

Students' Data Import Script Documentation

TERABYTE

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	<code>`input()`</code> for session_id_fk
3. Excel Data Loading	Reads Excel file, cleans data	Prepares data for database mapping	<code>`pd.read_excel()`</code> , data cleaning
4. Database Connection	Establishes MySQL connection	Maintains consistent database access	<code>`mysql.connector.connect()`</code>
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	<code>`INSERT INTO tbl_courses_test`</code>
7. Create students_courses Table	Creates staging table if missing	Prevents import failures	<code>`CREATE TABLE IF NOT EXISTS`</code>
8. Populate & Sync Student Data	Inserts records, handles dept changes	Maintains student master data	<code>`INSERT/UPDATE tbl_students_master_test`</code>
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	<code>`INSERT INTO tbl_students_transactions_test`</code>
11. Create User Accounts	Generates student login accounts	Enables portal access	<code>`INSERT INTO student_users_test`</code>
12. Export Skipped Transactions	Logs failed transactions	Supports troubleshooting	Error logging to csv form
13. Course Registration	Records student-course links	Core registration system	<code>`INSERT INTO tbl_course_registered_test`</code>
14. Update course_dept Registration Map	Aggregates course registrations	Supports reporting/analytics	<code>`INSERT INTO course_dept_reg_map_test`</code>
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

```
Academic session ID set to: 11

=== LOADING EXCEL DATA ===
=== CONNECTING TO DATABASE ===
=== LOADING FACULTY, DEPARTMENT, AND COURSE MAPPINGS ===
=== INSERTING MISSING COURSES INTO tbl_courses_test (IF ANY) ===
=== CREATING students_courses TABLE IF NOT EXISTS ===
=== POPULATING students_courses, tbl_students_master_test, AND SYNCING DEPARTMENT CHANGES ===
Populating students_courses & tbl_students_master_test:: 100%| 244658/244658 [16:32<00:00, 246.63it/s]
✓ Department changed for 2 students

=== POPULATING tbl_students_transactions_test TABLE ===
Populating tbl_students_transactions_test:: 100%| 15493/15493 [00:03<00:00, 4545.95it/s]
✓ Created 15491 registration transactions
→ Details saved to: skipped_transactions sheet

=== CREATING STUDENT USER ACCOUNTS (student_users_test TABLE) ===
Creating student user accounts:: 100%| 244658/244658 [00:02<00:00, 91988.16it/s]
✓ Created 15491 new user accounts
! 0 accounts had issues (logged in skipped_student_users sheet for review)

=== POPULATING tbl_course_registered_test TABLE ===
Populating tbl_course_registered_test:: 100%| 244658/244658 [01:07<00:00, 3608.79it/s]
✓ Registered 238990 new course records (total in table: 238990)
Courses not found in the database are logged into the skipped_courses sheet
! 0 course registrations skipped (see skipped_registrations sheet for details)

=== POPULATING course_dept_reg_map_test TABLE ===
Populating course_dept_reg_map_test:: 100%| 8214/8214 [00:01<00:00, 4962.04it/s]
✓ Inserted 8196 new records into course_dept_reg_map_test
! Skipped 18 duplicate records

✓ Logs compiled into: import_logs.xlsx

TOTAL PROCESSING TIME: 18 minutes 45.57 seconds

OPERATION COMPLETED SUCCESSFULLY
uirms@ip-172-31-19-223:~/public_html$
```

SCREENSHOTS OF THE IMPORT_LOGS.XLSX FILE GENERATED:

New course code detected

[illegible]

Students with change in departments

[illegible]

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.