ETL Automation : Excel to MySQL using Streamlit

# Project Title:

Excel to MySQL ETL Automation using Streamlit

# Objective:

To automate the Extract, Transform, and Load (ETL) process by uploading Excel files and loading their content into a MySQL database using an interactive Streamlit web interface.

# Technologies Used:

|  |  |
| --- | --- |
| Technology | Purpose |
| Streamlit | Frontend Web UI |
| Pandas | Reading and handling Excel data |
| SQLAlchemy | ORM for MySQL connectivity |
| PyMySQL | MySQL driver for SQLAlchemy |
| OpenPyXL | Backend Excel file reader (XLSX) |

# Python Code (app.py):

import streamlit as st  
import pandas as pd  
from sqlalchemy import create\_engine  
  
st.set\_page\_config(page\_title="Excel to MySQL ETL", layout="wide")  
  
st.title("📊 Excel to MySQL ETL Automation")  
  
# Step 1: Upload Excel file  
uploaded\_file = st.file\_uploader("Upload your Excel file", type=["xlsx"])  
  
if uploaded\_file:  
 # Step 2: Read Excel into DataFrame  
 try:  
 df = pd.read\_excel(uploaded\_file)  
 st.subheader("Preview of Excel Data")  
 st.dataframe(df)  
 except Exception as e:  
 st.error(f"Error reading Excel file: {e}")  
   
 # Step 3: Get MySQL details  
 st.subheader("Database Connection")  
 host = st.text\_input("Host", value="localhost")  
 user = st.text\_input("User", value="root")  
 password = st.text\_input("Password", type="password")  
 database = st.text\_input("Database")  
 table\_name = st.text\_input("Table Name", value="uploaded\_data")  
  
 if st.button("Upload to MySQL"):  
 try:  
 # Create connection  
 engine = create\_engine(f'mysql+pymysql://{user}:{password}@{host}/{database}')  
   
 # Load data into MySQL  
 df.to\_sql(name=table\_name, con=engine, if\_exists='replace', index=False)  
 st.success(f"✅ Data successfully uploaded to `{table\_name}` table in `{database}` database!")  
 except Exception as e:  
 st.error(f"❌ Failed to upload data: {e}")

# Required Python Libraries:

pip install streamlit pandas sqlalchemy pymysql openpyxl

# ETL Process Breakdown:

## 1. Extract

The user uploads an Excel file through Streamlit’s UI. The file is parsed using Pandas (pd.read\_excel()).

## 2. Transform

Optional: You can insert transformation logic here (e.g., column cleanup, type casting). The current version only previews the raw DataFrame.

## 3. Load

Streamlit captures MySQL credentials from user input. A connection is created via SQLAlchemy and PyMySQL. The DataFrame is pushed to MySQL with .to\_sql().

# Sample Use Cases:

|  |  |
| --- | --- |
| Use Case | Benefit |
| Business team uploads monthly reports | Automated push to reporting DB |
| School admin uploads student data | Easily save to central SQL system |
| Data analyst loads ad-hoc Excel data | Skip manual SQL scripting |

# Security Considerations:

• Credentials are entered by the user and not stored permanently.  
• Always secure the Streamlit app using VPN or Streamlit Cloud permissions for sensitive environments.

# Best Practices:

• Validate Excel Data before uploading (e.g., mandatory columns).

• Error Handling: Use try-except for every critical step.

• Transformation Layer: Add intermediate logic if needed.