2025

Students’ Data Import Script Documentation

AFEEZ AJADI AYOBAMI

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, pymysql, openpyxl, tqdm

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

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 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()`, 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
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

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:

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.

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:

- 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

course_code	title	department	level	unit	dept_id
GEO339/GIT339	Industrial Training	geography (science)	3	2	50

new_courses dept_changes +

Students with change in departments

matricNo	previous_dept_name	previous_dept_id	new_dept_name	new_dept_id
238119	Philosophy	18	Classical Studies	21
243681	Physics	54	Statistics	55

new_courses dept_changes +

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.