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.

**REQUIREMENTS TO USE THE SCRIPT:**

* **Python 3.6+**
* **Required packages:**
  + pandas: Used to read student data from CSV/Excel files and to manage tabular data efficiently before inserting into the database.
  + pymysql: Used to connect, insert, update, or retrieve student records in/from a MySQL database.
  + openpyxl: Used for reading, writing, modifying and programmatically formatting Excel files (.xlsx) in Python.
  + numpy: Used for numerical operations and array processing, often alongside pandas.
  + tqdm: Used to add a progress bar to for-loops when processing multiple students for better user feedback.
  + Pytz: Used to ensure timestamps are in the correct timezone (e.g., West Africa Time) when creating or updating student records.
  + datetime: Used to get current timestamps for record creation and modification tracking.
  + csv: Used to import/export student data in bulk
  + os: Used for file path manipulations, handling command-line arguments, or other system tasks.
  + re: Standard library: Regular expressions for string searching/matching.
* **MySQL Database Access:** The script must have valid credentials and access to a running MySQL server/database where student records will be inserted/updated. Ensure the required database tables and columns exist as expected by the script.
* **Student Data File:** If importing students in bulk, ensure you have the student data in a supported format (Excel file named **‘studentdata.xlsx’**).
* **Network/Permissions:** The machine running the script must have network access to the MySQL server and permission to read/write data files.

**REQUIREMENTS FOR EFFICIENT SCRIPT EXECUTION**

* Correction of errors in the "Observations from the registration data for last session" file
* Correct UTME registration number (for password to the result portal) and correct programmes to update the programme\_id\_fk field
* Comprehensive list of transfer students to properly update their mode\_id in the tbl\_students\_master\_test table
* 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.

**USAGE INSTRUCTIONS**

* Prepare your Excel file as `studentdata.xlsx` in the script directory.
* Run the script in your Python environment.
* Enter the session ID when prompted.
* Monitor the console for progress bars and warning messages.
* Review the `import\_logs.xlsx` file for any errors, skips, or department updates.

**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 (...)`

**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 & 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 result 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](#_Configuration_(File_Paths):) |
| 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](#_Data_Cleaning_1) | Reads Excel file, cleans data | Prepares data for database mapping | `pd.read\_excel()`, [data cleaning](#_Data_Cleaning) |
| 4. Database Connection | Establishes MySQL connection | Maintains consistent database access | `mysql.connector.connect()`, [data connection](#_Data_Connection) |
| 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`, [data processing](#_Data_Processing) |
| 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](#_Logging_Overview_and) |

**EXPLANATIONS OF SOME PARTS OF THE PROCESS FLOW:**

# Path Configuration (File Paths):

* **`excel\_file\_path`:** Path to the input Excel file ('studentdata.xlsx')
* **`sheet\_name`:** Worksheet name to read ('Sheet1')
* **`table\_name`:** Target table for student courses ('students\_courses')
* **`skipped\_output\_file`:** CSV file for skipped student records ('skipped\_students.csv')

# Data Cleaning

* Cleans column names (removes spaces, converts to lowercase)
* Handles null/NaN values

# Data Connection

**`mysql.connector.connect()`**

connection = pymysql.connect( host='xxxxxxx', user='xxxxx', password='xxxxxxxxxxx', database='xxxxx')

# Data Processing

* Processes each student record with progress bar visualization
* Extracts and normalizes the required data:
  + Student names breaking one Fullname column in the excel file into 3 fields in the database (surname, firstname, middlename), Level and mode (regular/DE), Gender, Semester information, Course status.

# Logging Overview and Output Sheets.

The script maintains comprehensive logs for every major operation, error, or data exception encountered during execution. All such logs are written to CSV files as the process runs, and at the end, these are compiled into a single Excel file (`import\_logs.xlsx`) with each log as a separate sheet for easy review by administrators. If a particular log type is not needed (e.g., no skipped students), the sheet may be omitted or a note inserted indicating no records were skipped. Below is a summary table:

|  |  |  |
| --- | --- | --- |
| Sheet Name | CSV Source File | Logged Content Description |
| new\_courses | new\_courses.csv | All new courses encountered in the Excel file and inserted into `tbl\_courses\_test`. |
| skipped\_students | skipped\_students.csv | Students not processed due to unmapped or missing faculty/department information. |
| skipped\_transactions | skipped\_transactions.csv | Student registration transactions that could not be created (duplicates, missing references, or DB errors). |
| skipped\_student\_users | skipped\_student\_users.csv | User accounts that could not be created (missing UTME, other credential issues). |
| skipped\_registrations | skipped\_course\_registered.csv | Course registration records not inserted due to missing data, duplicates, or other issues. |
| dept\_changes | dept\_changes.csv | All department changes for students (previous and new department IDs/names). |

**KEY FEATURES OF THE SCRIPT:**

* Each block represents a distinct processing stage
* Uses tqdm for progress visualization
* 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

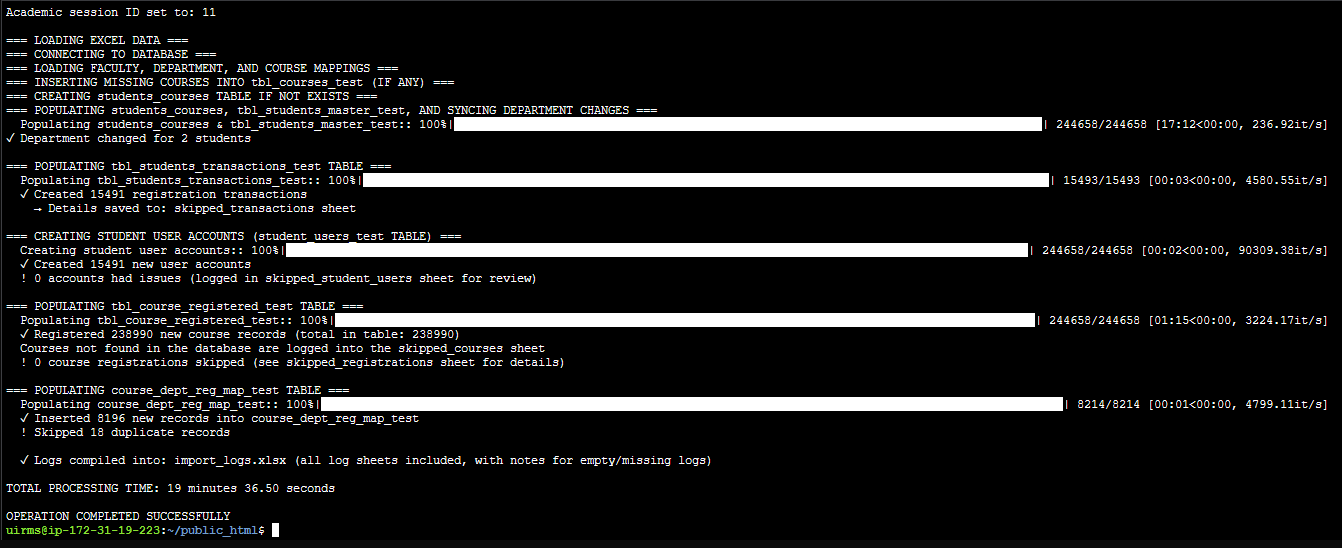
**HOW THIS SCRIPT ADDRESSES MANUAL DATA ENTRY ISSUES**

|  |  |
| --- | --- |
| 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. |

**PROCESSING CAPACITY**

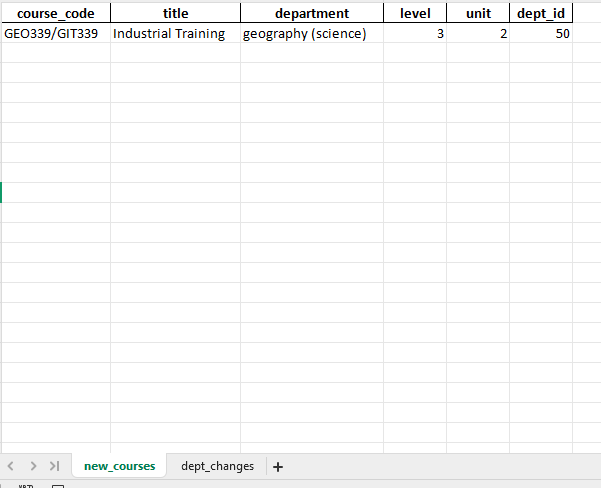
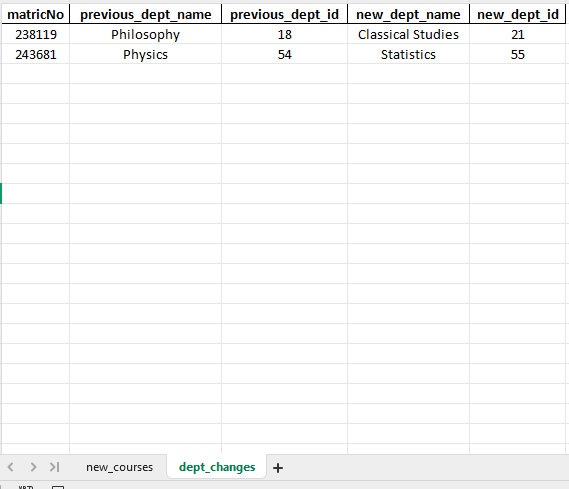
* **Typical Dataset Size**: 244,659 registration records from last session’s registration data for testing.
* **Average Processing Time**: 20 minutes
* **Records Processed/Minute**: ~9,800
* **Total Operations**: 15 sequential stages

**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 includes comprehensive error handling that:**

* Continues processing after non-critical errors.
* Provides clear feedback about any issues encountered.
* Skips invalid records while preserving the data.
* 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 (as seen for the negative course units in the registration data which were converted to positive course units during the operation since course units are always positive).