans.sort(key=lambda x: x[1])

    return optimized\_query\_plans[0][0]

if \_\_name\_\_ == '\_\_main\_\_':

    query\_plans = generate\_query\_plans()

    print("Query plans:")

    for query in query\_plans:

        print(query)

    optimized\_query = optimize\_queries\_parallel(query\_plans)

    print("\nOptimized query:", optimized\_query)