**Day 10 Task Allocation :-**

**Asset Inventory Management System for pan shops using Python and PostgreSQL:**

**Prerequisites**

1. **Install psycopg2**: This is a PostgreSQL adapter for Python.

**Python Source Code ;-**

**pip install psycopg2**

1. **Set up PostgreSQL Database**: Ensure that PostgreSQL is installed and running on your machine. Create a database named asset\_inventory.

**Python Script Source Code:-**

**import psycopg2**

**from psycopg2 import sql**

**# Database connection**

**def connect\_db():**

**try:**

**conn = psycopg2.connect(**

**dbname="asset\_inventory",**

**user="your\_username",**

**password="your\_password",**

**host="localhost"**

**)**

**return conn**

**except Exception as e:**

**print(f"Error connecting to the database: {e}")**

**return None**

**# Create tables**

**def create\_tables(conn):**

**commands = (**

**"""**

**CREATE TABLE IF NOT EXISTS assets (**

**asset\_id SERIAL PRIMARY KEY,**

**asset\_name VARCHAR(255) NOT NULL,**

**asset\_type VARCHAR(255) NOT NULL,**

**quantity INTEGER NOT NULL,**

**purchase\_date DATE,**

**supplier VARCHAR(255)**

**)**

**""",**

**"""**

**CREATE TABLE IF NOT EXISTS shops (**

**shop\_id SERIAL PRIMARY KEY,**

**shop\_name VARCHAR(255) NOT NULL,**

**location VARCHAR(255) NOT NULL**

**)**

**""",**

**"""**

**CREATE TABLE IF NOT EXISTS shop\_assets (**

**id SERIAL PRIMARY KEY,**

**shop\_id INTEGER REFERENCES shops(shop\_id),**

**asset\_id INTEGER REFERENCES assets(asset\_id),**

**quantity INTEGER NOT NULL**

**)**

**"""**

**)**

**try:**

**with conn.cursor() as cur:**

**for command in commands:**

**cur.execute(command)**

**conn.commit()**

**print("Tables created successfully.")**

**except Exception as e:**

**print(f"Error creating tables: {e}")**

**# CRUD Operations**

**# Add new asset**

**def add\_asset(conn, asset\_name, asset\_type, quantity, purchase\_date, supplier):**

**try:**

**with conn.cursor() as cur:**

**cur.execute(**

**"""**

**INSERT INTO assets (asset\_name, asset\_type, quantity, purchase\_date, supplier)**

**VALUES (%s, %s, %s, %s, %s) RETURNING asset\_id;**

**""",**

**(asset\_name, asset\_type, quantity, purchase\_date, supplier)**

**)**

**conn.commit()**

**print(f"Asset '{asset\_name}' added successfully.")**

**except Exception as e:**

**print(f"Error adding asset: {e}")**

**# Get asset by ID**

**def get\_asset(conn, asset\_id):**

**try:**

**with conn.cursor() as cur:**

**cur.execute("SELECT \* FROM assets WHERE asset\_id = %s;", (asset\_id,))**

**asset = cur.fetchone()**

**if asset:**

**print("Asset details:", asset)**

**else:**

**print("Asset not found.")**

**except Exception as e:**

**print(f"Error retrieving asset: {e}")**

**# Update asset**

**def update\_asset(conn, asset\_id, asset\_name=None, asset\_type=None, quantity=None, purchase\_date=None, supplier=None):**

**try:**

**with conn.cursor() as cur:**

**query = sql.SQL("UPDATE assets SET {} WHERE asset\_id = %s;").format(**

**sql.SQL(', ').join([**

**sql.SQL("{} = %s").format(sql.Identifier(k)) for k, v in {**

**'asset\_name': asset\_name,**

**'asset\_type': asset\_type,**

**'quantity': quantity,**

**'purchase\_date': purchase\_date,**

**'supplier': supplier**

**}.items() if v is not None**

**])**

**)**

**cur.execute(query, [v for v in [asset\_name, asset\_type, quantity, purchase\_date, supplier] if v is not None] + [asset\_id])**

**conn.commit()**

**print(f"Asset ID '{asset\_id}' updated successfully.")**

**except Exception as e:**

**print(f"Error updating asset: {e}")**

**# Delete asset**

**def delete\_asset(conn, asset\_id):**

**try:**

**with conn.cursor() as cur:**

**cur.execute("DELETE FROM assets WHERE asset\_id = %s;", (asset\_id,))**

**conn.commit()**

**print(f"Asset ID '{asset\_id}' deleted successfully.")**

**except Exception as e:**

**print(f"Error deleting asset: {e}")**

**# Generate Reports**

**def generate\_report(conn):**

**try:**

**with conn.cursor() as cur:**

**cur.execute("""**

**SELECT a.asset\_name, a.asset\_type, a.quantity, s.shop\_name, sa.quantity**

**FROM assets a**

**JOIN shop\_assets sa ON a.asset\_id = sa.asset\_id**

**JOIN shops s ON sa.shop\_id = s.shop\_id;**

**""")**

**report = cur.fetchall()**

**if report:**

**print("Inventory Report:")**

**for row in report:**

**print(f"Asset: {row[0]}, Type: {row[1]}, Total Quantity: {row[2]}, Shop: {row[3]}, Quantity in Shop: {row[4]}")**

**else:**

**print("No data available for the report.")**

**except Exception as e:**

**print(f"Error generating report: {e}")**

**# Main function to demonstrate functionality**

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

**conn = connect\_db()**

**if conn:**

**create\_tables(conn)**

**# Example CRUD operations**

**add\_asset(conn, "Pan", "Consumable", 100, "2024-08-26", "Supplier A")**

**get\_asset(conn, 1)**

**update\_asset(conn, 1, quantity=120)**

**delete\_asset(conn, 1)**

**# Generate report**

**generate\_report(conn)**

**conn.close()**

**Explanation:**

1. **Database Connection**: The connect\_db() function establishes a connection to the PostgreSQL database.
2. **Creating Tables**: The create\_tables() function creates the necessary tables (assets, shops, shop\_assets).
3. **CRUD Operations**: Functions to add, get, update, and delete assets in the assets table.
4. **Generate Report**: The generate\_report() function generates a report that shows the asset inventory in different shops.

**Usage:**

* Ensure your PostgreSQL database is set up with the necessary permissions.
* Modify the user, password, and dbname parameters in the connect\_db() function as per your database configuration.
* Run the script to create the tables, perform CRUD operations, and generate reports.

This implementation is basic and can be expanded with more features, such as user authentication, a web interface, or additional reporting capabilities, depending on the requirements.