Project Report: Automated Feed Generation, Duplicate Handling, and Comparison in PostgreSQL

# 1. Project Objective

The goal of this project was to design and implement an automated data processing system in PostgreSQL that can:  
1. Generate test feeds (tables) dynamically with configurable rows and columns.  
2. Identify and store duplicate records from feeds.  
3. Remove duplicates and keep unique records.  
4. Verify that duplicates are removed successfully.  
5. Compare data between different feeds and store mismatches or missing records.  
6. Run automated tests to validate the above steps.  
  
This setup simulates real-world BFSI (Banking, Financial Services, and Insurance) use cases where multiple data feeds must be validated, cleaned, and compared before being used in reporting or regulatory submissions.

# 2. Dataset Details

- Data is generated automatically using random strings from MD5(RANDOM()).  
- Each feed (Feed1, Feed2, etc.) can have variable numbers of rows and columns.  
- Example: Feed1 had 10 columns × 10 rows, while Feed2 had 15 columns × 15 rows.  
- An extra duplicate row is inserted deliberately for testing duplicate handling.

# 3. Work Done (Step by Step)

## Step 1 – Generate Feeds

A stored procedure `generate\_feed` was created. It dynamically builds tables with random rows and columns.  
Example: `CALL generate\_feed('Feed1', 10, 10);` created Feed1 with 10 columns and 10 rows.

## Step 2 – Identify and Store Duplicates

A table `duplicates` was created. The procedure `find\_and\_store\_duplicates` finds duplicate rows and stores them in JSON format.  
Example: Feed1 had 1 duplicate row group.

## Step 3 – Remove Duplicates

A procedure `replace\_duplicates\_with\_unique` removes duplicate rows while keeping distinct ones.  
Example: Feed1 reduced from 11 rows → 10 rows.

## Step 4 – Verify No Duplicates

A procedure `verify\_no\_duplicates` checks the feed after cleanup.  
Result: Feed1 had 0 duplicate groups remaining.

## Step 5 – Compare Feeds

A table `comparison\_results` was created. The procedure `compare\_feeds` compares two feeds row by row.  
Differences are logged in JSON with statuses: 'In source only' or 'In target only'.  
Example: Comparing Feed2 with Feed1 generated multiple mismatches.

## Step 6 – Automated Testing

The procedure `run\_automated\_tests` runs end-to-end testing:  
- Feed generation  
- Duplicate detection  
- Duplicate removal  
- Verification  
- Feed comparison  
  
Example Test Cases:  
TC-01: Feed1 generated → PASSED  
TC-04: Duplicate found in Feed1 → PASSED  
TC-05: Duplicates removed → PASSED  
TC-07: Feed2 compared with Feed1 → PASSED

# 4. Key Learnings

- How to use dynamic SQL in PostgreSQL.  
- Difference between MySQL and PostgreSQL (e.g., AUTO\_INCREMENT vs SERIAL, no DELIMITER in Postgres).  
- How to generate dummy data with random functions.  
- How to use JSONB to store duplicate and mismatch details.  
- Importance of automated validation in BFSI consulting for ensuring correctness of data feeds.

# 5. Project Outcome

At the end of this project, I successfully built an end-to-end automated framework in PostgreSQL which can:  
- Generate synthetic feeds  
- Detect and remove duplicates  
- Compare feeds  
- Run automated tests  
  
This framework is reusable and can be extended for ETL testing, reconciliation, and BFSI data quality checks.