```python
student_transformer = Pipeline(steps=[
    # Map binary categorical variables to 0/1
    ('map_gender', CustomMappingTransformer('gender', {'Male': 0, 'Female': 1, 'Other': 2})),
    ('map_part_time_job', CustomMappingTransformer('part_time_job', {'No': 0, 'Yes': 1})),
    ('map_extracurricular', CustomMappingTransformer('extracurricular_participation', {'No': 0, 'Yes': 1})),

    # Map ordinal categorical variables (preserve order)
    ('map_diet_quality', CustomMappingTransformer('diet_quality', {'Poor': 0, 'Fair': 1, 'Good': 2})),
    ('map_internet_quality', CustomMappingTransformer('internet_quality', {'Poor': 0, 'Average': 1, 'Good': 2})),
    ('map_parental_education', CustomMappingTransformer('parental_education_level',
                                                        {'None': 0, 'High School': 1, 'Bachelor': 2, 'Master': 3})),

    # Apply Tukey outlier detection to numeric features
    ('tukey_age', CustomTukeyTransformer(target_column='age', fence='outer')),
    ('tukey_study_hours', CustomTukeyTransformer(target_column='study_hours_per_day', fence='outer')),
    ('tukey_social_media', CustomTukeyTransformer(target_column='social_media_hours', fence='outer')),
    ('tukey_netflix', CustomTukeyTransformer(target_column='netflix_hours', fence='outer')),
    ('tukey_attendance', CustomTukeyTransformer(target_column='attendance_percentage', fence='outer')),
    ('tukey_sleep', CustomTukeyTransformer(target_column='sleep_hours', fence='outer')),

    # Apply robust scaling to numeric features
    ('scale_age', CustomRobustTransformer(target_column='age')),
    ('scale_study_hours', CustomRobustTransformer(target_column='study_hours_per_day')),
    ('scale_social_media', CustomRobustTransformer(target_column='social_media_hours')),
    ('scale_netflix', CustomRobustTransformer(target_column='netflix_hours')),
    ('scale_attendance', CustomRobustTransformer(target_column='attendance_percentage')),
    ('scale_sleep', CustomRobustTransformer(target_column='sleep_hours')),

    # Drop the original exam_score and student_id columns
    ('drop_columns', CustomDropColumnsTransformer(['exam_score', 'student_id'], action='drop')),

    # Impute any remaining missing values
    ('impute', CustomKNNTransformer(n_neighbors=5)),
], verbose=True)
```