## Cross-DB ETL Sync System on Azure VMs

**1. Project Objective**

To design and implement a **scalable**, **automated**, and **database-agnostic ETL-based solution** for syncing structured data across three Azure-hosted Linux VMs running PostgreSQL or other database engines in future. This setup ensures partial/full data synchronization from a primary database server (DB1) to two downstream servers (DB2 and DB3) using dynamic ETL pipelines.

**2. Scope of Work**

**Server Architecture:**

| **Server Name** | **Role** | **Function** |
| --- | --- | --- |
| DB1-test-server | Source Database Server | Stores multiple databases. Acts as the primary data source. |
| DB2-test-server | Sync Server (7-day view) | Stores filtered last 7 days of data from DB1. Updated daily. |
| DB3-test-server | Backup & Trimmed Sync Server | Restores full DB1 data initially, trims to last 7 days, then syncs daily. |

### ****DB1-test-server**** (Source Server)

* Host 4 or more databases (PostgreSQL for now, but could be any in future).
* Each database has tables:
  + Some tables have **30 days of date-wise data**.
  + Some tables have **non-date random data**.
* Acts as the **master/source database server**.

### ****DB2-test-server**** (Sync Server)

* Maintain a **7-day copy of all date-based tables** from DB1.
* For tables **without date columns**, copy full data.
* Daily sync using a **universal ETL script** (cron job).

### ****DB3-test-server**** (Backup & Sync Server)

* Initially restore full backup of DB1.
* Trim all tables to keep **only 7-day data**.
* Set up same **universal sync** as DB2 for daily updates from DB1.

## 3. ****Key Requirements****

* **Universal support** for PostgreSQL, MySQL, SQL Server, etc.
* Discover and process all available **databases and tables** on source server.
* Identify if a table has a **date/timestamp** column to support **date-based filtering**.
* Perform **daily sync** of:
  + **Last 7 days of data** for date-based tables.
  + **Full data** for tables without a date column.
* Automatically **truncate and load** target tables (overwrite mode).
* Centralized configuration using YAML.
* Script should be **modular, portable, and secure**.

## 4. ****Solution Design****

### Components:

| **Component** | **Description** |
| --- | --- |
| etl\_script.py | Main ETL pipeline: reads config, loops through DBs/tables, filters & syncs. |
| db\_utils.py | Handles DB connections, table structure detection, column checks, etc. |
| config.yaml | Stores DB connection info; easily switch between PostgreSQL, MySQL, etc. |
| cron | Runs the ETL job daily on DB2 and DB3 servers. |

### Directory Structure:

universal\_etl/

* config.yaml
* etl\_script.py
* db\_utils.py
* requirements.txt

**5. Technology Stack**

* **Python 3.x**
* **SQLAlchemy** – universal database connector
* **Pandas** – fast data manipulation and loading
* **ODBC/Native drivers** – PostgreSQL, MySQL, SQL Server
* **Cron** – job scheduler for daily sync / Azure DevOps pipelines
* (Optional) Azure CLI or SSH scripts for remote management

**6. Workflow Overview**

**Daily ETL Flow (DB1 → DB2/DB3):**

1. Discover all non-template databases on DB1.
2. For each database:
   * List all tables.
   * Check if a **date/timestamp** column exists.
     + If yes → fetch data from last **7 days**.
     + If no → fetch **entire table**.
3. Truncate corresponding table on target (DB2 or DB3).
4. Insert the new dataset into the target table.
5. Log status for each table/database.

**7. Benefits**

* **Database Agnostic**: Works with PostgreSQL, MySQL, SQL Server, etc.
* **Dynamic**: Auto-detects databases and tables.
* **Efficient**: Only recent data is synced (if applicable).
* **Modular and Configurable**: Easily portable across environments.
* **Future-Proof**: No reliance on native DB replication features.

**8. Future Enhancements**

* Add **logging and email alerts**.
* Integrate **Azure Blob/S3 backup** functionality.
* Implement **CDC (Change Data Capture)** for real-time sync.
* Parallel processing of large tables.
* Dashboard for ETL monitoring.

**9. Deliverables**

* Fully functional universal\_etl/ folder with working scripts.
* Initial setup instructions and test cases.
* Cron job configuration on DB2 and DB3.
* Documentation for modifying or extending the solution.