**Project Scope – Data Feed Processing & Validation Automation**

**Executive Summary**

This project focuses on the design, automation, and validation of data feeds using SQL. The scope included generating random input data across three feeds (Feed-1, Feed-2, and Feed-3), detecting and removing duplicates, and comparing datasets to ensure consistency and integrity. The effort was complemented by a structured testing framework (manual and automated) and full documentation of the process.

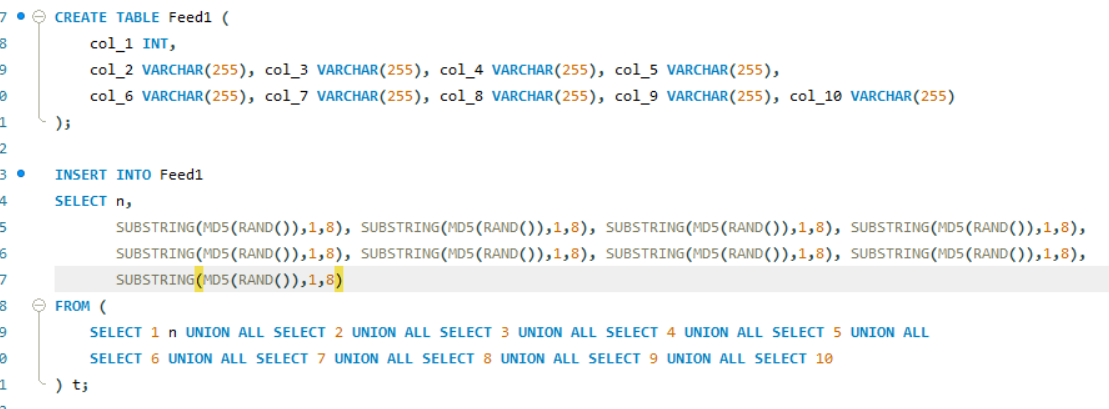
The solution was executed in multiple stages:

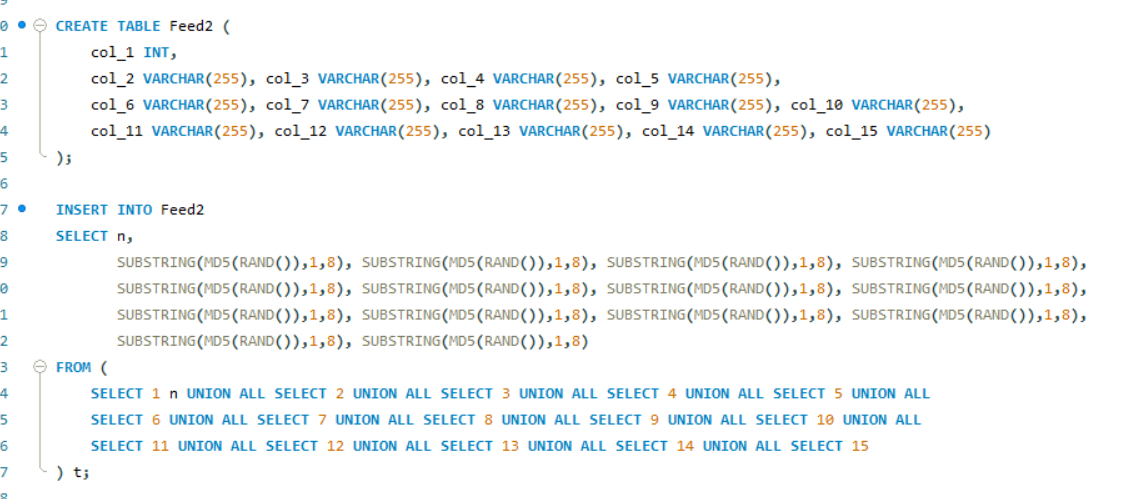
1. Feed Generation: Automated creation of parameterized feeds with configurable row counts and column structures.
2. Duplicate Management: Identification of duplicate records, extraction into a separate output file for audit purposes, and cleansing to retain only unique rows.
3. Data Comparison: Direct comparison of Feed-2 and Feed-3 against Feed-1 to identify mismatches and gaps.
4. Validation: Comprehensive SQL queries ensured zero residual duplicates and correct feed alignment.
5. Testing: A detailed test plan was prepared with end-to-end scenarios, ranging from functional validation to data integrity checks. Automation was introduced to execute tests and improve repeatability.
6. Documentation: The entire process, including objectives, SQL scripts, test results, and automation, was compiled into a professional project report with screenshots for traceability.

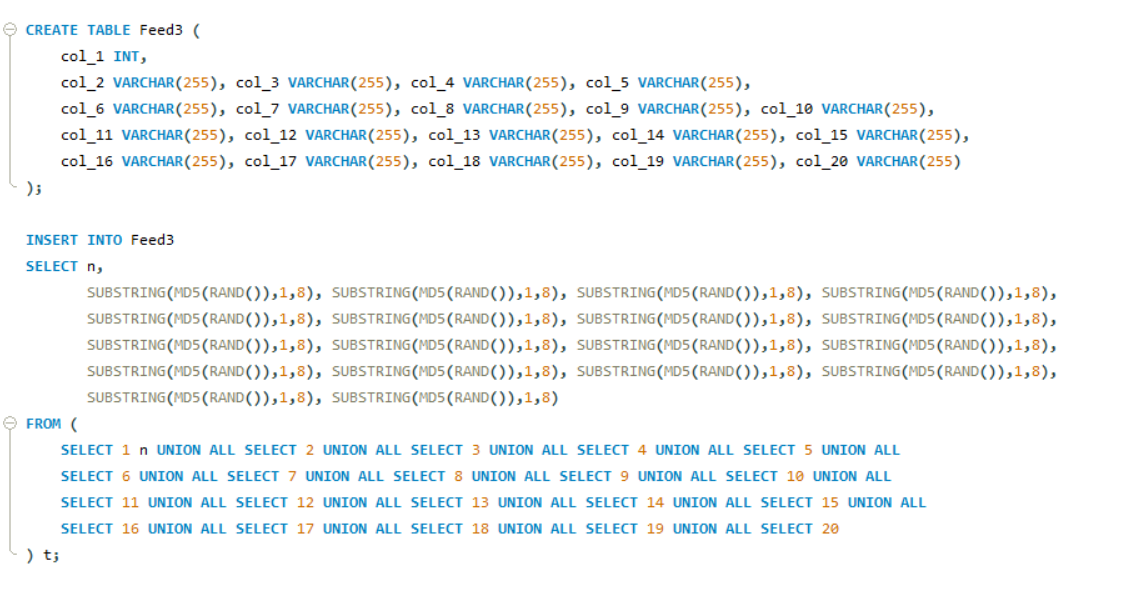
Through this approach, the project delivered a robust, repeatable, and auditable process for managing data feeds, ensuring clean, consistent, and comparable datasets. This framework not only demonstrates technical accuracy but also establishes a reusable model for future data integration and validation initiatives.

**Req 1: Generate Random Data Input Files**

**Objective:** To create baseline test datasets (Feed-1, Feed-2, Feed-3) with defined rows and columns. These controlled inputs ensure consistency in downstream validation and provide a foundation for repeatable testing.

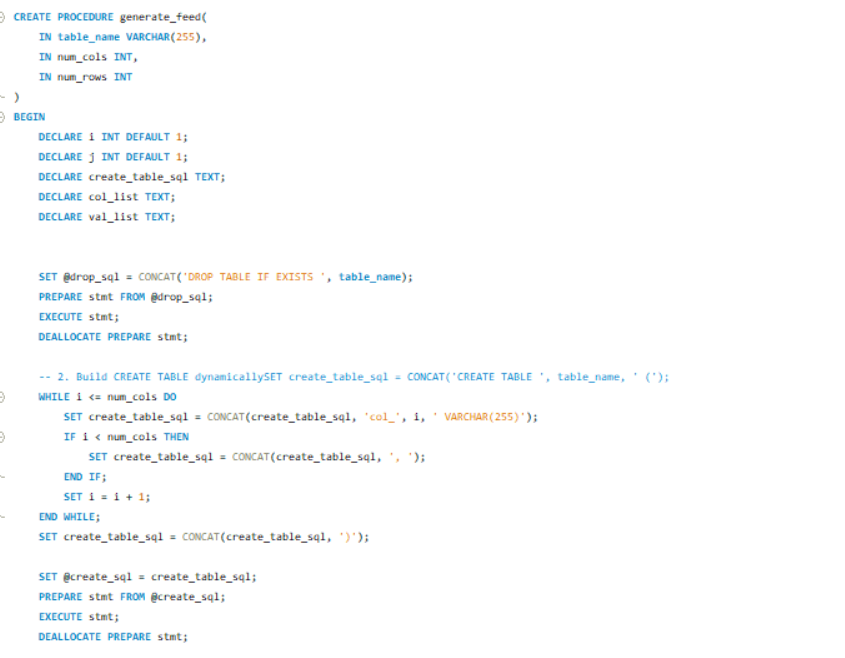


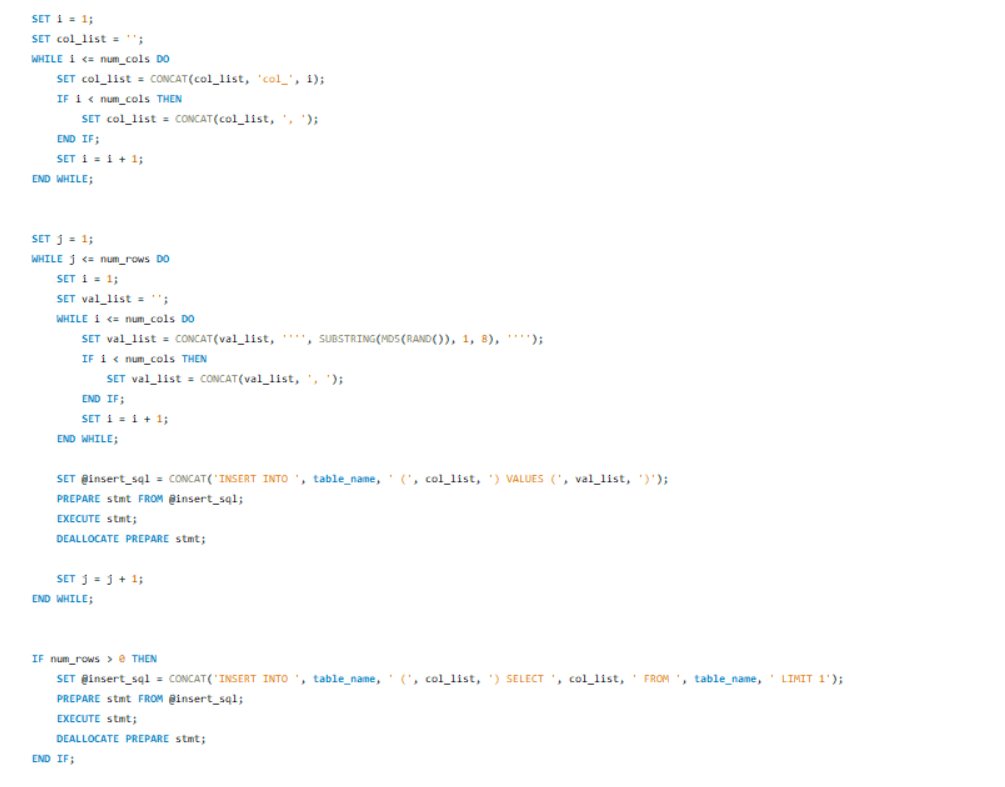




**Req 2: Automate Feed Generation using SQL**

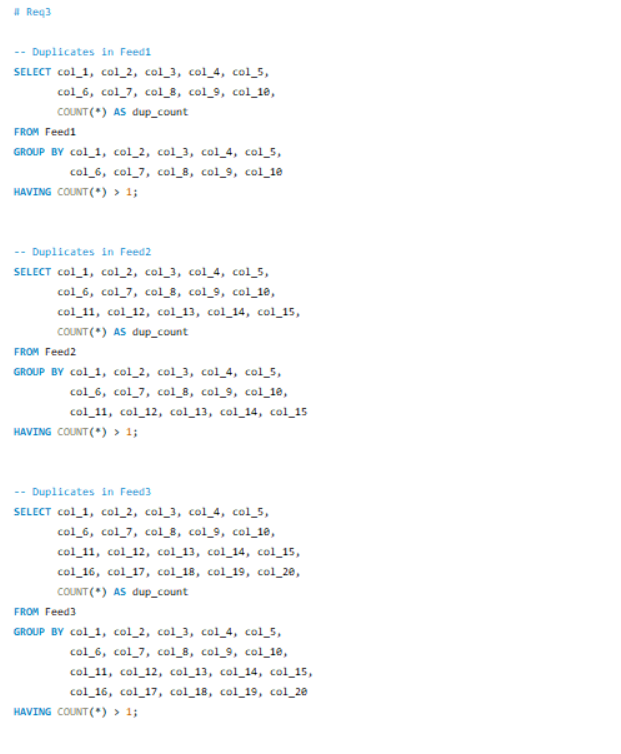
Objective: To develop a parameter-driven SQL procedure that dynamically generates feeds based on user input (feed name and number of rows). This eliminates manual effort, enhances reusability, and ensures scalability for multiple feed configurations.





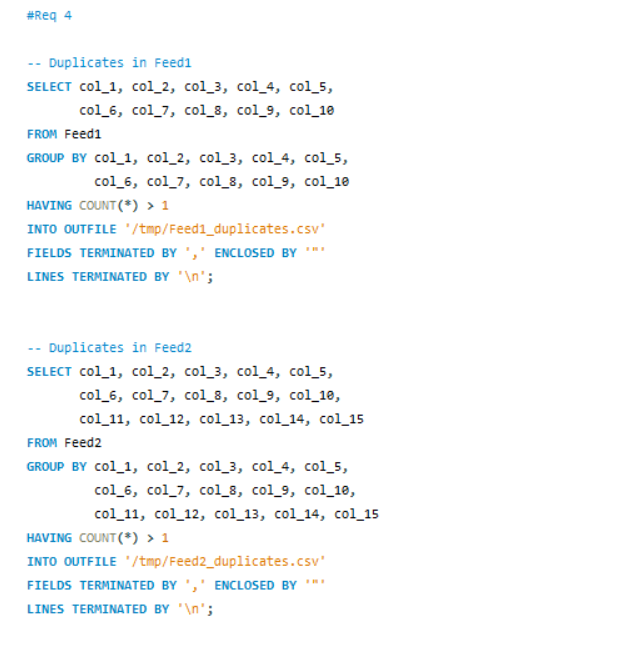
**Req 3: Identify Duplicate Records**

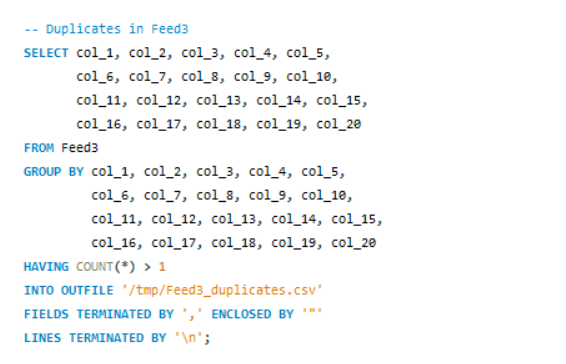
Objective: To validate data quality by detecting duplicate rows within each feed. Early identification of duplicates ensures accuracy in reporting and prevents propagation of erroneous data into business processes.



**Req 4: Write Duplicate Records to Output File**

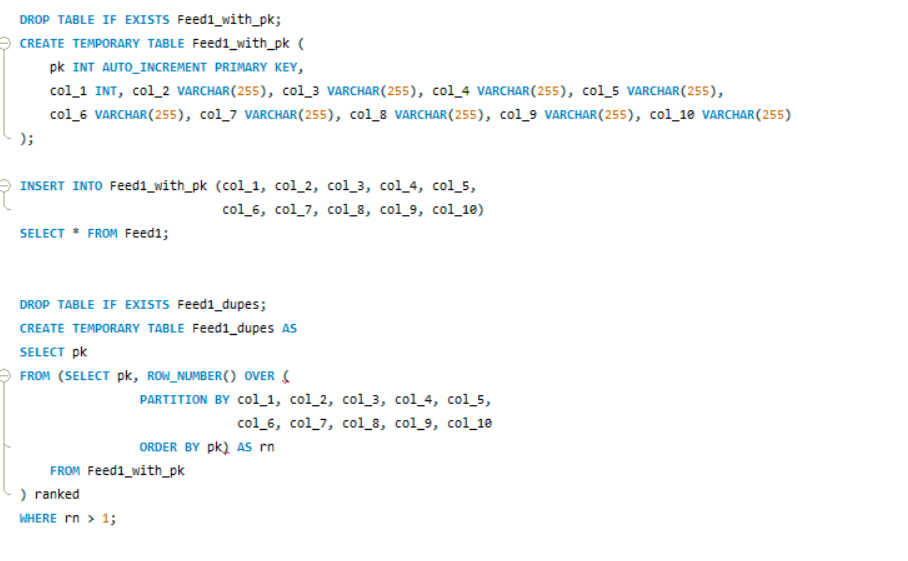
**Objective:** To provide traceability by capturing identified duplicates into a separate “Duplicates” output file. This creates an audit trail for validation teams and supports transparent error reporting.

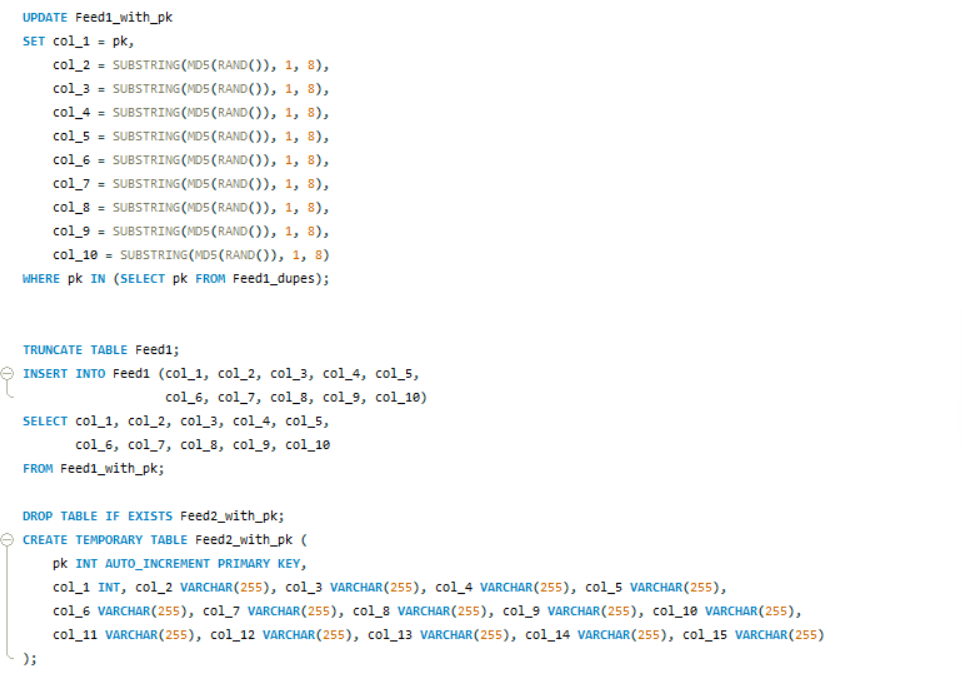


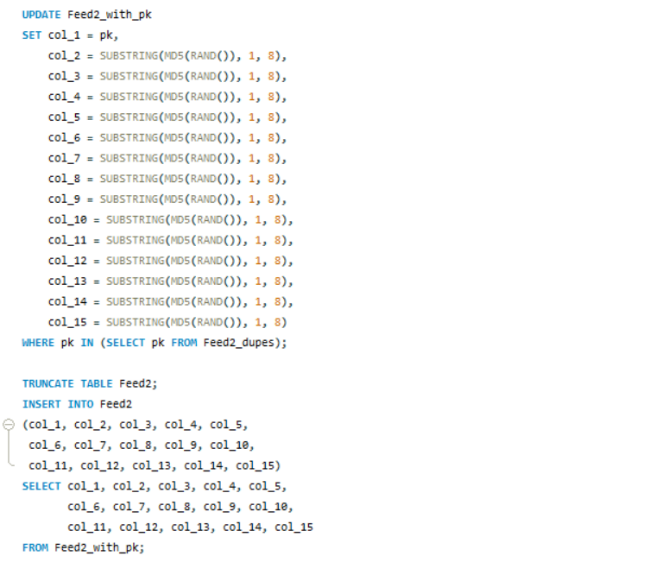
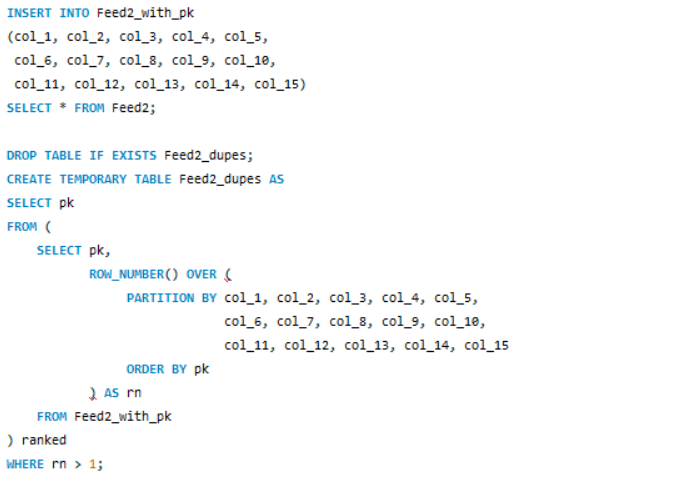


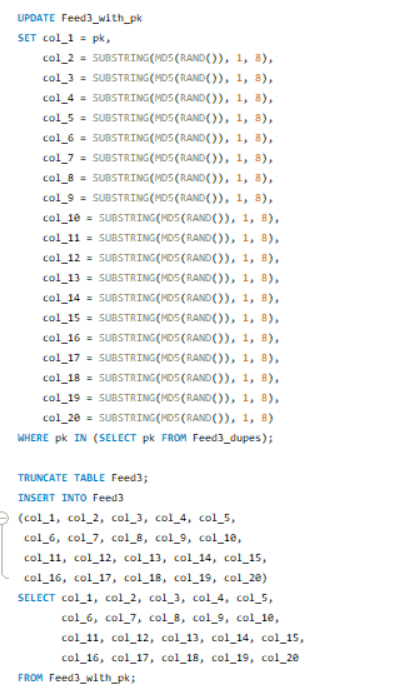
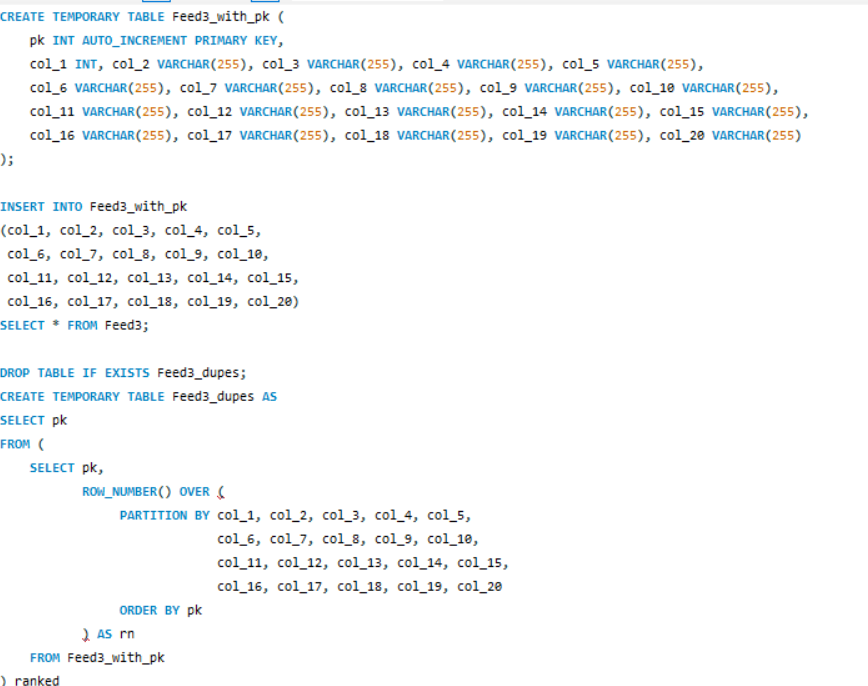
**Req 5: Replace Duplicates with Unique Rows**

**Objective:** To cleanse each feed by eliminating duplicates and retaining only unique rows. This guarantees data integrity and ensures the feed is reliable for further use or comparison.



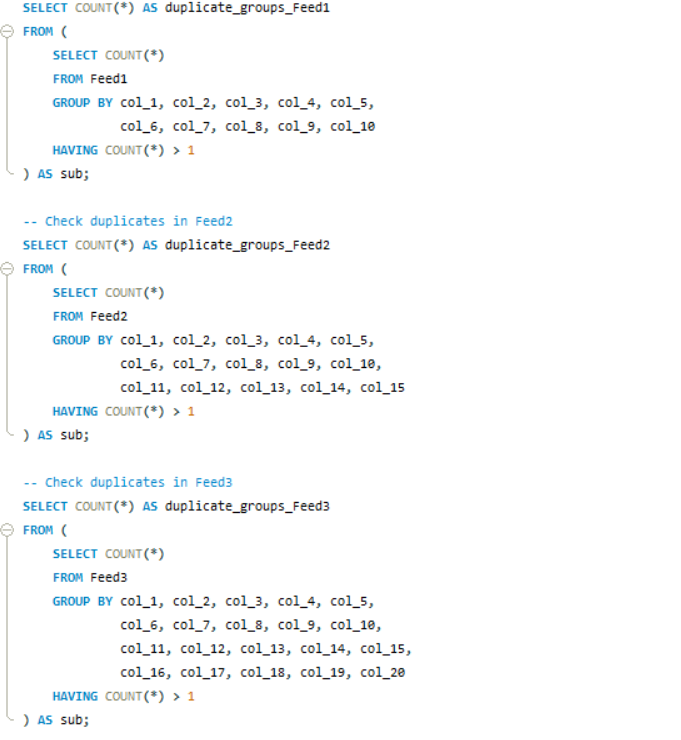


****

****

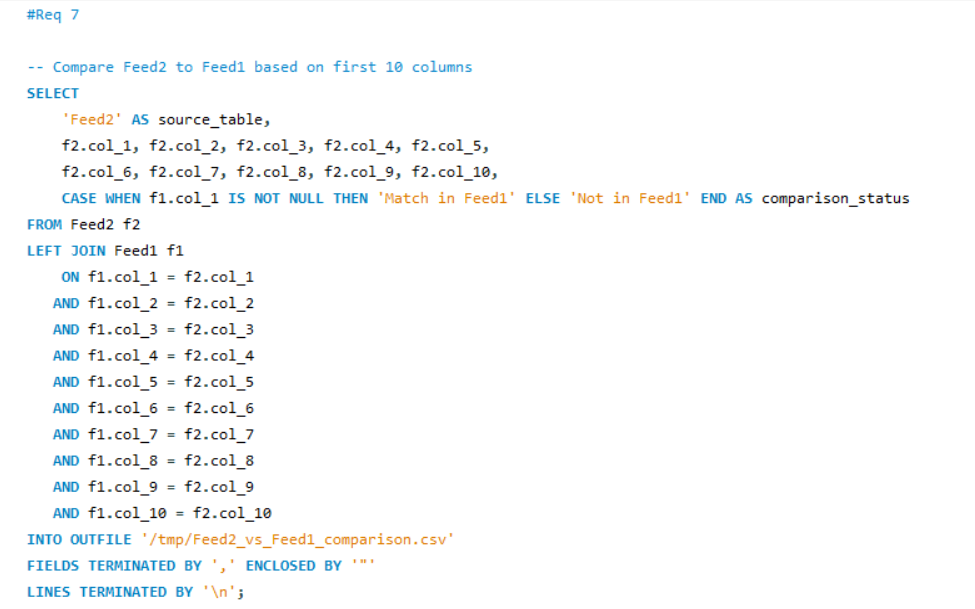
**Req 6: Execute Script and Verify Zero Duplicates**

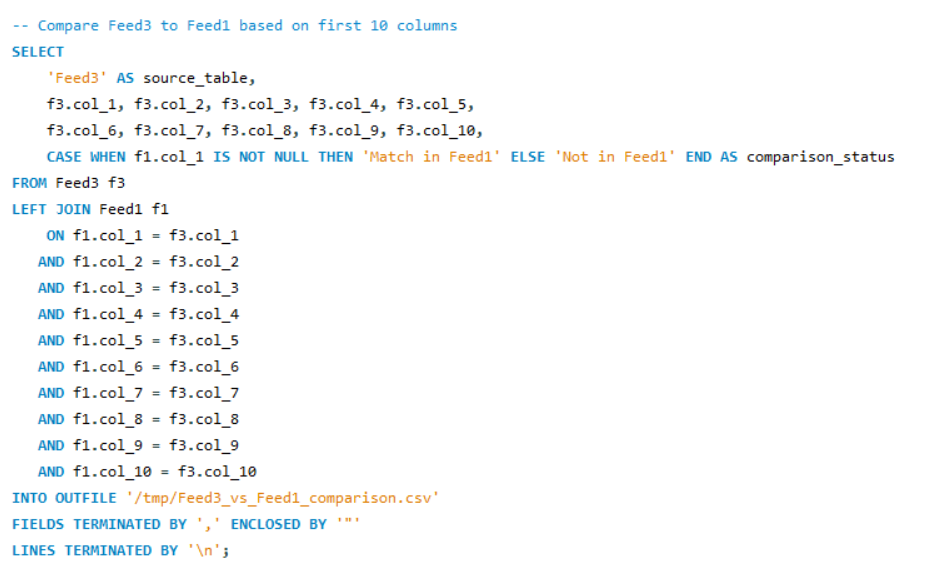
**Objective:** To validate the effectiveness of the deduplication process. The script should confirm that no duplicate records remain, proving the cleansing step has succeeded.

****

**Req 7: Compare Feed-2 and Feed-3 with Feed-1**

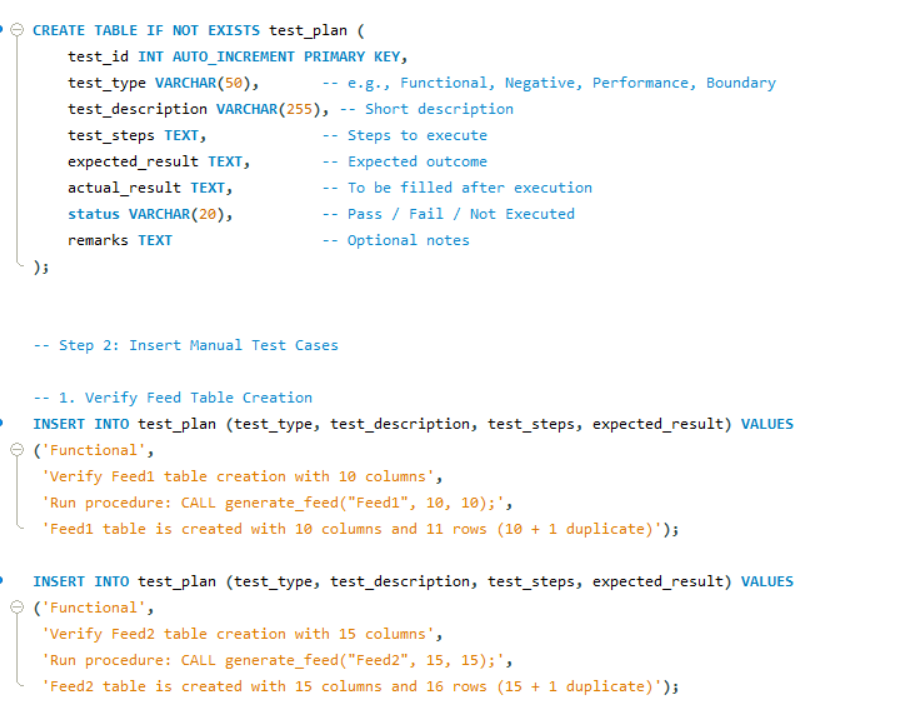
**Objective:** To assess data consistency and alignment across feeds by identifying mismatches and variances. This comparison provides insights into feed integrity and highlights data gaps.

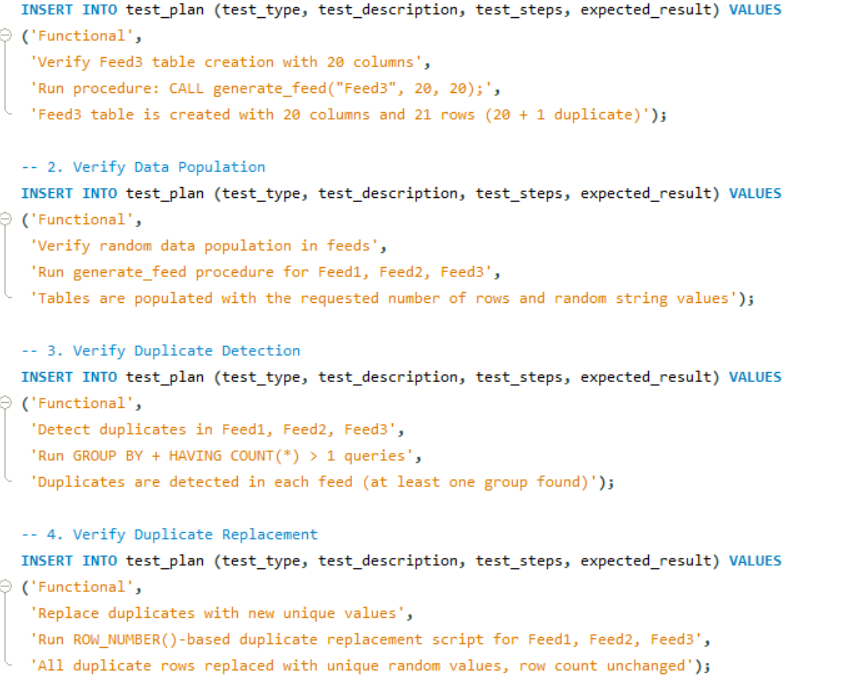


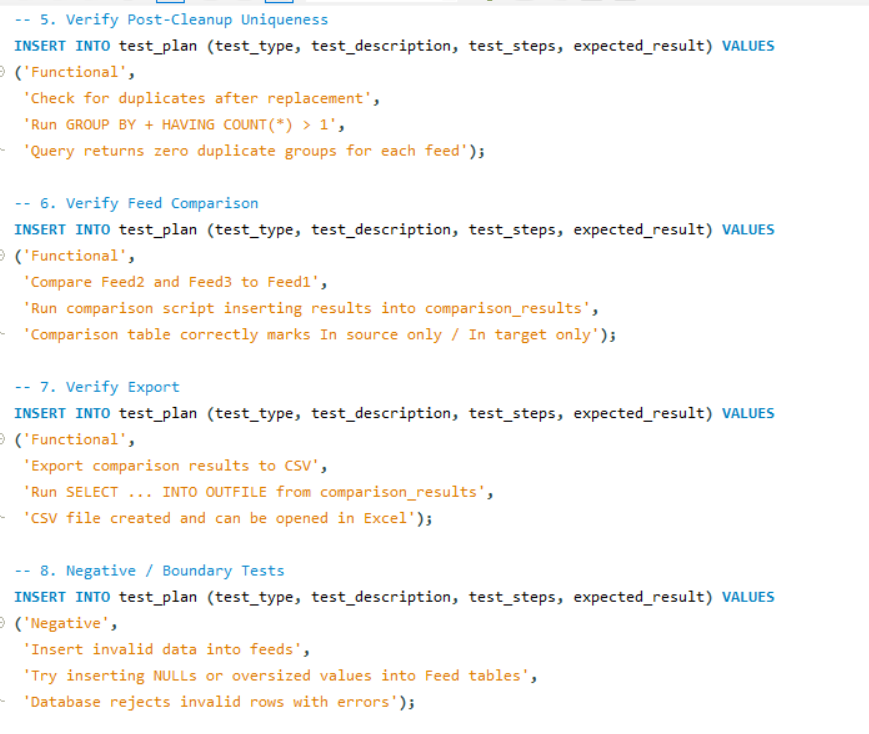


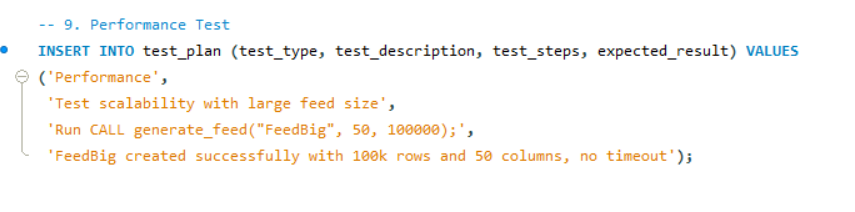
**Req 8: Create Test Plan (Manual)**

**Objective:** To ensure comprehensive functional validation by preparing structured test cases. The plan defines objectives, test steps, and expected results to systematically validate end-to-end functionality.



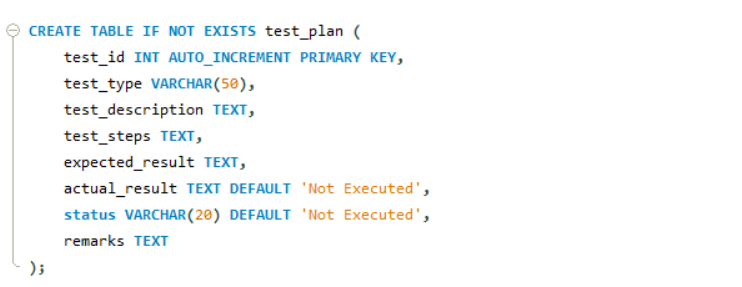




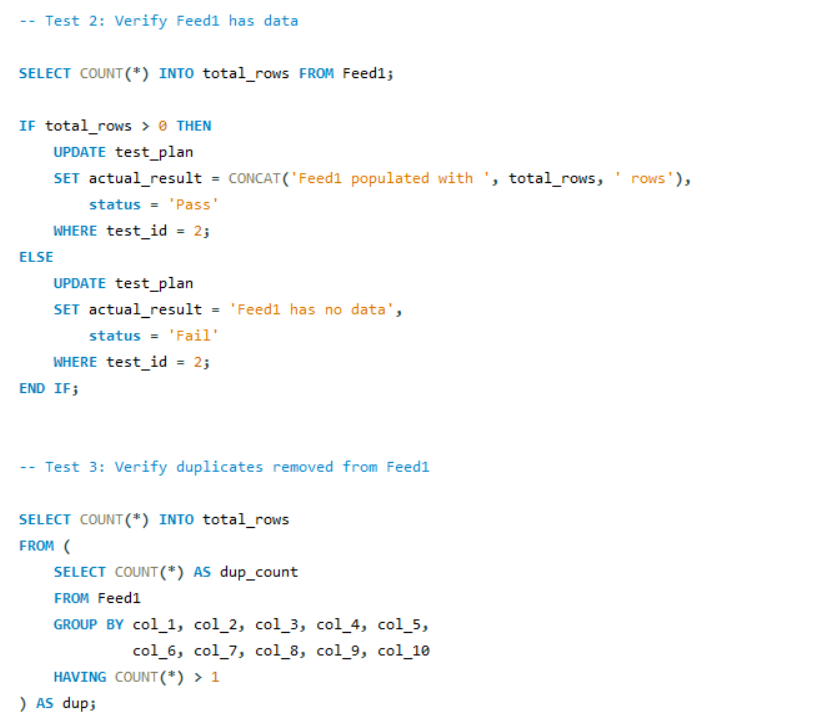


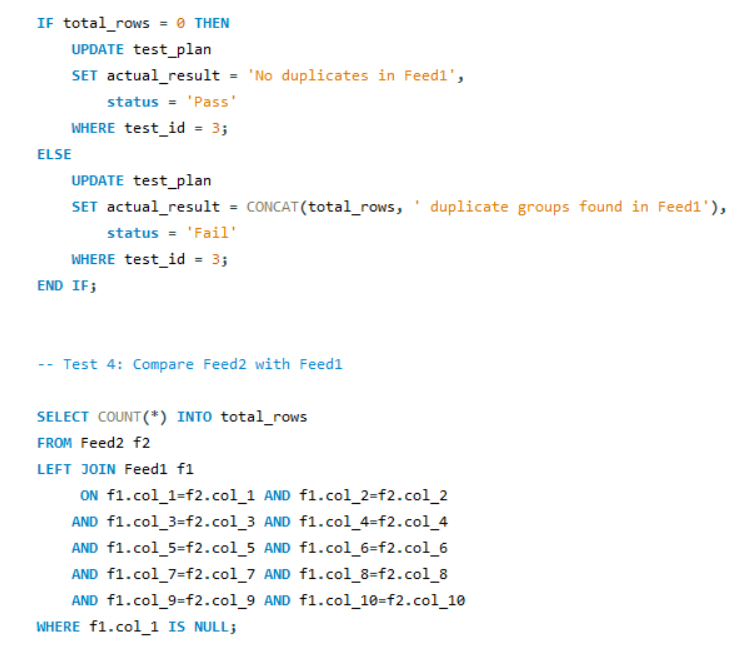
**Req 9: Automate Test Cases**

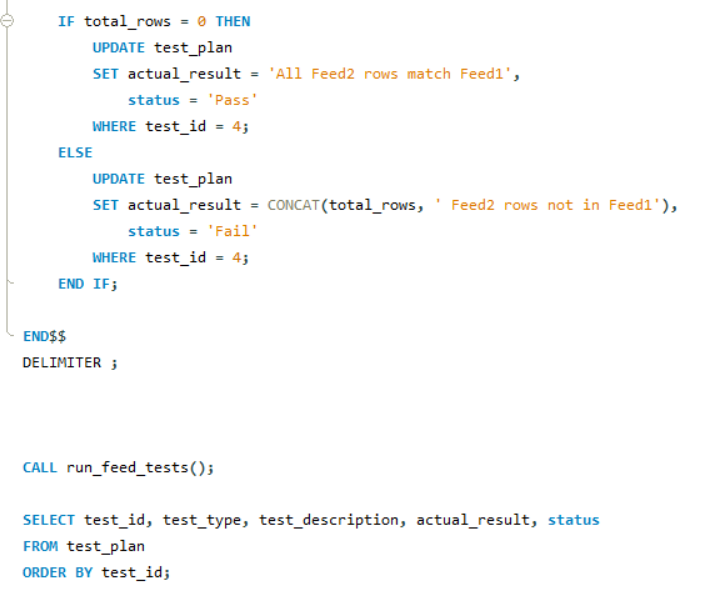
**Objective:** To increase efficiency and reliability in testing by implementing automation (via SQL testing frameworks or external scripting). Automated testing reduces manual effort, ensures repeatability, and supports continuous validation.











**Documentation (Req10)**

All aspects of the project were documented to ensure traceability and transparency. SQL scripts for feed generation, duplicate handling, feed comparison, and test automation were compiled as part of the deliverables. CSV outputs of duplicates and comparison results were validated and included as evidence. A professional project report was prepared, structured around requirements, implementations, and validation outcomes. Screenshots of SQL query outputs, CSV exports, and test results were incorporated to further substantiate the findings. This level of documentation ensures that the project can be easily reviewed, reproduced, and extended in the future.

**Validation Evidence**

Validation was performed at each stage to confirm the accuracy of results. Duplicate detection queries successfully identified intentional duplicate records. After cleansing, verification queries confirmed that no duplicate groups remained in any of the feeds. Exported CSV files, such as duplicates and comparison results, were opened in Excel for independent review. Automated test outputs further validated that the solution functioned as intended, with clear Pass/Fail results logged in the test\_plan table.

**Risks and Mitigation**

Several risks were identified during the project. File permission issues may arise when using SELECT INTO OUTFILE; this can be mitigated by ensuring proper database privileges and configurable export paths. Large datasets may affect runtime performance, which can be addressed with indexing strategies. Environment dependencies, particularly across different versions of MySQL, can be mitigated by testing scripts on supported platforms (this solution was validated on MySQL 8). Additionally, to avoid skewed data distributions, randomness in data generation was carefully managed using hashing functions.

**Conclusion and Next Steps**

This project successfully delivered a complete, end-to-end SQL framework for managing data feeds. It automated feed generation, duplicate detection, cleansing, and comparison, while embedding both manual and automated testing mechanisms. The framework ensures high data quality, reliability, and scalability, and establishes a strong foundation for future data integration and validation initiatives.

Looking ahead, the framework can be integrated into ETL pipelines to support enterprise-scale data workflows. It can also be scheduled as part of nightly or CI/CD jobs to ensure continuous validation. Future improvements may include extending test automation using Python and frameworks like pytest to provide richer reporting and integration with dashboarding tools. With these enhancements, the solution can evolve into a fully automated, enterprise-grade data quality assurance platform.