mini-project-1

Mini Project - Implement Parallelization of Database Query optimization

[ ]:
```
from multiprocessing import Process,freeze_support, Pool
import psycopg2
import time
import os
import pandas as pd
import datetime,random
import concurrent.futures
```

[ ]:
```
USER = "user_1"
DB = "test_1"
PASS = "postgres"
HOST = "127.0.0.1"
PORT = 5432
df = None
```

[ ]:
```
def serial_read():
    start_time = time.time()
    conn = psycopg2.connect(database = DB, user = USER, password = PASS, host = HOST, port= PORT)
    cursor = conn.cursor()
    cursor.execute("SELECT * from employees.employee ORDER BY id ASC")
    records = cursor.fetchall()
    conn.close()
    print(f"Execution time for serial read:{round(time.time() - start_time,3)} s")
    df = pd.DataFrame(records,columns=["ID","Birth Date", "First Name", "Last Name", "Gender", "Hire Date"])
    return df
```

[ ]:
```
df = serial_read()
df
```

Execution time for serial read:0.477 s

[ ]:

	ID	Birth Date	First Name	Last Name	Gender	Hire Date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
...	...	...	...	...	...	...
341029	541005	2007-09-27	{gap	dsbp	M	2007-10-15
341030	541006	2021-01-21	q	sgbhxa	F	1975-06-06
341031	541007	1981-10-16	ccbmn	ylbpo	F	2009-12-23
341032	541008	2022-08-15	e	niaef	F	2014-09-19
341033	541009	2009-07-12	ix	dwcxgb	F	2006-11-13

[341034 rows x 6 columns]

[ ]:
```
def execute_select():
    conn = psycopg2.connect(database = DB, user = USER, password = PASS, host = HOST, port= PORT)
    cursor = conn.cursor()
    cursor.execute("SELECT * from employees.employee ORDER BY id ASC")
    records = cursor.fetchall()
    cursor.close()
    return records

def parallel_read():
    start_time = time.time()
    records = []
    with concurrent.futures.ProcessPoolExecutor() as executor:
        proc = [executor.submit(execute_select)]
        for f in concurrent.futures.as_completed(proc):
            records.extend(f.result())

    print(f"Execution time for parallel read:{round(time.time() - start_time,3)} s")
#       print(records[])
    df = pd.DataFrame(records,columns=["ID","Birth Date", "First Name", "Last Name", "Gender", "Hire Date"])
    return df
```

[ ]:
```
df = parallel_read()
```

Execution time for parallel read:2.201 s

[ ]:
```
def random_date():
    d = random.randint(1, int(time.time()))
    return datetime.date.fromtimestamp(d).strftime('%Y-%m-%d')
```

```
def generate_name():
    length = random.randint(1,6)
    name = ""
    for i in range(length):
        j = random.randint(0,26)
        name += chr(97+j)
    return name
def create_record(id):
    id = int(id)
    seed = random.randint(0,1)
    gender = ""
    if seed == 0:
        gender = 'M'
    else:
        gender = 'F'
    query = """INSERT INTO employees.employee (id, birth_date, first_name,
 last_name, gender, hire_date) VALUES (%s,%s,%s,%s,%s,%s)"""
    values =
 (id,random_date(),generate_name(),generate_name(),gender,random_date())
    return query,values
def generate_records(n):
    records = []
    for i in range(n):
        records.append(create_record(int(df.iloc[df.shape[0] - 1,0]) + i + 1))
    return records
```

[ ]:
```
df.iloc[df.shape[0] - 1,0]
```

[ ]:
```
541009
```

[ ]:
```

```

[ ]:
```
def insert_serially(n):
    records = generate_records(n)
    orignal_size = df.shape[0]
    start_time = time.time()
    conn = psycopg2.connect(database = DB, user = USER, password = PASS, host =
 HOST, port= PORT)
    cursor = conn.cursor()
    for record in records:
        try:
            query, values = record
            cursor.execute(query,values)
            conn.commit()
        except Exception as e:
            pass
```

```
    print(f"Execution time for sequential insert:{round(time.time() - start_time,3)} s")
    print(f"{n} records inserted successfully")

    conn.close()
```

[ ]:
```
insert_serially(100000)
df = serial_read()
df
```

```
Execution time for sequential insert:341.939 s
100000 records inserted successfully
Execution time for serial read:0.526 s
```

[ ]:

	ID	Birth Date	First Name	Last Name	Gender	Hire Date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
...	...	...	...	...	...	...
441029	641005	2000-12-05	quxgx	w	F	1988-08-12
441030	641006	1974-04-14	wzvlzz	rrfqin	M	1986-02-15
441031	641007	1983-06-12	onftaa	q	M	1975-11-08
441032	641008	2004-01-30	nx	jqrie{	M	2018-09-23
441033	641009	1989-11-19	b	el	F	1984-10-31

[441034 rows x 6 columns]

[ ]:
```
def insert(query,values):
    conn = psycopg2.connect(database = DB, user = USER, password = PASS, host = HOST, port= PORT)
    cursor = conn.cursor()
    cursor.execute(query,values)
    conn.commit()
    conn.close()




def parallel_write(n):
    records = generate_records(n)
    orignal_size = df.shape[0]
    start_time = time.time()
    with concurrent.futures.ProcessPoolExecutor() as executor:
        proc = [executor.submit(insert,query=query,values=values) for query,values in records]
```

```
    print(f"Execution time for parallel insert:{round(time.time() - start_time,3)} s")
    conn = psycopg2.connect(database = DB, user = USER, password = PASS, host = HOST, port= PORT)
    cursor = conn.cursor()
    cursor.execute('select count(*) from employees.employee')
    rows = cursor.fetchall()
    if rows[0][0] - orignal_size == n:
        print(f"{n} records inserted successfully")

    conn.close()
```

[ ]:
```
parallel_write(100000)
df = parallel_read()
df
```

```
Execution time for parallel insert:143.09 s
100000 records inserted successfully
Execution time for parallel read:2.906 s
```

[ ]:

	ID	Birth Date	First Name	Last Name	Gender	Hire Date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
...	...	...	...	...	...	...
541029	741005	2009-04-28	mwqvpi	b	M	1973-12-25
541030	741006	2006-08-10	vkfs	i	F	1972-07-17
541031	741007	2012-06-15	rt{	kheuc	F	2010-12-19
541032	741008	1996-05-03	bejqz	b	F	1989-07-19
541033	741009	1977-11-05	k	wx	M	2022-03-05

[541034 rows x 6 columns]