```python
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics.pairwise import cosine_similarity

# Step 1: Load Data
file_path = "university_dataset_real_2000 final.xlsx"
xls = pd.ExcelFile(file_path)
df = xls.parse('university_dataset_real_2000')

# Step 2: Preprocessing
df = df.drop_duplicates()
df = df.dropna(subset=['University', 'Average_GPA_Requirement', 'IELTS_Requirement'])

num_cols = ['Tuition_Fee_USD', 'Application_Fee_USD', 'Living_Cost_Per_Year_USD',
            'Acceptance_Rate', 'Job_Placement_Rate']
df[num_cols] = df[num_cols].fillna(df[num_cols].mean())

cat_cols = ['Country', 'City', 'Field_Specialization', 'Scholarship_Available',
            'Public_or_Private', 'Language_of_Instruction', 'Intake_Months',
            'Mode_of_Study', 'Internship_Opportunities', 'Region', 'Degree_Level']

ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
ohe_encoded = pd.DataFrame(ohe.fit_transform(df[cat_cols]), columns=ohe.get_feature_names_out(cat_cols))
df_encoded = pd.concat([df.drop(columns=cat_cols).reset_index(drop=True), ohe_encoded], axis=1)

# Step 3: User Input
print("Enter your details to get university recommendations:")
raw_input = {
    'Field_Specialization': input("Field of Specialization (e.g., Artificial Intelligence): "),
    'Degree_Level': input("Degree Level (Bachelor/Master/PhD): "),
    'Average_GPA_Requirement': float(input("Your GPA (e.g., 3.5): ")),
    'IELTS_Requirement': float(input("IELTS Score (e.g., 7.0): ")),
    'Tuition_Fee_USD': float(input("Max Tuition Fee in USD (e.g., 15000): ")),
    'Living_Cost_Per_Year_USD': float(input("Max Living Cost in USD (e.g., 12000): ")),
    'Scholarship_Available': input("Scholarship Available (Yes/No): "),
    'Region': input("Preferred Region (e.g., EU, US, Asia): ").strip(),
    'Country': 'None',
    'City': 'None',
    'Public_or_Private': 'None',
    'Language_of_Instruction': 'None',
    'Intake_Months': 'None',
    'Mode_of_Study': 'None',
    'Internship_Opportunities': 'None'
}

# Apply Hard Constraints
df_filtered = df[
    (df['Tuition_Fee_USD'] <= raw_input['Tuition_Fee_USD']) &
    (df['Living_Cost_Per_Year_USD'] <= raw_input['Living_Cost_Per_Year_USD']) &
    (df['IELTS_Requirement'] <= raw_input['IELTS_Requirement']) &
    (df['Region'].str.lower() == raw_input['Region'].lower())
]

if df_filtered.empty:
    print("⚠️ No universities matched your strict filters. Showing top matches regardless of budget/region...")
    df_filtered = df.copy()

# Filter strictly by Degree_Level match if user explicitly requested one
df_filtered = df_filtered[df_filtered['Degree_Level'].str.lower() == raw_input['Degree_Level'].lower()]

# Re-encode after filtering
ohe_encoded_filtered = pd.DataFrame(ohe.transform(df_filtered[cat_cols]), columns=ohe.get_feature_names_out(cat_cols))
df_encoded_filtered = pd.concat([df_filtered.drop(columns=cat_cols).reset_index(drop=True), ohe_encoded_filtered], axis=1)

# Encode input
input_df = pd.DataFrame([raw_input])
ohe_input = pd.DataFrame(ohe.transform(input_df[cat_cols]), columns=ohe.get_feature_names_out(cat_cols))
input_encoded = pd.concat([input_df.drop(columns=cat_cols).reset_index(drop=True), ohe_input], axis=1)

# Step 4: Similarity Calculation
selected_features = input_encoded.columns.intersection(df_encoded_filtered.columns)

scaler = StandardScaler()
df_scaled = df_encoded_filtered.copy()
df_scaled[selected_features] = scaler.fit_transform(df_scaled[selected_features])
input_encoded[selected_features] = scaler.transform(input_encoded[selected_features])

# Optional Weights
weights = {
    'Average_GPA_Requirement': 1.5,
    'IELTS_Requirement': 2,
        'Tuition_Fee_USD': 2,
        'Living_Cost_Per_Year_USD': 1.2
    }

    for feature in selected_features:
        df_scaled[feature] *= weights.get(feature, 1)
        input_encoded[feature] *= weights.get(feature, 1)

    similarities = cosine_similarity(df_scaled[selected_features], input_encoded[selected_features])
    df_filtered = df_filtered.copy()  # Avoid SettingWithCopyWarning
    df_filtered['Similarity'] = similarities

    # Output Recommendations
    top_matches = df_filtered.sort_values(by='Similarity', ascending=False).head(5)
    print("\n🌟 Top 5 Recommended Universities:\n")
    print(top_matches[['University', 'Country', 'Field_Specialization', 'Tuition_Fee_USD', 'Living_Cost_Per_Year_USD', 'Similarity', 'Degree_
```

```
Enter your details to get university recommendations:
Field of Specialization (e.g., Artificial Intelligence): Physics
Degree Level (Bachelor/Master/PhD): PhD
Your GPA (e.g., 3.5): 3
IELTS Score (e.g., 7.0): 7
Max Tuition Fee in USD (e.g., 15000): 1000
Max Living Cost in USD (e.g., 12000): 1500
Scholarship Available (Yes/No): Yes
Preferred Region (e.g., EU, US, Asia): US
⚠️ No universities matched your strict filters. Showing top matches regardless of budget/region...

🌟 Top 5 Recommended Universities:

                            University  Country Field_Specialization  \
1056          University College London       UK              Physics
1160              University of Toronto   Canada              Physics
611   Karlsruhe Institute of Technology  Germany              Physics
1990      Humboldt University of Berlin  Germany              Physics
1492            Imperial College London       UK              Physics

      Tuition_Fee_USD  Living_Cost_Per_Year_USD  Similarity Degree_Level
1056            10247                      8887    0.392677          PhD
1160             6256                      9605    0.379005          PhD
611                 0                      9766    0.341361          PhD
1990                0                      8210    0.338857          PhD
1492             5822                     14435    0.313759          PhD
```