Students’ Data Import Script Documentation

HP | Software Development Unit

TERABYTE

2025

**Students’ data Import & Department Sync Script Documentation**

**OVERVIEW**

This Python script provides a robust, fully automated pipeline for importing student course registration records from the students’ registration data **usually received in Excel format** into the **uirms database**. It ensures referential integrity, comprehensive error handling, department change tracking, and produces detailed logs of all operations and issues.

**LIST OF TABLES AFFECTED BY THE SCRIPT**

Below are the main MySQL tables that the script interacts with, either by inserting, updating, or referencing data during the import and synchronization process:

|  |  |  |
| --- | --- | --- |
| Table Name | Operations | Purpose |
| tbl\_faculty | Read | Used to map faculty names in the Excel file to their unique IDs required by other tables. |
| tbl\_departments | Read | Used to map department names to their IDs and details. |
| tbl\_courses\_test | Read/Insert | Holds all course records. The script checks for the existence of each course from the Excel file and inserts new ones if missing. |
| students\_courses | Create/Insert | A staging table for all imported student-course records from Excel. Each row represents a student’s registration for a course. |
| tbl\_students\_master\_test | Read/Insert/Update | The master student record. New students are inserted if not present; department changes are updated and logged. |
| tbl\_students\_transactions\_test | Read/Insert | Records each student’s registration/transaction for a session and level. Ensures no duplicate transactions. |
| student\_users\_test | Read/Insert | Stores student user accounts for portal access. New accounts are created if missing. |
| tbl\_course\_registered\_test | Read/Insert/Update | Stores individual course registration records for students, with full referencing. |
| course\_dept\_reg\_map\_test | Read/Insert/Update | Aggregates and maps course registrations by course, department, session, and semester. |

**PROCESS FLOW**

**BLOCK-BY-BLOCK EXPLANATION**

|  |  |  |  |
| --- | --- | --- | --- |
| Block | What It Does | Purpose | Key Operations |
| 1. Imports & Configuration | Loads required libraries and configures paths | Ensures all dependencies are available | Import statements, path configurations |
| 2. User Prompt for Session ID | Prompts user to enter academic session ID | Links records to correct session | `input()` for session\_id\_fk |
| 3. Excel Data Loading | Reads Excel file, cleans data | Prepares data for database mapping | `pd.read\_excel()`, data cleaning |
| 4. Database Connection | Establishes MySQL connection | Maintains consistent database access | `mysql.connector.connect()` |
| 5. Reference Data Loading | Loads faculty/dept/course references | Enables automated data mapping | SELECT queries to load reference tables |
| 6. Insert Missing Courses | Adds new courses not in database | Ensures complete course catalog | `INSERT INTO tbl\_courses\_test` |
| 7. Create students\_courses Table | Creates staging table if missing | Prevents import failures | `CREATE TABLE IF NOT EXISTS` |
| 8. Populate & Sync Student Data | Inserts records, handles dept changes | Maintains student master data | `INSERT/UPDATE tbl\_students\_master\_test` |
| 9. Export Dept Changes | Saves department change logs | Provides audit trail | CSV export of change records |
| 10. Populate Transactions | Creates registration transactions | Tracks student enrollment | `INSERT INTO tbl\_students\_transactions\_test` |
| 11. Create User Accounts | Generates student login accounts | Enables portal access | `INSERT INTO student\_users\_test` |
| 12. Export Skipped Transactions | Logs failed transactions | Supports troubleshooting | Error logging to csv form |
| 13. Course Registration | Records student-course links | Core registration system | `INSERT INTO tbl\_course\_registered\_test` |
| 14. Update course\_dept Registration Map | Aggregates course registrations | Supports reporting/analytics | `INSERT INTO course\_dept\_reg\_map\_test` |
| 15. Compile Logs | Combines all logs into Excel | Centralized audit trail | Multi-sheet Excel export |

**KEY FEATURES:**

- Each block represents a distinct processing stage

- Clear separation of concerns

- Progressive data validation

- Comprehensive logging at each stage

- Maintains referential integrity throughout

**VISUAL FLOW:**

Excel → Data Cleaning → DB Connection → Reference Setup → Student Processing → Course Registration → Logging → Final Output

**HOW THIS SCRIPT ADDRESSES MANUAL DATA ENTRY ISSUES**

|  |  |
| --- | --- |
| Manual Issue | Script Solution |
| Typos and mapping errors | Automated lookups, normalization, and logging of unmapped items. |
| Missed/duplicate records | Duplicate checks before every insert, progress bars for completeness. |
| No tracking of department changes | Logs every department update, including previous and new values. |
| Missing course data | Inserts any missing courses automatically and logs them. |
| Manual user account creation | Bulk account creation with fallback passwords and issue logging. |
| No audit trail | Skipped/changed/new records all logged in an Excel file. |
| Slowness and tedium | Processes thousands of records quickly, with progress feedback. |

**USAGE INSTRUCTIONS**

1. Prepare your Excel file as `studentdata.xlsx` in the script directory.

2. Edit DB connection details at the top of the script if needed.

3. Run the script in your Python environment.

4. Enter the session ID when prompted.

5. Monitor the console for progress bars and warning messages.

6. Review the `import\_logs.xlsx` file for any errors, skips, or department updates.

**REQUIREMENTS FOR EFFICIENT SCRIPT EXECUTION**

* Correct UTME registration number for the password to the student result portal
* Students and their correct programmes
* Comprehensive list of transfer students to properly update their mode\_id in the tbl\_students\_master\_test table
* Students with distinct registration data
* Correction of errors in the "Observations from the registration data for last session" file
* Update of courses associated with Veterinary Medicine department (dept\_id of 68) in the tbl\_courses\_test table (they have units of 0), which is also an issue with some other courses in this table too

**SUMMARY OF BASIC INSERT/UPDATE OPERATIONS**

**-** **Insert student-course record:** `INSERT INTO students\_courses (...) VALUES (...)`

**- Insert new student:** `INSERT INTO tbl\_students\_master\_test (...) VALUES (...)`

**- Update student's department:** `UPDATE tbl\_students\_master\_test SET previous\_dept = ..., department\_id\_fk = ... WHERE matricNo = ...`

**- Insert registration transaction:** `INSERT INTO tbl\_students\_transactions\_test (...) VALUES (...)`

**- Create user account:** `INSERT INTO student\_users\_test (...) VALUES (...)`

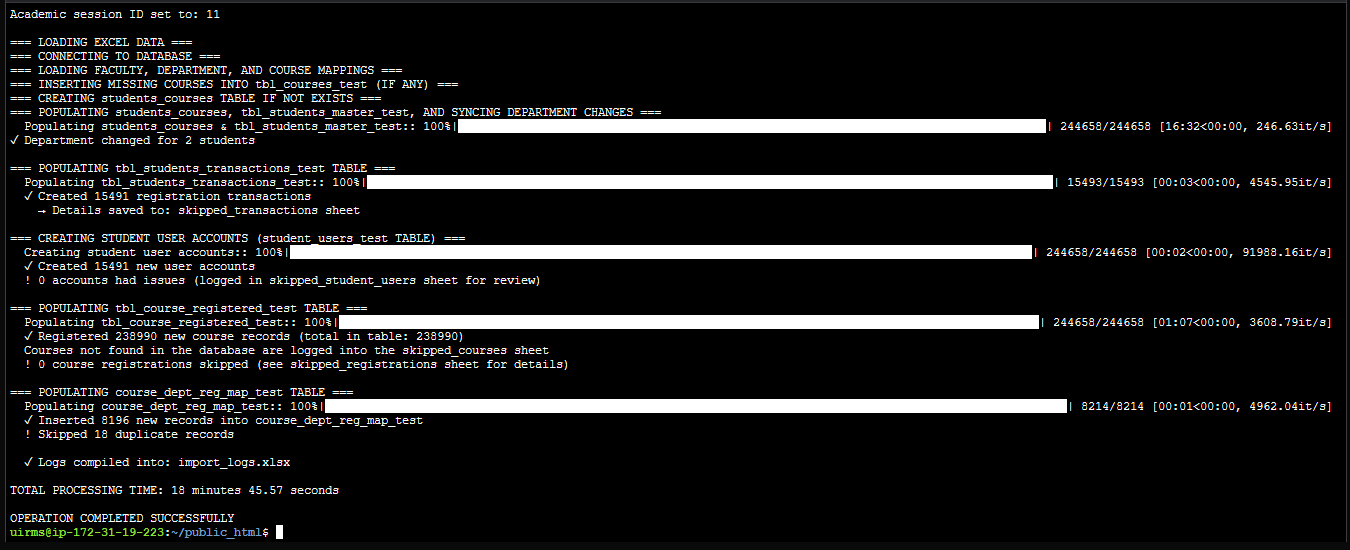
**- Register student for a course:** `INSERT INTO tbl\_course\_registered\_test (...) VALUES (...)`

**- Update department mapping:**  `INSERT INTO course\_dept\_reg\_map\_test (...) VALUES (...)`

**PROCESSING CAPACITY**

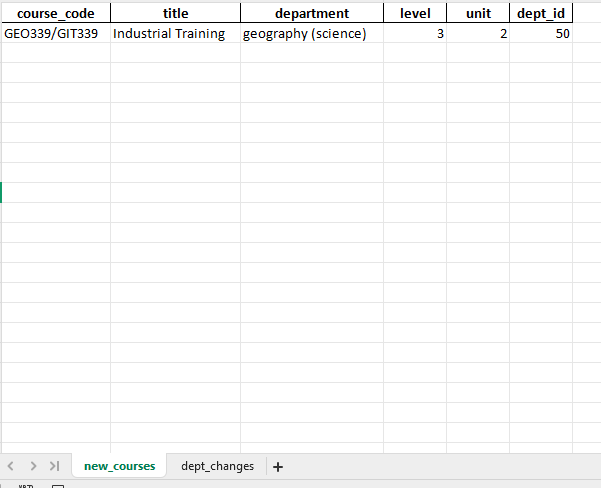
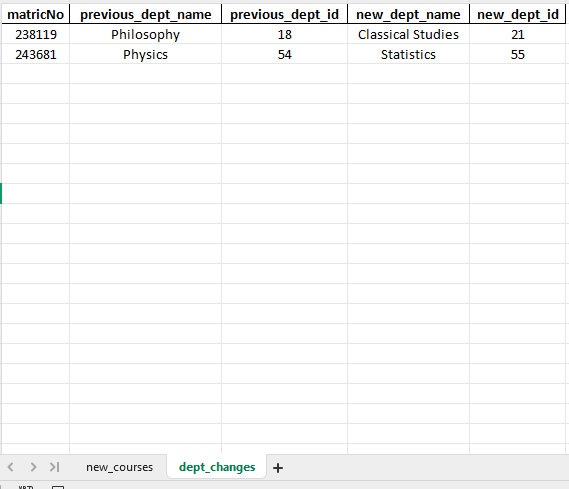
* **Typical Dataset Size**: 244,659 registration records from last session’s registration data for testing.
* **Average Processing Time**: 20 minutes
* **Records Processed/Minute**: ~9,800
* **Total Operations**: 15 sequential stages

**Screenshot of the operation using a test data**



**SCREENSHOTS OF THE IMPORT\_LOGS.XLSX FILE GENERATED:**

**New course code detected Students with change in departments**

**CONCLUSION**

**This script:**

- Eliminates manual, error-prone steps in student-course registration data import.

- Handles all reference lookups, inserts, and updates atomically and efficiently.

- Tracks every skipped, changed, or new record for transparency.

- Provides a one-stop Excel audit log for easy review and troubleshooting.

- Greatly reduces administrative workload and risk of data entry mistakes.