**Day 11 Task Allocation**

**Python implementation of an Asset Inventory Management System for pan shops using PostgreSQL. This example includes connecting to a PostgreSQL database, creating tables, performing CRUD operations, and generating daily, weekly, monthly, and yearly reports.**

**Prerequisites**

Ensure you have the psycopg2 library installed to interact with PostgreSQL from Python. You can install it using:

**Bash Source Code**

**pip install psycopg2-binary**

**Python Script Source Code**

import psycopg2

from datetime import datetime, timedelta

import pandas as pd

# Database connection parameters

DB\_PARAMS = {

'dbname': 'inventory\_db',

'user': 'your\_username',

'password': 'your\_password',

'host': 'localhost',

'port': '5432'

}

def connect\_db():

""" Connect to the PostgreSQL database """

conn = psycopg2.connect(\*\*DB\_PARAMS)

return conn

def create\_tables():

""" Create tables in the PostgreSQL database """

conn = connect\_db()

cur = conn.cursor()

cur.execute("""

CREATE TABLE IF NOT EXISTS inventory (

id SERIAL PRIMARY KEY,

item\_name VARCHAR(255) NOT NULL,

quantity INT NOT NULL,

price DECIMAL NOT NULL,

date\_added TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

""")

conn.commit()

cur.close()

conn.close()

def add\_item(item\_name, quantity, price):

""" Add a new item to the inventory """

conn = connect\_db()

cur = conn.cursor()

cur.execute("""

INSERT INTO inventory (item\_name, quantity, price)

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

""", (item\_name, quantity, price))

conn.commit()

cur.close()

conn.close()

def read\_items():

""" Read all items from the inventory """

conn = connect\_db()

cur = conn.cursor()

cur.execute("SELECT \* FROM inventory;")

rows = cur.fetchall()

cur.close()

conn.close()

return rows

def update\_item(item\_id, quantity=None, price=None):

""" Update an existing item in the inventory """

conn = connect\_db()

cur = conn.cursor()

if quantity is not None:

cur.execute("""

UPDATE inventory

SET quantity = %s

WHERE id = %s;

""", (quantity, item\_id))

if price is not None:

cur.execute("""

UPDATE inventory

SET price = %s

WHERE id = %s;

""", (price, item\_id))

conn.commit()

cur.close()

conn.close()

def delete\_item(item\_id):

""" Delete an item from the inventory """

conn = connect\_db()

cur = conn.cursor()

cur.execute("""

DELETE FROM inventory

WHERE id = %s;

""", (item\_id,))

conn.commit()

cur.close()

conn.close()

def generate\_report(start\_date, end\_date):

""" Generate a report of inventory within a date range """

conn = connect\_db()

cur = conn.cursor()

query = """

SELECT \* FROM inventory

WHERE date\_added BETWEEN %s AND %s;

"""

cur.execute(query, (start\_date, end\_date))

rows = cur.fetchall()

df = pd.DataFrame(rows, columns=['ID', 'Item Name', 'Quantity', 'Price', 'Date Added'])

cur.close()

conn.close()

return df

def daily\_report():

today = datetime.now().date()

start\_date = datetime.combine(today, datetime.min.time())

end\_date = start\_date + timedelta(days=1)

return generate\_report(start\_date, end\_date)

def weekly\_report():

today = datetime.now().date()

start\_date = today - timedelta(days=today.weekday())

end\_date = start\_date + timedelta(days=7)

return generate\_report(start\_date, end\_date)

def monthly\_report():

today = datetime.now().date()

start\_date = today.replace(day=1)

next\_month = today.replace(day=28) + timedelta(days=4)

end\_date = next\_month - timedelta(days=next\_month.day)

return generate\_report(start\_date, end\_date)

def yearly\_report():

today = datetime.now().date()

start\_date = today.replace(month=1, day=1)

end\_date = today.replace(month=12, day=31)

return generate\_report(start\_date, end\_date)

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

create\_tables()

# Example usage:

add\_item('Pan', 100, 50.00)

print("Current Inventory:")

print(read\_items())

print("Daily Report:")

print(daily\_report())

print("Weekly Report:")

print(weekly\_report())

print("Monthly Report:")

print(monthly\_report())

print("Yearly Report:")

print(yearly\_report())

### Explanation

1. **Database Connection**: connect\_db() sets up the connection to the PostgreSQL database.
2. **Table Creation**: create\_tables() creates the inventory table if it doesn't exist.
3. **CRUD Operations**: Functions like add\_item(), read\_items(), update\_item(), and delete\_item() perform the respective operations.
4. **Reports**: Functions daily\_report(), weekly\_report(), monthly\_report(), and yearly\_report() generate reports based on date ranges.
5. **Example Usage**: Demonstrates adding an item and generating various reports.

Remember to replace the database parameters in DB\_PARAMS with your actual PostgreSQL configuration. Also, ensure that you have the pandas library installed for data handling.