```
#!/usr/bin/env python3
import psycopg2

#####################################################
##  Database Connection
#####################################################

'''
Connect to the database using the connection string
'''

def openConnection():
    # connection parameters - ENTER YOUR LOGIN AND PASSWORD HERE

    userid = "y21s2c9120_yimo6410"
    passwd = "510202271"
    myHost = "soit-db-pro-2.ucc.usyd.edu.au"

    # Create a connection to the database
    conn = None
    try:
        # Parses the config file and connects using the connect string
        conn = psycopg2.connect(database=userid,
                                user=userid,
                                password=passwd,
                                host=myHost)
    except psycopg2.Error as sqle:
        print("psycopg2.Error : " + sqle.pgerror)

    # return the connection to use
    return conn


'''
Validate user login request based on username and password
'''

def checkUserCredentials(username, password):
    # open connections
    conn = openConnection()
    # create cursor
    curs = conn.cursor()
    # '-' is not allowed to login
    if username == '-':
        return None

    # return official info that meet given username and password in oficial table
    query = "SELECT * FROM official WHERE username = %s AND password = %s"
    # pass username and password to query
    curs.execute(query, (username, password))

    # return result
    userInfo  = curs.fetchone()

    # close cursor and connection and transaction
    conn.commit()

    curs.close()
    conn.close()
    return userInfo


'''
List all the associated events in the database for a given official
'''

def findEventsByOfficial(official_id):
    conn = openConnection()
    curs = conn.cursor()

    curs.execute("BEGIN;")
    # call function in sql with official_id
    curs.callproc('SearchAssociatedEvents', [int(official_id)])
    event_db = curs.fetchall()
    event_list = [{
        'event_id': str(row[0]),
        'event_name': row[1],
        'sport': row[2],
        'referee': row[3],
        'judge': row[4],
        'medal_giver': row[5]
    } for row in event_db]

    # close cursor and connection and transaction
    conn.commit()

    curs.close()
    conn.close()
    return event_list


'''
Find a list of events based on the searchString provided as parameter
See assignment description for search specification
'''

def findEventsByCriteria(searchString):
    conn = openConnection()
    curs = conn.cursor()

    curs.execute("BEGIN;")
    # call function in sql with searchString
    curs.callproc('SearchSpecifiedEvents', [str(searchString)])
    event_db = list(curs.fetchall())
    event_list = [{
        'event_id': str(row[0]),
        'event_name': row[1],
        'sport': row[2],
        'referee': row[3],
        'judge': row[4],
        'medal_giver': row[5]
    } for row in event_db]

    # close cursor and connection and transaction
    conn.commit()

    curs.close()
    conn.close()
    return event_list


'''
Add a new event
'''

'''
Since we input strings, such as sport, as sportname, but we end up storing it in a
table as sportid. We need to find these names in other tables first and convert
them to the corresponding IDs. If the corresponding id is not found, an error is
reported.
'''
def addEvent(event_name, sport, referee, judge, medal_giver):
    conn = openConnection()
    curs = conn.cursor()

    # use str.replace() to change "'" to "''" to avoid error
    event_name = event_name.replace("'", "''")

    curs.execute("BEGIN;")
    curs.callproc('searchIds', [str(sport), str(referee), str(judge),
str(medal_giver)])

    res = curs.fetchone()
    # If the value of the new "add" does not exist in the original table, it will
report invalid
    if None in res:
        return False

    query = "INSERT INTO event (eventname, sportid, referee, judge, medalgiver)
VALUES ('%s', %d, %d, %d, %d)"% (event_name, res[0], res[1], res[2], res[3])
    curs.execute(query)

    # close cursor and connection and transaction
    conn.commit()

    curs.close()
    conn.close()

    return True


'''
Update an existing event
'''

'''
Similar to "add"
'''
def updateEvent(event_id, event_name, sport, referee, judge, medal_giver):
    conn = openConnection()
    curs = conn.cursor()

    # use str.replace() to change "'" to "''" to avoid error
    event_name = event_name.replace("'", "''")

    curs.execute("BEGIN;")
    curs.callproc('searchIds', [str(sport), str(referee), str(judge),
str(medal_giver)])

    res = curs.fetchone()
    if None in res:
        return False

    query = "UPDATE event SET eventname = '%s', sportid = %d, referee = %d, judge = 
%d, medalgiver = %d WHERE eventid = %d " % (event_name, res[0], res[1], res[2], 
res[3], int(event_id))
    curs.execute(query)

    # close cursor and connection and transaction
    conn.commit()

    curs.close()
    conn.close()

    return True
```