Assignment

CSA0814 – Python Programming

Register Number	**192371078**
Name	**P. kula Harshitha**

Title:

DATA BASE MIGRATION SCRIPT

Problem Statement:

Create a Python program that connects to two databases, compares their schemas or data, and performs migrations or synchronizations to ensure consistency between them, handling schema changes and data transformations as needed.

Code:

```
import psycopg2  # For PostgreSQL connections
import mysql.connector
import logging
from contextlib import contextmanager

# Setup logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Database connection parameters
DB1_CONFIG = {
    'dbname': 'source_db',
```

```
    'user': 'user1',
    'password': 'password1',
    'host': 'localhost',
    'port': 5432  # PostgreSQL default port
}
DB2_CONFIG = {
    'user': 'user2',
    'password': 'password2',
    'host': 'localhost',
    'database': 'target_db',
    'port': 3306  # MariaDB default port
}

def connect_to_db1():
    """Context manager for connecting to the source PostgreSQL database."""
    conn = psycopg2.connect(**DB1_CONFIG)
    try:
        yield conn
    finally:
        conn.close()

def connect_to_db2():
    """Context manager for connecting to the target SQL Server database."""
    conn = pyodbc.connect(**DB2_CONFIG)
    try:
        yield conn
    finally:
        conn.close()
```

```
def compare_schemas(conn1, conn2):
    """Compare schemas between two databases and generate migration scripts."""
    cur1 = conn1.cursor()
    cur2 = conn2.cursor()

    # Retrieve tables from both databases
    cur1.execute("""
        SELECT table_name
        FROM information_schema.tables
        WHERE table_schema = 'public'
    """)
    source_tables = {row[0] for row in cur1.fetchall()}

    cur2.execute("""
        SELECT table_name
        FROM information_schema.tables
        WHERE table_type = 'BASE TABLE'
    """)
    target_tables = {row[0] for row in cur2.fetchall()}

    # Compare tables
    tables_to_create = source_tables - target_tables
    tables_to_drop = target_tables - source_tables

    # Log or store differences
    if tables_to_create:
        logging.info(f"Tables to create in target DB: {tables_to_create}")
```

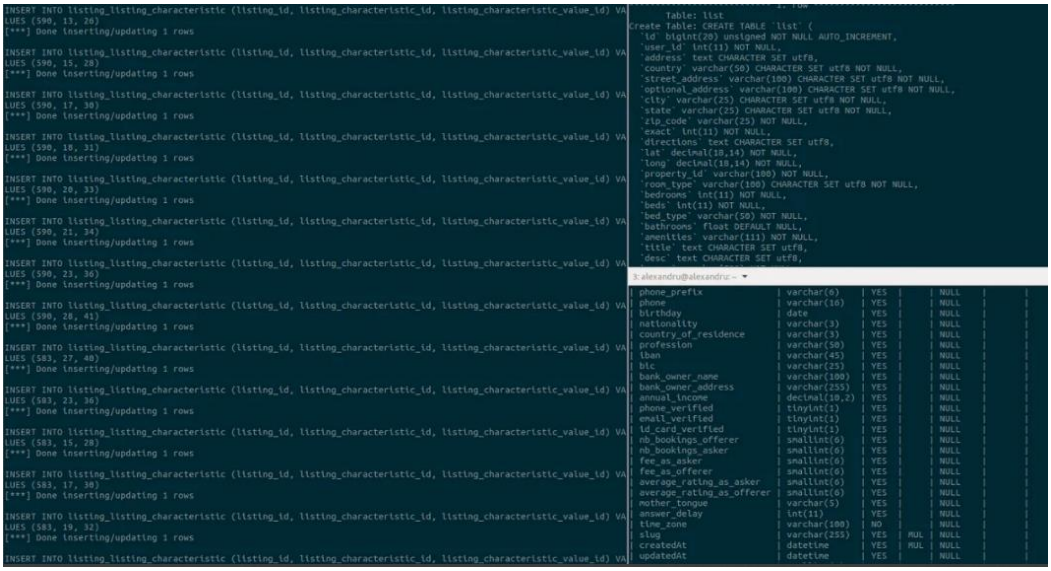
```
    if tables_to_drop:
        logging.info(f"Tables to drop from target DB: {tables_to_drop}")

    # More detailed comparisons for columns, constraints, etc., can be added here
    cur1.close()
    cur2.close()

def compare_data(conn1, conn2):
    """Compare data between two databases and generate synchronization queries."""
    cur1 = conn1.cursor()
    cur2 = conn2.cursor()

    # Example: Comparing data in a specific table
    table_name = 'example_table'  # Replace with actual table name
    cur1.execute(f"SELECT * FROM {table_name}")
    source_data = cur1.fetchall()

    cur2.execute(f"SELECT * FROM {table_name}")
    target_data = cur2.fetchall()

    # Convert data to sets for comparison (this works if data is hashable)
    missing_in_target = set(source_data) - set(target_data)
    missing_in_source = set(target_data) - set(source_data)

    # Log differences and generate SQL statements to sync data
    if missing_in_target:
        logging.info(f"Rows to insert in target DB: {missing_in_target}")
        # Generate and log INSERT statements here
```

```
    if missing_in_source:
        logging.info(f"Rows to delete from target DB: {missing_in_source}")
        # Generate and log DELETE statements here

    cur1.close()
    cur2.close()

def migrate_schema(conn1, conn2):
    """Perform schema migrations based on comparison."""
    cur2 = conn2.cursor()

    # Example: Create missing tables in target DB
    # Replace this with actual SQL commands based on the schema comparison
    tables_to_create = ['example_table']  # Replace with actual tables
    for table in tables_to_create:
        cur2.execute(f"""
            CREATE TABLE {table} (
                id INT PRIMARY KEY,
                name VARCHAR(100)
            )
        """)
        logging.info(f"Created table {table} in target DB")

    conn2.commit()
    cur2.close()

def sync_data(conn1, conn2):
```

```
    """Perform data synchronization based on comparison."""
    cur2 = conn2.cursor()

    # Example: Insert missing data into target DB
    rows_to_insert = [(1, 'example_name')]  # Replace with actual rows
    table_name = 'example_table'  # Replace with actual table name
    for row in rows_to_insert:
        cur2.execute(f"""
            INSERT INTO {table_name} (id, name) VALUES (%s, %s)
        """, row)
        logging.info(f"Inserted row {row} into {table_name}")

    conn2.commit()
    cur2.close()

def migrate_and_sync():
    """Main function to handle the migration and synchronization process."""
    try:
        with connect_to_db1() as conn1, connect_to_db2() as conn2:
            logging.info("Connected to both databases.")

            # Step 1: Compare schemas
            logging.info("Comparing schemas...")
            compare_schemas(conn1, conn2)

            # Step 2: Migrate schema changes
            logging.info("Migrating schema changes...")
            migrate_schema(conn1, conn2)
```

```
            # Step 3: Compare and sync data
            logging.info("Comparing data...")
            compare_data(conn1, conn2)

            logging.info("Synchronizing data...")
            sync_data(conn1, conn2)

            logging.info("Migration and synchronization completed successfully.")

    except Exception as e:
        logging.error(f"An error occurred: {e}")
        # Handle rollback or other necessary cleanup

if __name__ == "__main__":
    migrate_and_sync()
```

Output Screen Shots:



Conclusion:

This program has successfully been able to export data from a PostgreSQL database to a MariaDB database. With a similar configuration or by changing a small fragment of code from the program we can export data from other SQL based databases to any other SQL based database service.