DataExploration

October 30, 2024

[146]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[147]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from ipywidgets import interact, interact_manual
from IPython.display import display
```

[148]:
```
# Load 7 schools dataset
castlemont = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/Castlemont.csv')
eof = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/East Oakland PRIDE.csv')
elmhurst_united = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/Elmhurst United.
 ↪csv')
mlk = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/MLK.csv')
mcClymonds = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/McClymonds.csv')
prescott = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/Prescott.csv')
woms = pd.read_csv('/content/drive/MyDrive/ONGB -Lan/WOMS.csv')
# castlemont.head()
```

[149]:
```
# Concatenate into one dataframe
schools = pd.concat([castlemont, eof, elmhurst_united, mlk, mcClymonds,␣
 ↪prescott, woms]).reset_index(drop=True)
schools.head(5)
```

[149]:

	ANON_ID	Birthdate	Gen	Eth	Fluency	SpEd	Grade	\
0	338	2003-07-21	F	Latino	RFEP	Not Special Ed	9	
1	340	2003-03-31	M	Latino	EL	Not Special Ed	9	
2	478	2003-09-06	F	Multiple Ethnicity	EO	Special Ed	9	
3	686	2000-04-02	M	Latino	EL	Not Special Ed	12	
4	693	2002-03-28	F	Latino	RFEP	Not Special Ed	10	

```
   AttRate  DaysEnr  DaysAbs  Susp  CurrWeightedTotGPA      SED      School  \
0   0.9889    180.0      2.0   NaN                3.65  Unknown  Castlemont
1   0.8389    180.0     29.0   2.0                0.06  Unknown  Castlemont
2   0.7263    179.0     49.0   1.0                0.24  Unknown  Castlemont
3   0.9611    180.0      7.0   NaN                2.00  Unknown  Castlemont
4   0.9889    180.0      2.0   NaN                2.59  Unknown  Castlemont

    Year
0  17-18
1  17-18
2  17-18
3  17-18
4  17-18
```

[150]:
```
# Add age (school year - birthdate) column
def calculate_age(df):
  df['Birthdate'] = pd.to_datetime(df['Birthdate'])
  end_year = df['Year'].str.split('-').str[0].astype(int)
  year = end_year.apply(lambda x: 2000 + x)
  df['Age'] = year - df['Birthdate'].dt.year
  return df

schools = calculate_age(schools)
schools.head(5)
```

[150]:
```
   ANON_ID  Birthdate Gen                Eth Fluency            SpEd  Grade  \
0      338 2003-07-21   F             Latino    RFEP  Not Special Ed      9
1      340 2003-03-31   M             Latino      EL  Not Special Ed      9
2      478 2003-09-06   F  Multiple Ethnicity      EO      Special Ed      9
3      686 2000-04-02   M             Latino      EL  Not Special Ed     12
4      693 2002-03-28   F             Latino    RFEP  Not Special Ed     10

   AttRate  DaysEnr  DaysAbs  Susp  CurrWeightedTotGPA      SED      School  \
0   0.9889    180.0      2.0   NaN                3.65  Unknown  Castlemont
1   0.8389    180.0     29.0   2.0                0.06  Unknown  Castlemont
2   0.7263    179.0     49.0   1.0                0.24  Unknown  Castlemont
3   0.9611    180.0      7.0   NaN                2.00  Unknown  Castlemont
4   0.9889    180.0      2.0   NaN                2.59  Unknown  Castlemont

    Year  Age
0  17-18   14
1  17-18   14
2  17-18   14
3  17-18   17
4  17-18   15
```

[151]:
```
# Add ChroAbs column with 1 is student who absent for more than 10% of enrolled days
abs_per = schools['DaysAbs'] / schools['DaysEnr']
schools['ChroAbs'] = [1 if x > 0.1 else 0 for x in abs_per]
schools.tail(5)
```

[151]:

	ANON_ID	Birthdate	Gen	Eth	Fluency	SpEd	\
21989	77312	2009-10-26	M	African American	EO	Not Special Ed	
21990	77803	2011-06-30	M	Latino	EO	Special Ed	
21991	78129	2011-10-09	F	Latino	RFEP	Not Special Ed	
21992	78498	2011-06-21	M	African American	EO	Not Special Ed	
21993	79427	2012-01-10	M	Latino	RFEP	Not Special Ed	

	Grade	AttRate	DaysEnr	DaysAbs	Susp	CurrWeightedTotGPA	SED	\
21989	8	0.9611	180.0	7.0	NaN	2.39	SED	
21990	7	0.6222	45.0	17.0	NaN	0.00	SED	
21991	6	0.9441	179.0	10.0	NaN	3.38	Not SED	
21992	7	0.9333	180.0	12.0	NaN	3.66	SED	
21993	6	0.7375	80.0	21.0	NaN	3.37	SED	

	School	Year	Age	ChroAbs
21989	WOMS	23-24	14	0
21990	WOMS	23-24	12	1
21991	WOMS	23-24	12	0
21992	WOMS	23-24	12	0
21993	WOMS	23-24	11	1

0.1 1. Dataset Overview

[152]:
```
# Shape of dataframe
schools.shape
```

[152]: (21994, 17)

[153]:
```
# Shape of each school
for s in schools['School'].unique():
  print(s, schools[schools['School'] == s].shape)
```

```
Castlemont (6456, 17)
East Oakland PRIDE (2634, 17)
Elmhurst United (4989, 17)
MLK (2720, 17)
McClymonds (2566, 17)
Prescott (1079, 17)
WOMS (1550, 17)
```

[154]:
```
# Shape of each school and each year
for s in schools['School'].unique():
  for y in schools['Year'].unique():
    print(s, y, schools[(schools['School'] == s) & (schools['Year'] == y)].
  ↪shape)
```

```
Castlemont 17-18 (984, 17)
Castlemont 18-19 (994, 17)
Castlemont 19-20 (931, 17)
Castlemont 20-21 (874, 17)
Castlemont 21-22 (896, 17)
Castlemont 22-23 (903, 17)
Castlemont 23-24 (874, 17)
East Oakland PRIDE 17-18 (402, 17)
East Oakland PRIDE 18-19 (406, 17)
East Oakland PRIDE 19-20 (393, 17)
East Oakland PRIDE 20-21 (351, 17)
East Oakland PRIDE 21-22 (353, 17)
East Oakland PRIDE 22-23 (353, 17)
East Oakland PRIDE 23-24 (376, 17)
Elmhurst United 17-18 (429, 17)
Elmhurst United 18-19 (432, 17)
Elmhurst United 19-20 (825, 17)
Elmhurst United 20-21 (805, 17)
Elmhurst United 21-22 (826, 17)
Elmhurst United 22-23 (839, 17)
Elmhurst United 23-24 (833, 17)
MLK 17-18 (321, 17)
MLK 18-19 (380, 17)
MLK 19-20 (453, 17)
MLK 20-21 (397, 17)
MLK 21-22 (397, 17)
MLK 22-23 (379, 17)
MLK 23-24 (393, 17)
McClymonds 17-18 (419, 17)
McClymonds 18-19 (408, 17)
McClymonds 19-20 (391, 17)
McClymonds 20-21 (400, 17)
McClymonds 21-22 (357, 17)
McClymonds 22-23 (302, 17)
McClymonds 23-24 (289, 17)
Prescott 17-18 (209, 17)
Prescott 18-19 (182, 17)
Prescott 19-20 (147, 17)
Prescott 20-21 (135, 17)
Prescott 21-22 (119, 17)
Prescott 22-23 (127, 17)
```

```
Prescott 23-24 (160, 17)
WOMS 17-18 (218, 17)
WOMS 18-19 (232, 17)
WOMS 19-20 (235, 17)
WOMS 20-21 (235, 17)
WOMS 21-22 (221, 17)
WOMS 22-23 (210, 17)
WOMS 23-24 (199, 17)
```

[363]:
```python
from IPython.display import display, HTML
# Generate HTML for each column's value counts
html_output = "<div style='display: flex; flex-wrap: wrap;'>"

for col in cat_cols:
    # Get the value counts for the column
    counts = schools[col].value_counts()

    # Create an HTML table for the value counts of each column
    html_output += f"""
    <div style='margin-right: 20px; padding: 10px; border: 1px solid #ddd;'>
        <h4>{col}</h4>
        {counts.to_frame().to_html(header=False)}
    </div>
    """

html_output += "</div>"

# Display all counts side by side in HTML format
display(HTML(html_output))
```

```
<IPython.core.display.HTML object>
```

[157]:
```python
# Info for all schools over past 7 years
schools.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21994 entries, 0 to 21993
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   ANON_ID             21994 non-null  int64
 1   Birthdate           21994 non-null  datetime64[ns]
 2   Gen                 21994 non-null  object
 3   Eth                 21994 non-null  object
 4   Fluency             21994 non-null  object
 5   SpEd                21994 non-null  object
 6   Grade               21994 non-null  int64
 7   AttRate             21965 non-null  float64
```

```
 8   DaysEnr              21965 non-null  float64
 9   DaysAbs              21965 non-null  float64
 10  Susp                 1438 non-null   float64
 11  CurrWeightedTotGPA   15553 non-null  float64
 12  SED                  21994 non-null  object
 13  School               21994 non-null  object
 14  Year                 21994 non-null  object
 15  Age                  21994 non-null  int64
 16  ChroAbs              21994 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(4), object(7)
memory usage: 2.9+ MB
```

[365]:
```
# Missing values total
schools.isna().sum()
```

[365]:
```
ANON_ID                 0
Birthdate               0
Gen                     0
Eth                     0
Fluency                 0
SpEd                    0
Grade                   0
AttRate                29
DaysEnr                29
DaysAbs                29
Susp                20556
CurrWeightedTotGPA     6441
SED                       0
School                    0
Year                      0
Age                       0
ChroAbs                   0
dtype: int64
```

[366]:
```
# Missing values percentage
schools.isna().mean()
```

[366]:
```
ANON_ID               0.000000
Birthdate             0.000000
Gen                   0.000000
Eth                   0.000000
Fluency               0.000000
SpEd                  0.000000
Grade                 0.000000
AttRate               0.001319
DaysEnr               0.001319
DaysAbs               0.001319
```

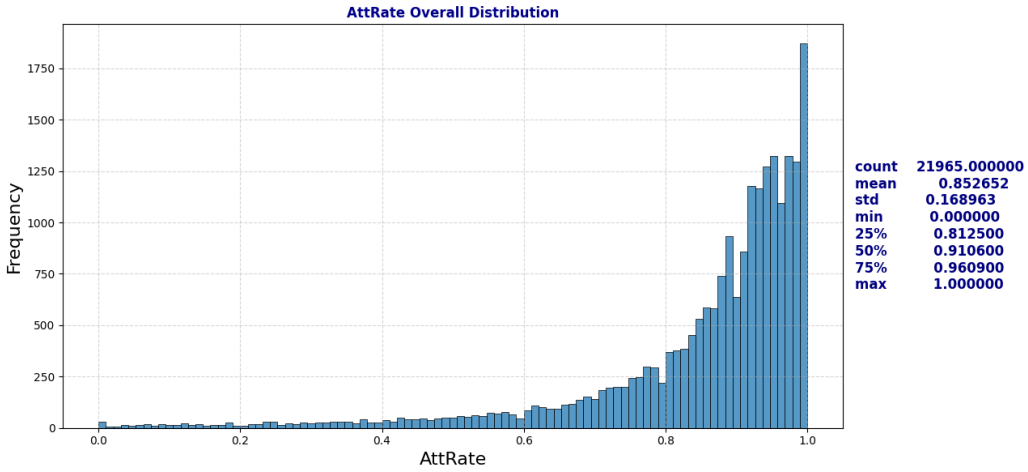
```
Susp                  0.934619
CurrWeightedTotGPA    0.292853
SED                   0.000000
School                0.000000
Year                  0.000000
Age                   0.000000
ChroAbs               0.000000
dtype: float64
```

0.2 2. Exploratory Data Analysis

0.3 2.a Numerical Distributions: Attendance Rate, Days Enrolled, Days Absent

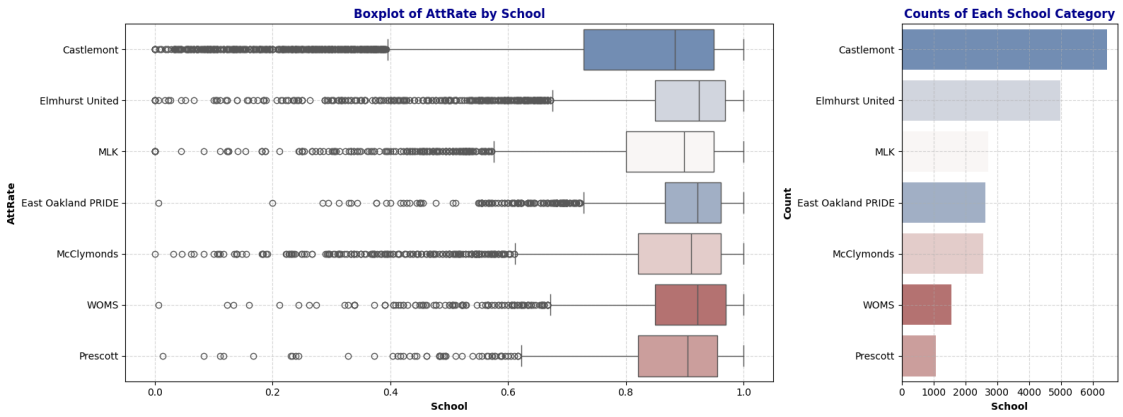
[461]:
```
def plot_distribution_and_summary(df, col, title=None):
    """
    Plots the distribution histogram and statistical summary for a specified␣
 ↪column.

    Parameters:
    - df: DataFrame containing the data
    - col: str, the name of the column to plot and summarize
    - title: modified title for the plot
    """
    # Calculate the statistical summary for the specified column
    summary_stats = df[col].describe()

    # Set up a figure with 2 subplots: 1 for the plot, 1 for the text
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6),␣
 ↪gridspec_kw={'width_ratios': [3, 1]})

    # Plot the histogram for the specified column
    if title:
        ax1.set_title(title, fontweight='bold', color='darkblue')
    else:
        ax1.set_title(f'{col} Overall Distribution', fontweight='bold',␣
 ↪color='darkblue')
    ax1.set_xlabel(f'{col}', fontsize=16)
    ax1.set_ylabel('Frequency', fontsize=16)
    sns.histplot(data=df, x=col, ax=ax1)
    ax1.grid(True, linestyle='--', alpha=0.5)

    # Display the statistical summary on the right subplot
    ax2.axis('off')  # Turn off the axis for the text display
    summary_text = summary_stats.to_string()
    ax2.text(0, 0.5, summary_text, fontsize=12, va='center', ha='left',␣
 ↪fontweight='bold', color='navy')
```

```
    # Adjust layout and show the plot
    plt.tight_layout()
    plt.show()

# Attendance Overall Distribution
plot_distribution_and_summary(schools, 'AttRate')
```

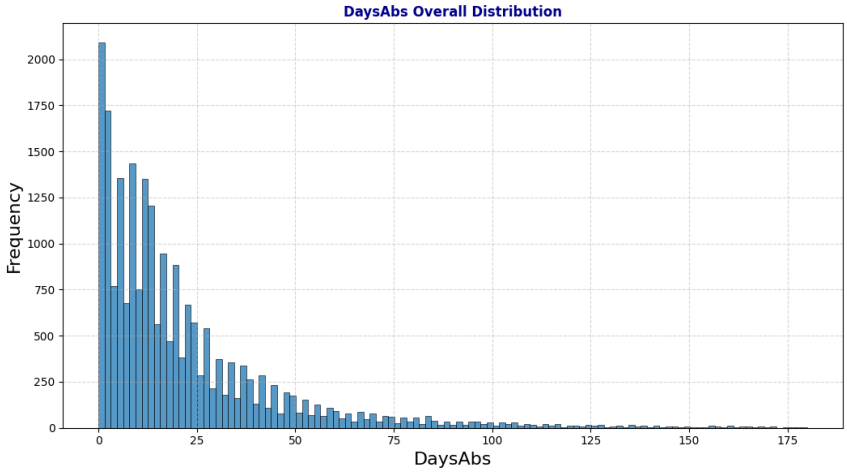


[415]:
```
def plot_boxplot_with_counts(df, numerical_col, categorical_col):
    """
    Plots a boxplot for a numerical column grouped by a categorical column
    along with the count of each category.

    Parameters:
    - df: DataFrame containing the data
    - numerical_col: str, the name of the numerical column to plot
    - categorical_col: str, the name of the categorical column to group by
    """

    # Define the category order based on the boxplot
    category_order = df[categorical_col].value_counts().index.tolist()

    # Set up the figure with two subplots: one for the boxplot and one for the␣
 ↪bar plot
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6),␣
 ↪gridspec_kw={'width_ratios': [3, 1]})

    # Plot the boxplot for the numerical column by the categorical column
    sns.boxplot(data=df, y=categorical_col, x=numerical_col, ax=ax1, hue␣
 ↪=categorical_col, palette="vlag", order=category_order)
```

```
    ax1.set_title(f'Boxplot of {numerical_col} by {categorical_col}',␣
 ↪fontweight='bold', color='darkblue')
    ax1.set_xlabel(categorical_col, fontweight='bold')
    ax1.set_ylabel(numerical_col, fontweight='bold')
    ax1.grid(True, linestyle='--', alpha=0.5)

    # Plot the count of each category as a bar plot
    sns.countplot(data=df, y=categorical_col, order=category_order, ax=ax2,␣
 ↪hue=categorical_col, palette='vlag')
    ax2.set_title(f'Counts of Each {categorical_col} Category',␣
 ↪fontweight='bold', color='darkblue')
    ax2.set_xlabel(categorical_col, fontweight='bold')
    ax2.set_ylabel('Count', fontweight='bold')
    ax2.grid(True, linestyle='--', alpha=0.5)

    # Adjust layout and display the plots
    plt.tight_layout()
    plt.show()

# Attendance Distributions by School
plot_boxplot_with_counts(schools, 'AttRate', 'School')
```

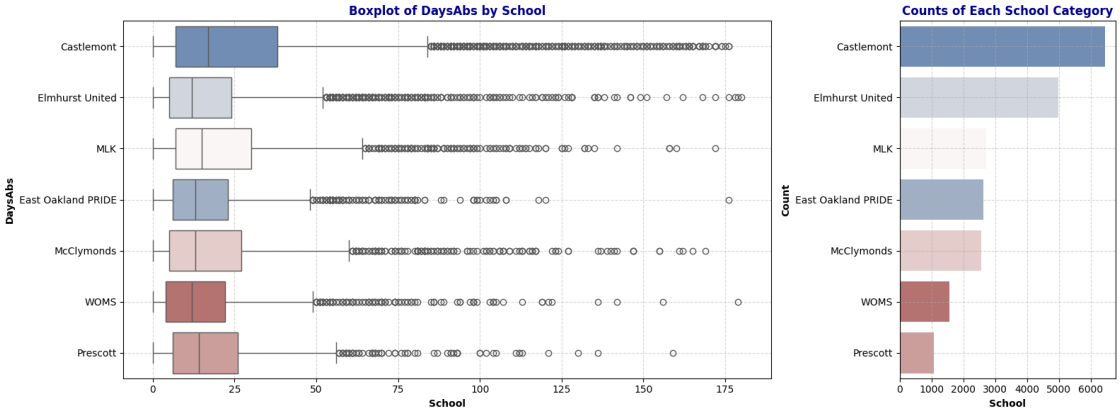


[462]:
```
# Days Absent Overall Distribution
plot_distribution_and_summary(schools, 'DaysAbs')
```



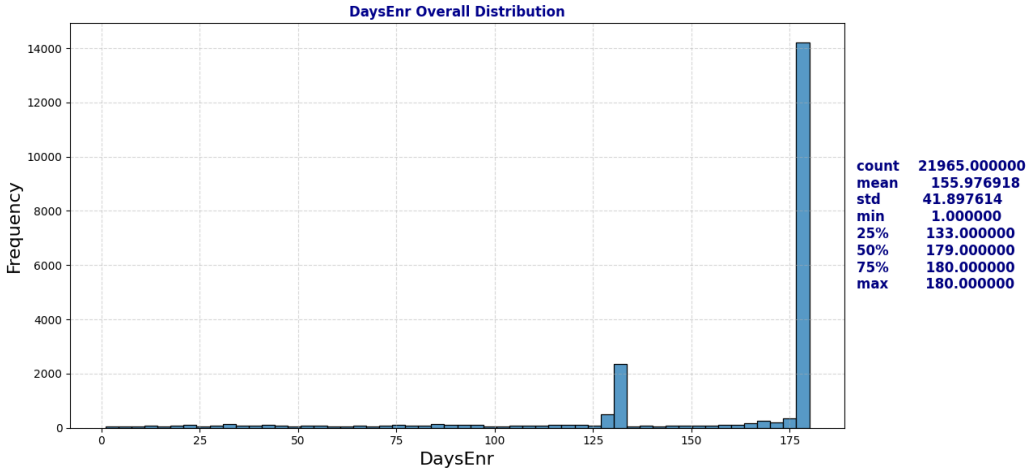

```
count    21965.000000
mean        21.738402
std         24.897842
min          0.000000
25%          6.000000
50%         14.000000
75%         28.000000
max        180.000000
```

[419]:
```
# Days Absent Distribution by School
plot_boxplot_with_counts(schools, 'DaysAbs', 'School')
```

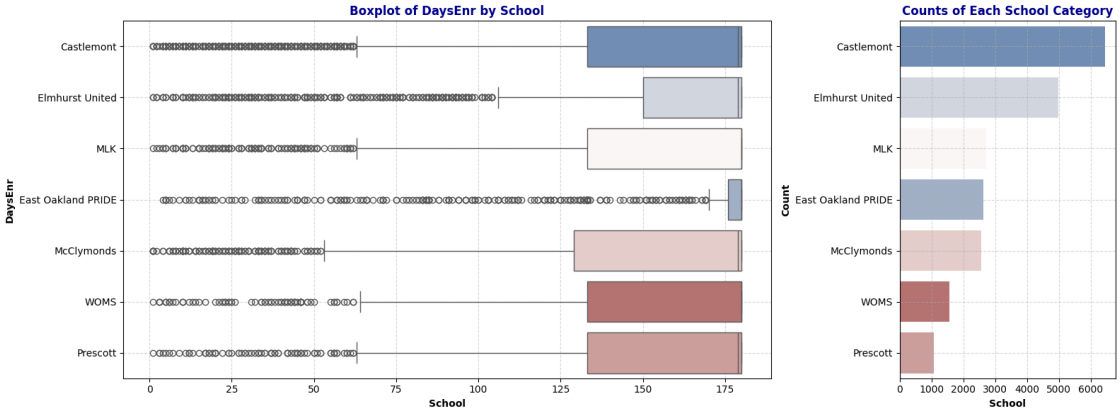


Observation: The distribution of Days Absent closely mirrors that of Attendance Rate, indicating a strong dependency between the two. We can consider dropping the DaysAbs column later to avoid redundancy.

[459]:
```
# Days Enrolled Overall Distribution
plot_distribution_and_summary(schools, 'DaysEnr')
```



[417]:
```
# Days Enrolled Distribution by School
plot_boxplot_with_counts(schools, 'DaysEnr', 'School')
```

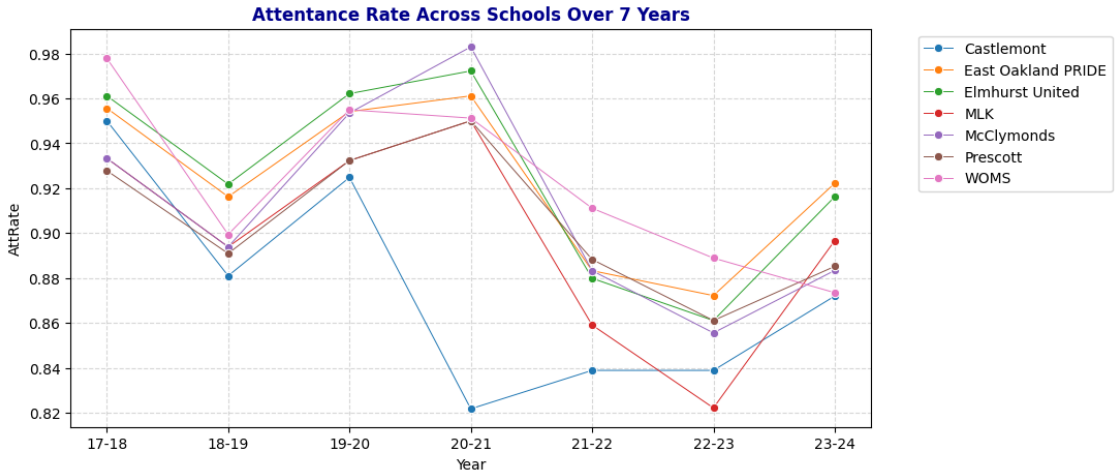


Observation: The Days Enrolled column contains numerous unusually low values. This raises questions about the underlying causes (further research on how schools report days enrolled may be necessary). We may also need to reconsider the definition of chronic absenteeism (where the ratio of days absent to days enrolled is greater than 10%), as low values for days enrolled could disproportionately impact the chronic absenteeism rate.

0.4 2b. Time Series over Year: Attendance Rate and Chronic Absenteeism Percentage

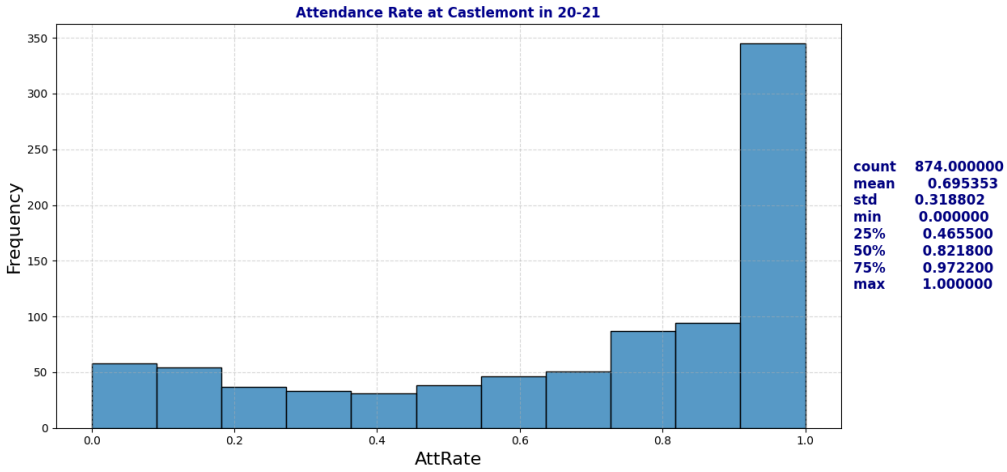
[429]:
```
# Time series plot for attendate rate, aggregating by median
plt.figure(figsize=(10, 5))
plt.title('Attentance Rate Across Schools Over 7 Years', fontweight='bold',␣
 ↪color='darkblue')
```

```
sns.lineplot(data=schools, x='Year', y='AttRate', hue='School',␣
 ↪estimator='median', errorbar=None,  marker='o', linewidth=0.7)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1));
```

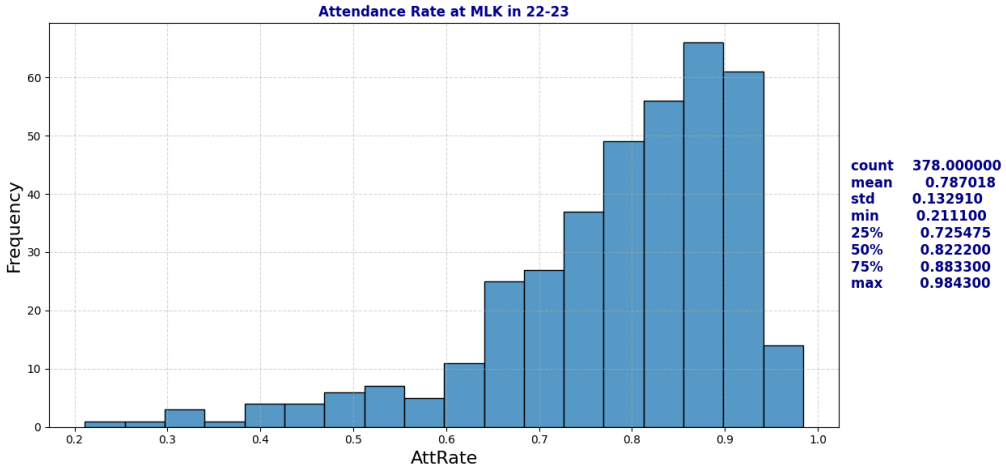


0.4.1 Dive deeper into MLK and Castlemont Attendance Rate Distribution

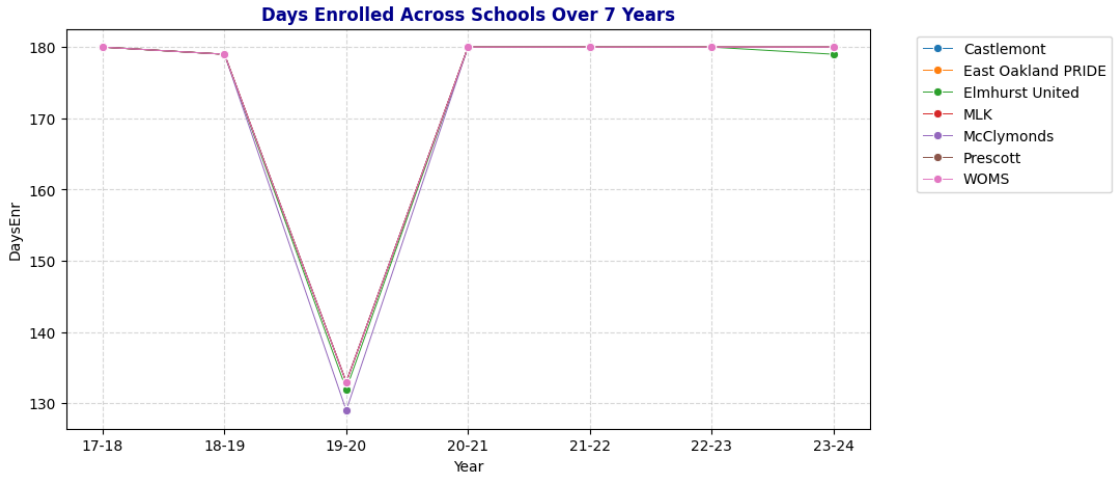
[463]:
```
# Examine attendance rate at Castlemont, espcially year 20-21
plot_distribution_and_summary(schools[(schools['School'] == 'Castlemont') &␣
 ↪(schools['Year'] == '20-21')], 'AttRate', "Attendance Rate at Castlemont in␣
 ↪20-21")
```



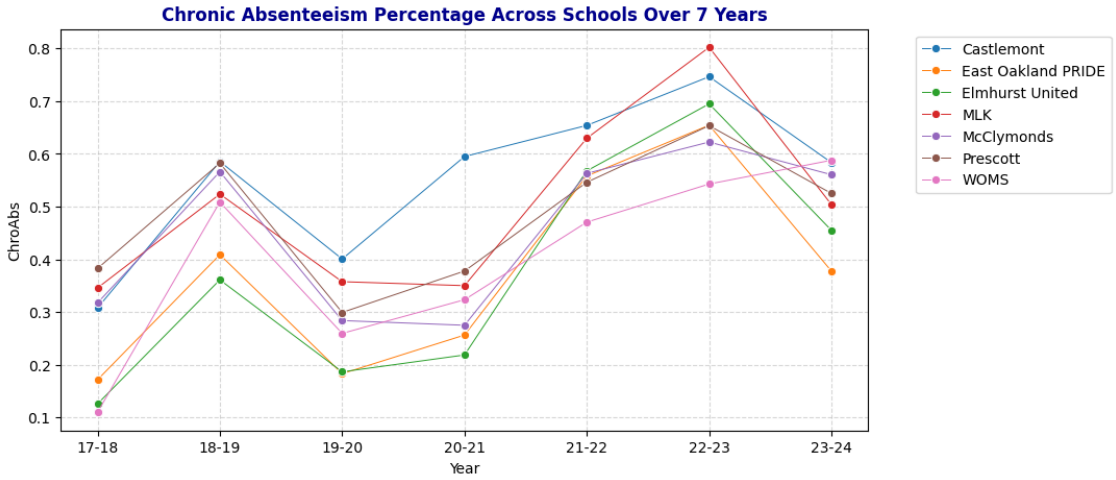
[464]:
```
# Examine attendance rate at MLK, espcially year 22-23
plot_distribution_and_summary(schools[(schools['School'] == 'MLK') &␣
 ↪(schools['Year'] == '22-23')], 'AttRate', "Attendance Rate at MLK in 22-23")
```



[447]:
```
# Time series plot for Days Enrolled, aggregating by median
plt.figure(figsize=(10, 5))
plt.title('Days Enrolled Across Schools Over 7 Years', fontweight='bold',␣
 ↪color='darkblue')
sns.lineplot(data=schools, x='Year', y='DaysEnr', hue='School',␣
 ↪estimator='median', errorbar=None,  marker='o', linewidth=0.7)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
plt.grid(True, linestyle='--', alpha=0.5);
```



[454]:
```
# ChronicAbsent Percentage by Year
plt.figure(figsize=(10, 5))
plt.title('Chronic Absenteeism Percentage Across Schools Over 7 Years',␣
 ↪fontweight='bold', color='darkblue')
sns.lineplot(data=schools, x='Year', y='ChroAbs', hue='School',␣
 ↪estimator='mean', errorbar=None,  marker='o', linewidth=0.7)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
plt.grid(True, linestyle='--', alpha=0.5);
```

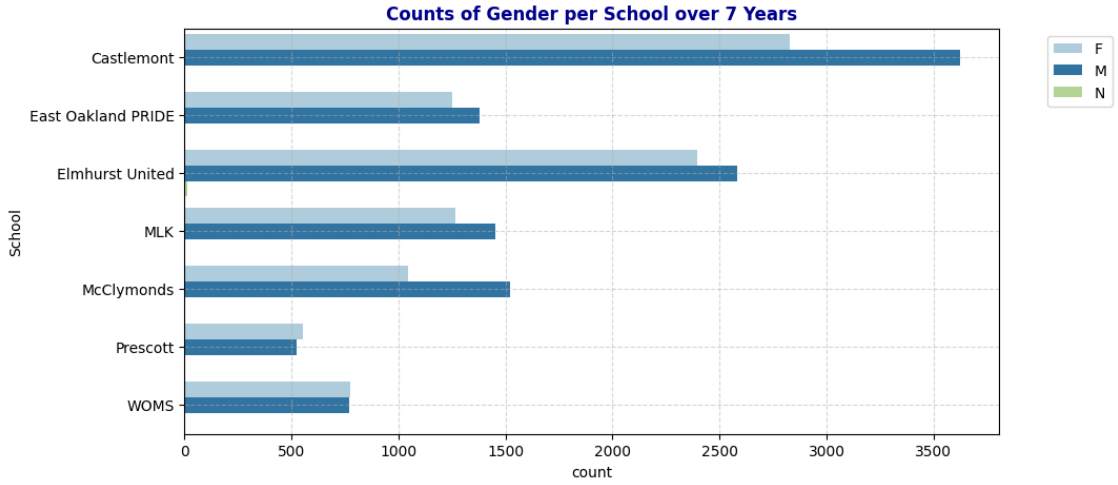


Quick observation: Even though 19-20 has lowest median days enrolled but the percentage of chronically absent student are under 40%. School years 21-22 and 22-23 have high chronic absenteeism percentage.

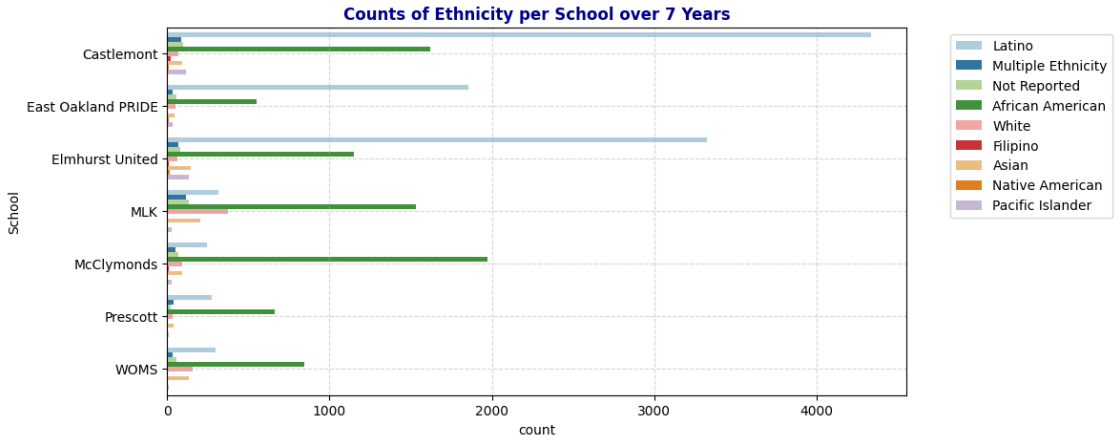
0.5 2c. Demographic Factors: Gender, Ethnicity, Fluency, Special Education and Socio-economically Disadvantaged Status

[481]:
```
def plot_bar(df, col, title=None):
    """
    Plots a bar chart for a specified column.

    Parameters:
    - df: DataFrame containing the data
    - col: str, the name of the column to plot
    - title: modified title for the plot
    """
    plt.figure(figsize=(10, 5))
    if title:
        plt.title(title, fontweight='bold', color='darkblue')
    else:
```

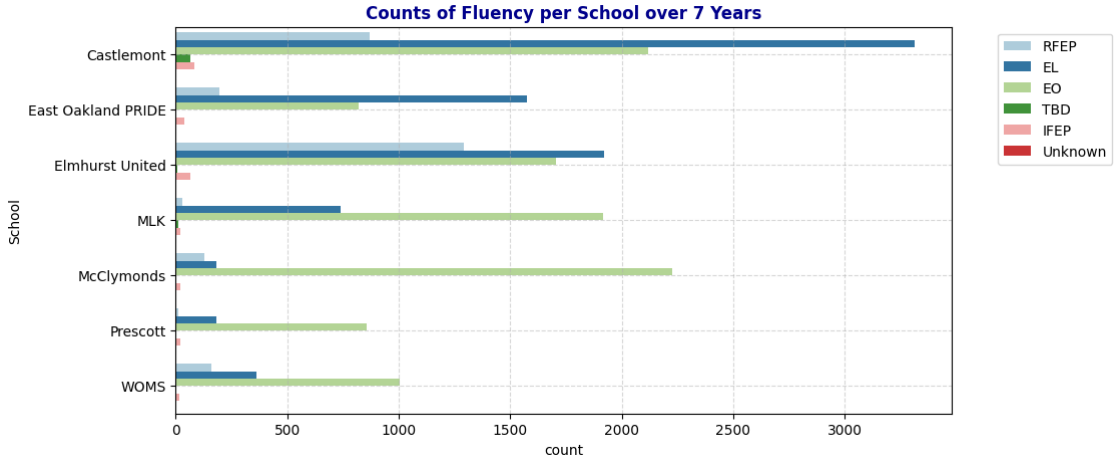
```
        plt.title(f'Counts of {col} per School over 7 Years',␣
↪fontweight='bold', color='darkblue')
    sns.countplot(data=df, y='School', hue=col, palette='Paired')
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
    plt.show()

# Bar plot of Gender per School
plot_bar(schools, 'Gen', "Counts of Gender per School over 7 Years")
```



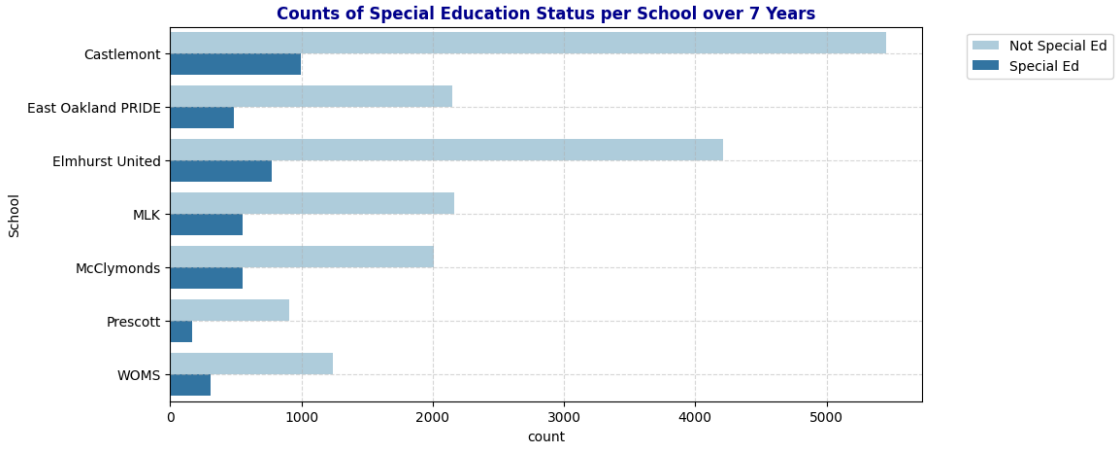
[480]:
```
# Bar plot of Ethnicity per School
plot_bar(schools, 'Eth', "Counts of Ethnicity per School over 7 Years")
```



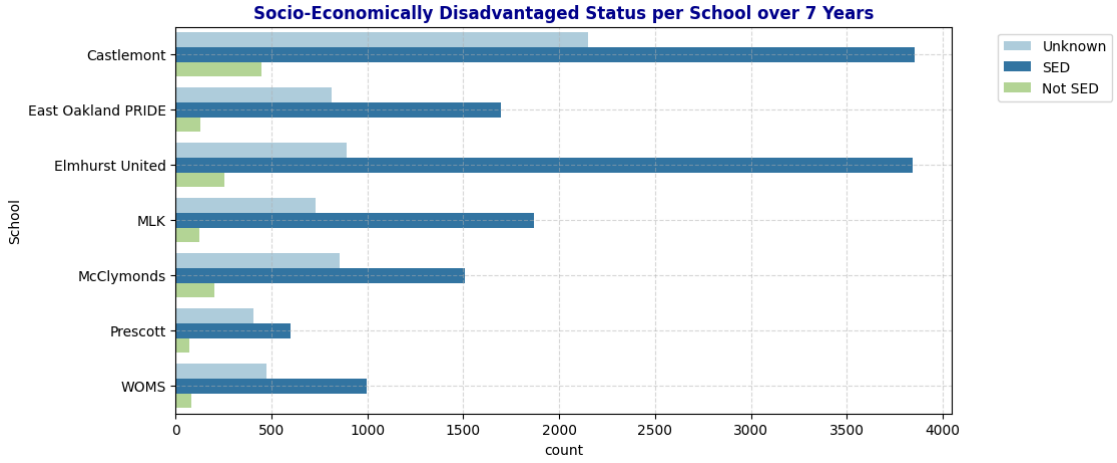
[478]:
```
# Bar plot of Fluency per School
plot_bar(schools, 'Fluency')
```



[483]:
```
# Bar plot of Special Education per School
plot_bar(schools, 'SpEd', "Counts of Special Education Status per School over 7␣
 ↪Years")
```



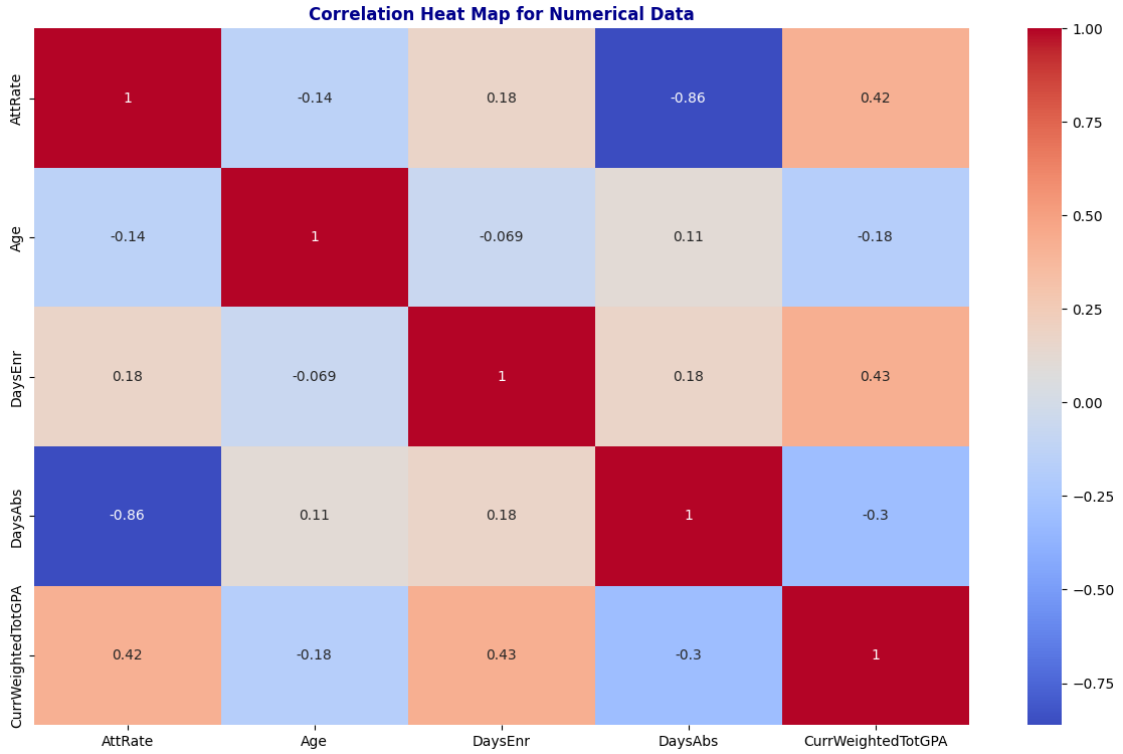
[479]:
```
# Bar plot of Socio-Economically Disadvantaged Counts per school
plot_bar(schools, 'SED', "Socio-Economically Disadvantaged Status per School␣
 ↪over 7 Years")
```



1 Heat Map

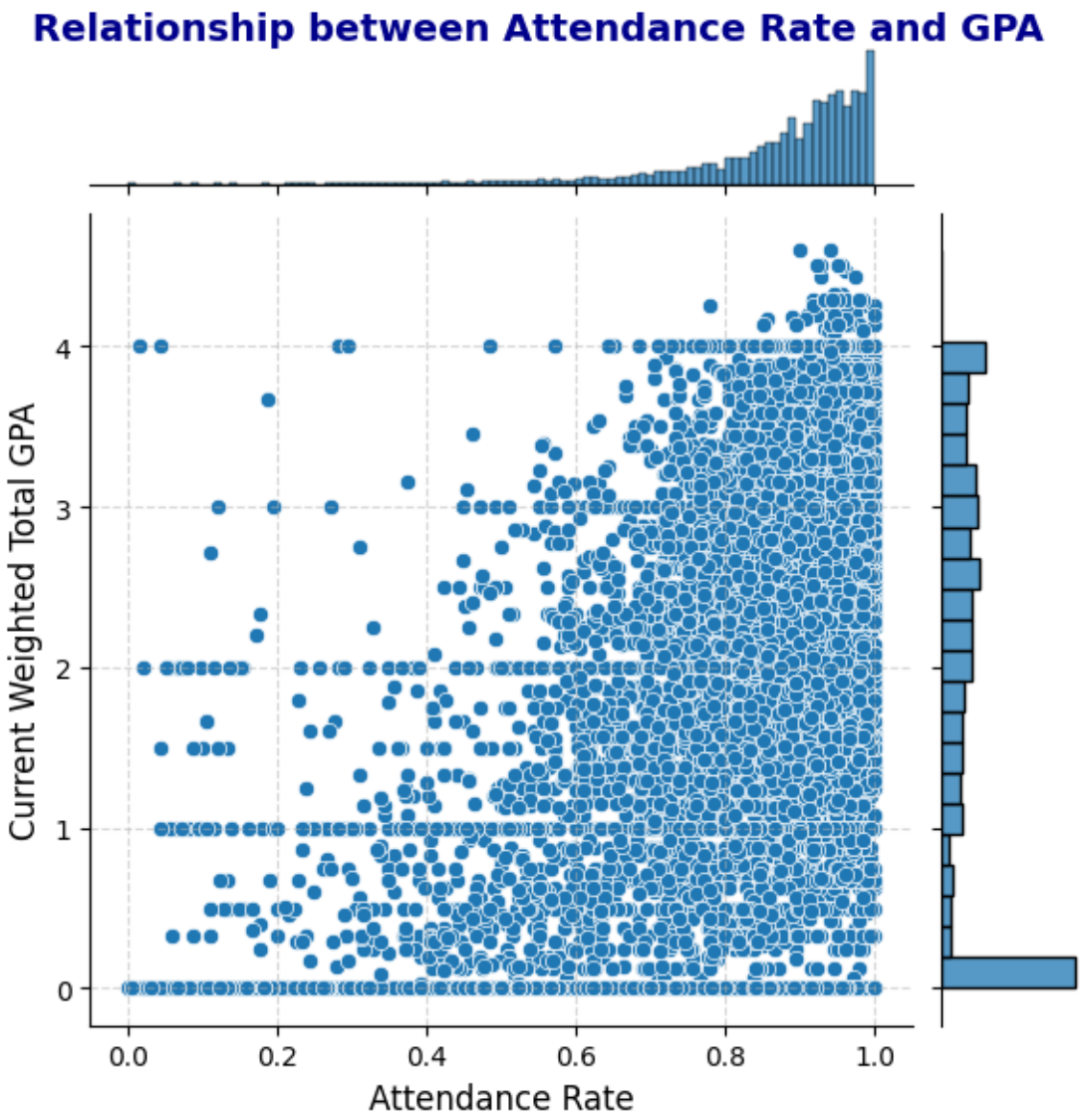
1.0.1 Numerical Data: Pearson's Correlation to Visualize Linear Relationship of Each Pair.

[491]:
```
def plot_corr(cols):
  corr_df = schools[cols]
  corr_df.corr()
  plt.figure(figsize=(15, 9))
  sns.heatmap(corr_df.corr(), cmap='coolwarm', annot=True)
  plt.title(f'Correlation Heat Map for Numerical Data', fontweight='bold',␣
 ↪color='darkblue');

# Numerical data for correlation map
corr_cols = ['AttRate', 'Age', 'DaysEnr', 'DaysAbs', 'CurrWeightedTotGPA']
plot_corr(corr_cols)
```



Observation: Pairs that has high correlation 1. (DaysAbs, AttRate): This is because of their dependency. 2. (DaysEnr, CurrWeightedTotGPA): positive linear relationship with correlation = 0.43. Not interested in this since it does not relate to our problem of chronic absenteeism. 3. (AttRate, CurrWeightedTotGPA): postive linear relationship with correlation = 0.42. Will investigate this further.

[495]:
```
# Examine the relationship between Attendance Rate and GPA
joint_plot = sns.jointplot(data=schools, x='AttRate', y='CurrWeightedTotGPA',␣
 ↪kind="scatter")
joint_plot.set_axis_labels('Attendance Rate', 'Current Weighted Total GPA',␣
 ↪fontsize=12)
joint_plot.fig.suptitle('Relationship between Attendance Rate and GPA',␣
 ↪fontsize=14, fontweight='bold', color='darkblue')
joint_plot.fig.subplots_adjust(top=0.95)  # Adjust the top margin to make room␣
 ↪for the title
plt.grid(True, linestyle='--', alpha=0.5);
```



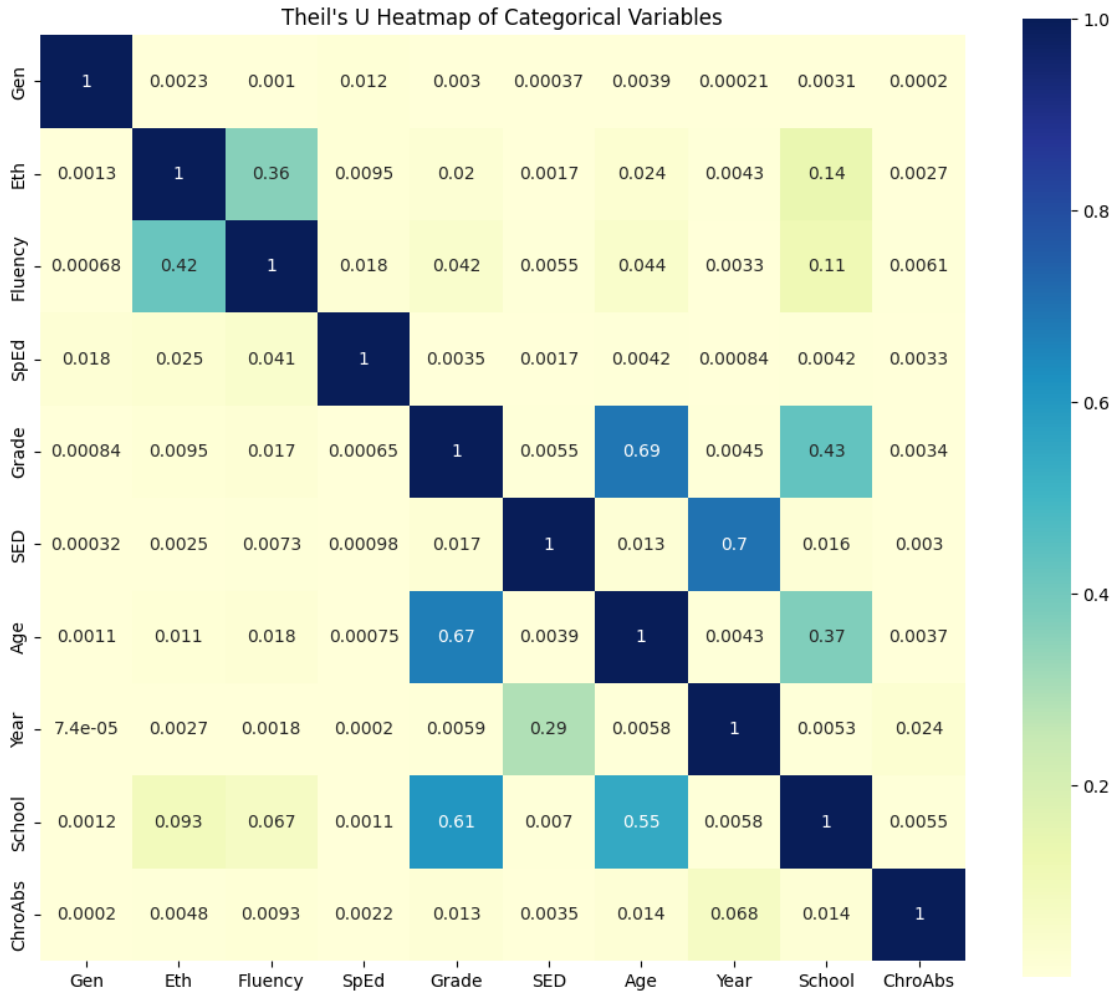
1.0.2 Categorical Correlations: Using Theil's U to Visualize Predictive Strength between Two Categorical Variables

Using **Theil's U** to see relationship between categorical variables, range is [0,1] and asymmetric.

More on this: https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9

[324]:
```
# Theil'U Heat Map for Categorical Variable
from collections import Counter
import math

# Define Theil's U Calculation Function
def conditional_entropy(x, y):
```

```
    y_counter = Counter(y)
    xy_counter = Counter(list(zip(x, y)))
    total_occurrences = sum(y_counter.values())
    entropy = 0.0
    for xy, xy_count in xy_counter.items():
        p_xy = xy_count / total_occurrences
        p_y = y_counter[xy[1]] / total_occurrences
        entropy += p_xy * math.log(p_y / p_xy)
    return entropy

def theils_u(x, y):
    s_xy = conditional_entropy(x, y)
    x_counter = Counter(x)
    total_occurrences = sum(x_counter.values())
    p_x = [x_count / total_occurrences for x_count in x_counter.values()]
    s_x = -sum([p * math.log(p) for p in p_x])
    return (s_x - s_xy) / s_x if s_x != 0 else 1
cat_cols = ['Gen', 'Eth', 'Fluency', 'SpEd', 'Grade', 'SED', 'Age', 'Year',␣
 ↪'School', 'ChroAbs']

# Create an empty DataFrame to store Theil's U values
theils_u_matrix = pd.DataFrame(index=cat_cols, columns=cat_cols)

# Fill the matrix with Theil's U values
for col1 in cat_cols:
    for col2 in cat_cols:
        if col1 == col2:
            theils_u_matrix.loc[col1, col2] = 1  # Perfect association with␣
 ↪itself
        else:
            theils_u_matrix.loc[col1, col2] = theils_u(schools[col1],␣
 ↪schools[col2])

# Convert to float for heatmap compatibility
theils_u_matrix = theils_u_matrix.astype(float)

# Plot the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(theils_u_matrix, annot=True, cmap="YlGnBu", square=True)
plt.title("Theil's U Heatmap of Categorical Variables")
plt.show()
```

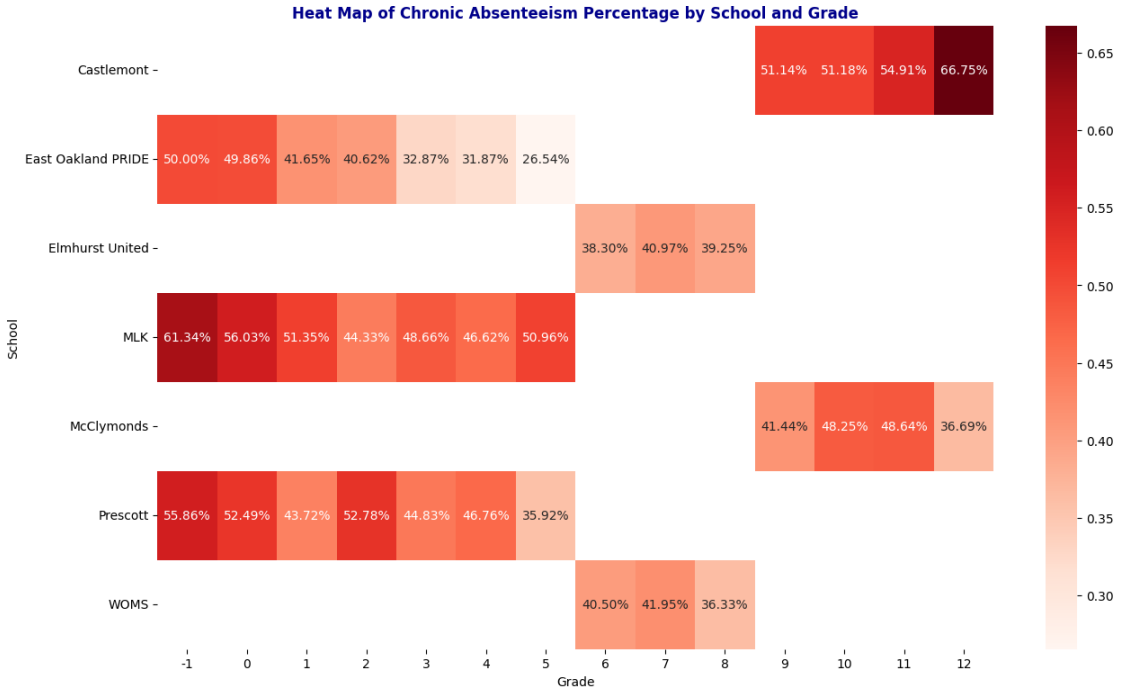


Observation: The factors with the strongest predictive power on Chronic Absenteeism are ranked as follows: Year > School > Age > Grade > Fluency > Ethnicity > SED > SpEd > Gender.

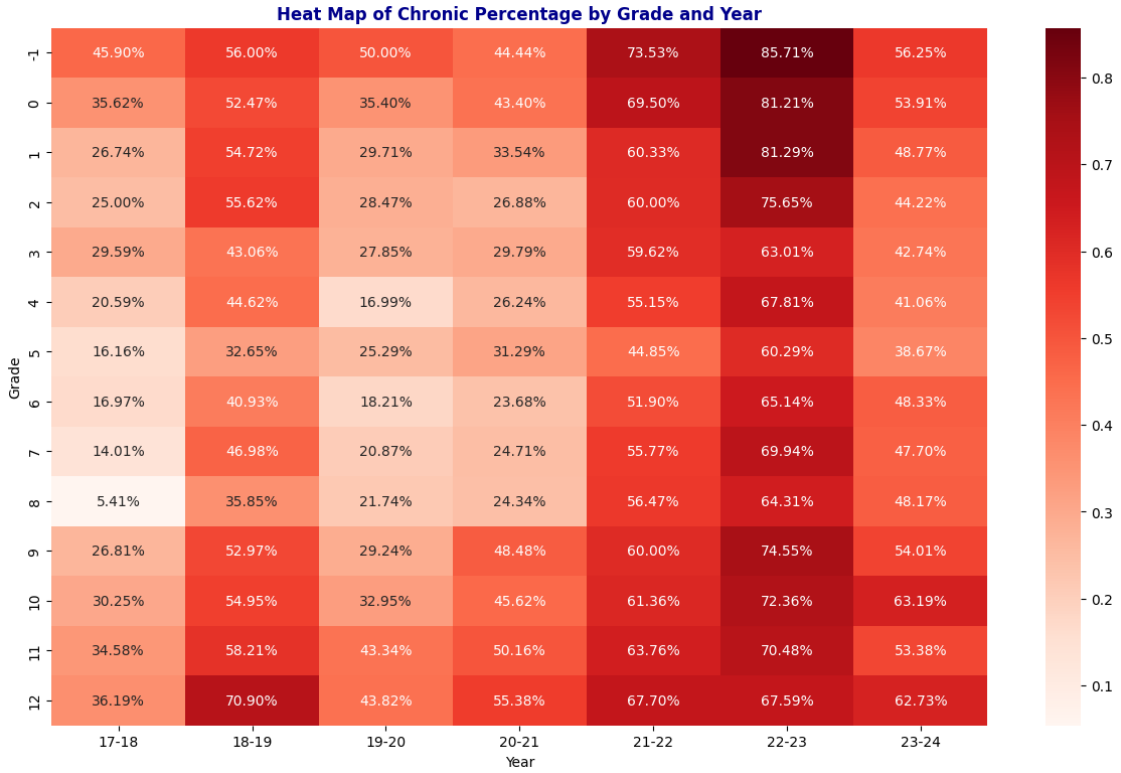
Additionally, the map shows that Grade has a strong predictive relationship with both School and Age (correlations of 0.81 and 0.67, respectively). This is logical, as schools are divided into levels (e.g., Elementary, Middle, High School), each covering specific grade ranges. Similarly, students are typically placed in grades based on their age. This alignment is reflected in the fact that these three variables—Grade, School, and Age—have similar correlation values with Chronic Absenteeism, indicating they offer comparable predictive power for absenteeism. Therefore, when building a predictive model, it may be beneficial to select only one of these variables to avoid redundancy and reduce model complexity.

Another noteworthy observation is the moderate predictive power of SED on Year (0.29), while Year has a stronger predictive power on SED (0.7). Further investigation is needed to understand the underlying reasons for this relationship.

[500]:
```
# Heat Map of Chronic Absenteeism Percentage by School and Year
chroabs_per_school = schools.pivot_table(index='School', columns='Grade',␣
 ↪values='ChroAbs', aggfunc='mean')


plt.figure(figsize=(15, 9))
sns.heatmap(chroabs_per_school, cmap='Reds', annot=True, fmt='.2%')
plt.title('Heat Map of Chronic Absenteeism Percentage by School and Grade',␣
 ↪fontweight='bold', color='darkblue');
```



[498]:
```
# Heat Map of Chronic Percentage by Grade and Year
chroabs_per_grade = schools.pivot_table(index='Grade', columns='Year',␣
 ↪values='ChroAbs', aggfunc='mean')

plt.figure(figsize=(15, 9))
sns.heatmap(chroabs_per_grade, cmap='Reds', annot=True, fmt='.2%')
plt.title('Heat Map of Chronic Percentage by Grade and Year',fontweight='bold',␣
 ↪color='darkblue');
```



2 3. Data Cleaning

##3a. Handle Missing Values

[501]:
```
schools.isna().sum()
```

[501]:
```
ANON_ID                0
Birthdate              0
Gen                    0
Eth                    0
Fluency                0
SpEd                   0
Grade                  0
AttRate               29
DaysEnr               29
DaysAbs               29
Susp                20556
CurrWeightedTotGPA   6441
SED                    0
School                 0
Year                   0
Age                    0
```

```
ChroAbs                    0
dtype: int64
```

We need to handle misisng values in columns `AttRate`,`DaysEnr`, `DaysAbs`,`Susp`, and `CurrWeightedTotGPA`

2.0.1 Suspension: Fill all missing values with 0, as students with no suspension records from schools are assumed to have none.

[276]:
```
schools_cleaned = schools.copy()
```

[278]:
```
schools_cleaned['Susp'] = schools_cleaned['Susp'].fillna(0)
schools_cleaned.isna().sum()
```

[278]:
```
ANON_ID                 0
Birthdate               0
Gen                     0
Eth                     0
Fluency                 0
SpEd                    0
Grade                   0
AttRate                29
DaysEnr                29
DaysAbs                29
Susp                    0
CurrWeightedTotGPA    6441
SED                     0
School                  0
Year                    0
Age                     0
ChroAbs                 0
dtype: int64
```

2.0.2 AttRate, DaysEnr, and Days Abs: Some students have records for the following year, meaning missing values for these fields are likely from newly enrolled students. Some students appear only once, suggesting they did not stay enrolled the following year. Since the percentage of these students per school is low, drop those rows with missing values in these columns, as they do not provide meaningful information and do not significantly impact the overall dataset.

[ ]:
```
# Look at 29 students with NaN Values in AttRate, DaysEnr, and DaysAbs and how many times they appear in the dataset
ids = schools_cleaned[schools_cleaned['DaysEnr'].isnull()]['ANON_ID'].unique()
schools_cleaned[schools_cleaned['ANON_ID'].isin(ids)]['ANON_ID'].value_counts()
```

[ ]:
```
ANON_ID
46788    3
```

```
74073    3
74195    3
45261    3
23826    3
76869    3
76885    3
67766    2
11215    2
17509    2
5645     2
37190    2
54615    2
31347    2
74203    2
54845    2
76930    1
44957    1
49492    1
49493    1
45784    1
58263    1
75819    1
60842    1
60070    1
51972    1
56537    1
73585    1
62638    1
Name: count, dtype: int64
```

[ ]:
```
# Look at one student
schools_cleaned[schools_cleaned['ANON_ID'] == 74195]
```

[ ]:

	ANON_ID	Birthdate	Gen	Eth	Fluency	SpEd	Grade	AttRate	\
2855	74195	2004-10-26	M	Latino	RFEP	Not Special Ed	10	NaN	
3737	74195	2004-10-26	M	Latino	RFEP	Not Special Ed	11	0.2333	
4621	74195	2004-10-26	M	Latino	RFEP	Not Special Ed	12	0.5056	

	DaysEnr	DaysAbs	Susp	CurrWeightedTotGPA	SED	School	Year	Age	\
2855	NaN	NaN	0.0	0.00	SED	Castlemont	19-20	15	
3737	180.0	138.0	0.0	0.00	SED	Castlemont	20-21	16	
4621	180.0	89.0	0.0	1.75	SED	Castlemont	21-22	17	

	ChroAbs
2855	0
3737	1
4621	1

```
[ ]: # Percentage of missing values with respect to school
     schools_cleaned.groupby('School')['AttRate'].apply(lambda x: x.isna().mean())
```

```
[ ]: School
     Castlemont            0.002169
     East Oakland PRIDE    0.001519
     Elmhurst United       0.000401
     MLK                   0.001103
     McClymonds            0.001949
     Prescott              0.000000
     WOMS                  0.000645
     Name: AttRate, dtype: float64
```

```
[ ]: # Drop the rows with those missing values of AttRate, DaysEnrolled, DaysAbs
     schools_cleaned.dropna(subset=['AttRate', 'DaysEnr', 'DaysAbs'], inplace=True)
     schools_cleaned.isna().sum()
```

```
[ ]: ANON_ID                 0
     Birthdate                0
     Gen                      0
     Eth                      0
     Fluency                  0
     SpEd                     0
     Grade                    0
     AttRate                  0
     DaysEnr                  0
     DaysAbs                  0
     Susp                     0
     CurrWeightedTotGPA    6434
     SED                      0
     School                   0
     Year                     0
     Age                      0
     ChroAbs                  0
     dtype: int64
```

2.0.3 CurrWeightedTotGPA:

All records for grades -1 through 5 have missing values in the GPA column, with a small percentage of missing values also present for grades 9, 10, and 11. Given the strong correlation between GPA and Absent Rate, as shown in the correlation heatmap in section 2, it's essential to retain this column in a way that minimizes noise. We cannot assign a numerical GPA to records in grades -1 to 5, as arbitrary values could misrepresent GPA meaningfully.

To address this, I propose creating a new variable, Academic Status, which categorizes GPA into four levels:

Good: GPA $\geq$ 3.0 Average: 2.0 $\leq$ GPA $<$ 3.0 At Risk: GPA $<$ 2.0 Unknown: GPA is NaN

After creating Academic Status, we can drop the GPA column and explore the relationship between this new variable and the response variable, ChroAbs.

[ ]:
```
#Percentage of missing GPA in each grade
schools.groupby('Grade')['CurrWeightedTotGPA'].apply(lambda x: x.isna().mean())
```

[ ]:
```
Grade
-1     1.000000
 0     1.000000
 1     1.000000
 2     1.000000
 3     1.000000
 4     1.000000
 5     1.000000
 6     0.000000
 7     0.000000
 8     0.000000
 9     0.001144
 10    0.001181
 11    0.000931
 12    0.000000
Name: CurrWeightedTotGPA, dtype: float64
```

[ ]:
```
# Define a function to categorize academic status based on the given criteria
def categorize_academic_status(gpa):
    if pd.isna(gpa):
        return "Unknown"
    elif gpa >= 3.0:
        return "Good"
    elif 2.0 <= gpa < 3.0:
        return "Average"
    else:
        return "AtRisk"

# Apply the function to create a new column 'AcademicStatus'
schools_cleaned['AcademicStatus'] = schools_cleaned['CurrWeightedTotGPA'].
 ↪apply(categorize_academic_status)
schools_cleaned.drop(columns=['CurrWeightedTotGPA'], inplace=True)
schools_cleaned.head()
```

[ ]:
```
   ANON_ID  Birthdate Gen                 Eth Fluency            SpEd  Grade  \
0      338 2003-07-21   F              Latino    RFEP  Not Special Ed      9
1      340 2003-03-31   M              Latino      EL  Not Special Ed      9
2      478 2003-09-06   F  Multiple Ethnicity      EO      Special Ed      9
3      686 2000-04-02   M              Latino      EL  Not Special Ed     12
4      693 2002-03-28   F              Latino    RFEP  Not Special Ed     10
```

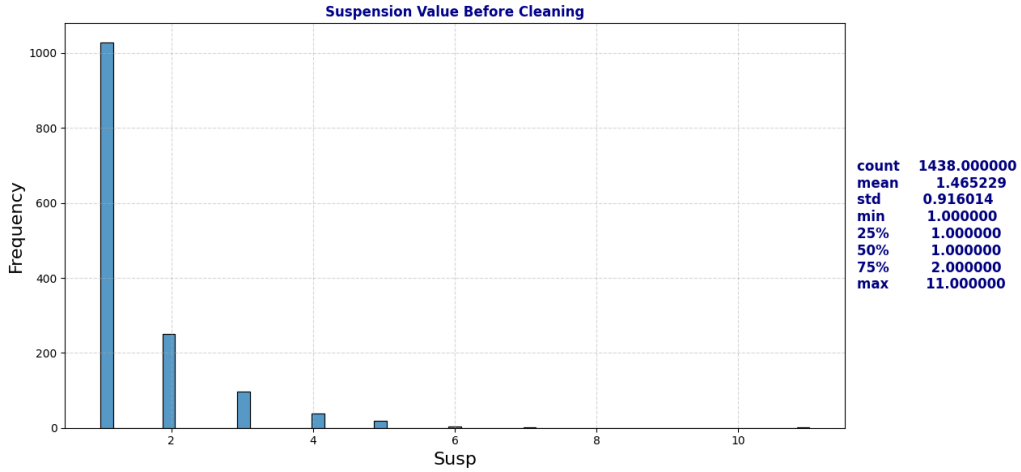
```
   AttRate  DaysEnr  DaysAbs  Susp      SED      School   Year  Age  ChroAbs  \
0   0.9889    180.0      2.0   0.0  Unknown  Castlemont  17-18   14        0
1   0.8389    180.0     29.0   2.0  Unknown  Castlemont  17-18   14        1
2   0.7263    179.0     49.0   1.0  Unknown  Castlemont  17-18   14        1
3   0.9611    180.0      7.0   0.0  Unknown  Castlemont  17-18   17        0
4   0.9889    180.0      2.0   0.0      SED  Castlemont  17-18   15        0

  AcademicStatus
0           Good
1         AtRisk
2         AtRisk
3        Average
4        Average
```

[ ]:
```
schools_cleaned.isna().sum()
```

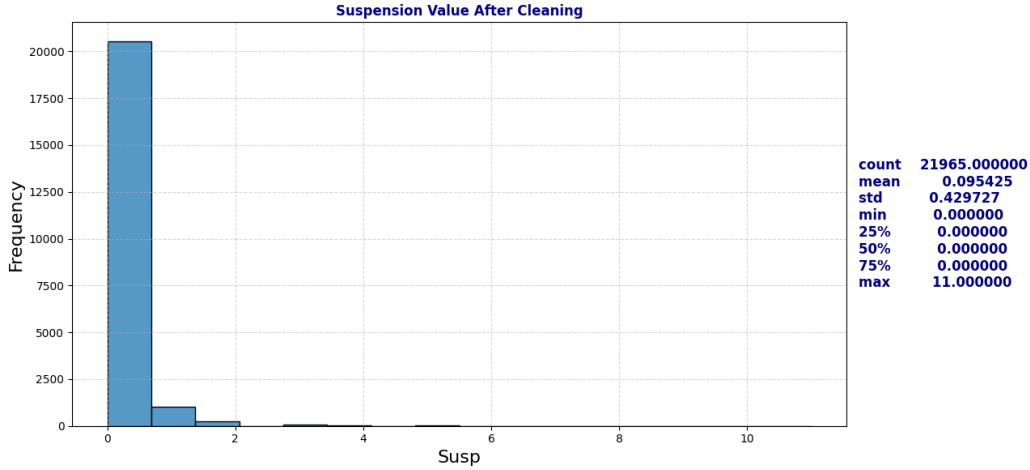
[ ]:
```
ANON_ID           0
Birthdate         0
Gen               0
Eth               0
Fluency           0
SpEd              0
Grade             0
AttRate           0
DaysEnr           0
DaysAbs           0
Susp              0
SED               0
School            0
Year              0
Age               0
ChroAbs           0
AcademicStatus    0
dtype: int64
```

2.0.4 Missing Values: Before and After

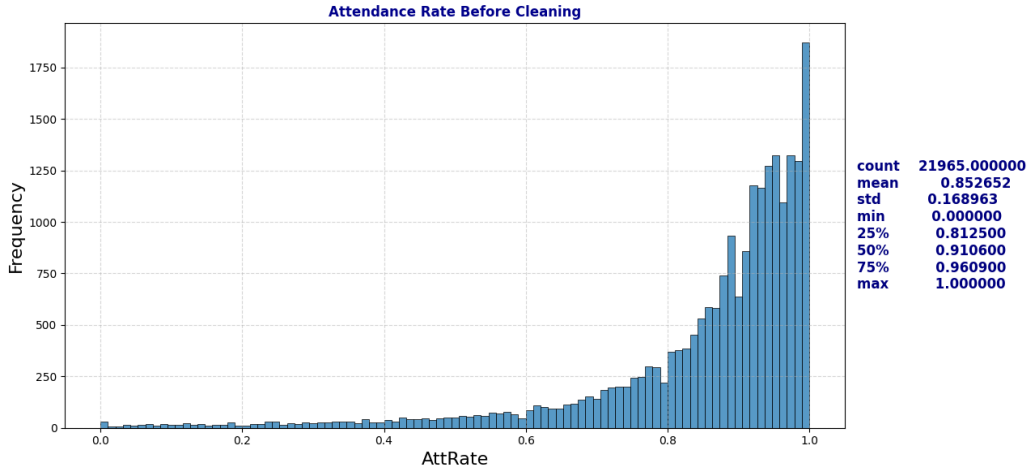
[506]:
```
# Suspension value before and after
plot_distribution_and_summary(schools, 'Susp', "Suspension Value Before␣
↪Cleaning")
```



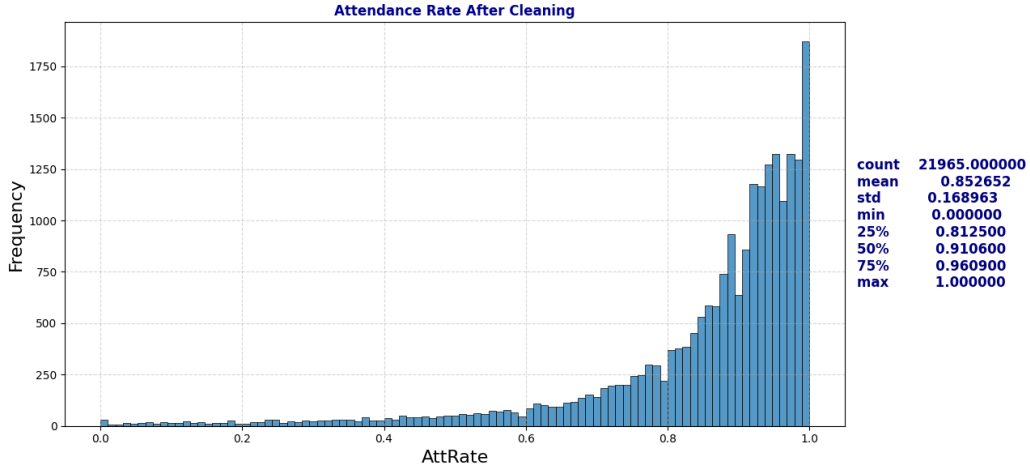
[507]:
```
plot_distribution_and_summary(schools_cleaned, 'Susp', "Suspension Value After␣
↪Cleaning")
```



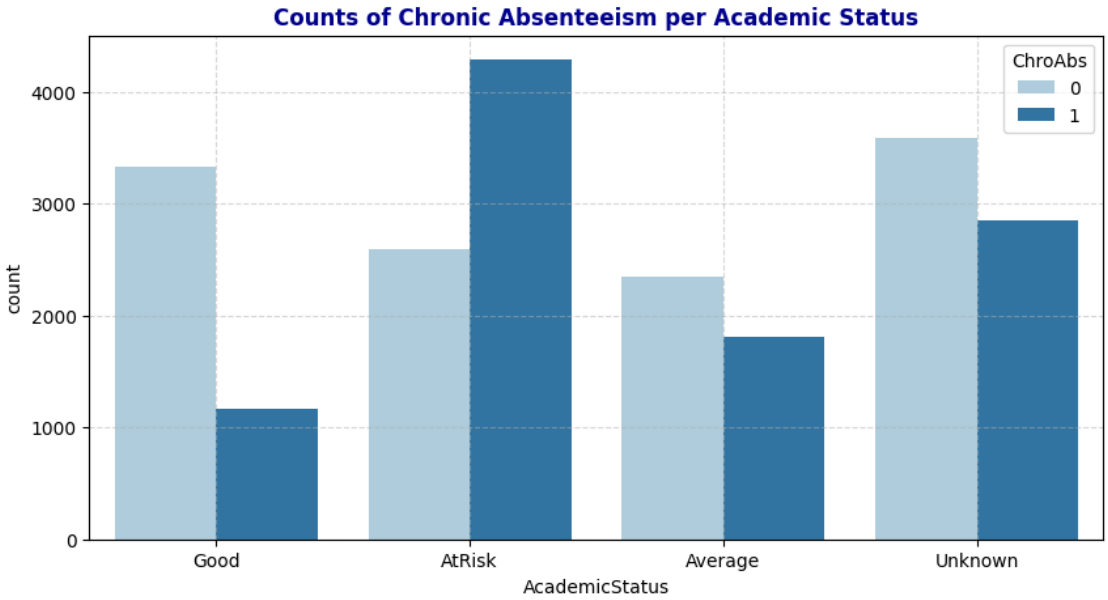
[508]:
```
# Attendance Rate Before Cleaning
plot_distribution_and_summary(schools, 'AttRate', "Attendance Rate Before␣
↪Cleaning")
```



[509]:
```
# Attendance Rate After Cleaning
plot_distribution_and_summary(schools_cleaned, 'AttRate', "Attendance Rate␣
↪After Cleaning")
```



[504]:
```
# Visualize Chronic Absenteeism by Academic Status
plt.figure(figsize=(10, 5))
plt.title('Counts of Chronic Absenteeism per Academic Status'␣
↪,fontweight='bold', color='darkblue')
sns.countplot(data=schools_cleaned, x='AcademicStatus', hue='ChroAbs',␣
↪palette='Paired')
plt.grid(True, linestyle='--', alpha=0.5);
```



2.1 3b. Handle Duplicates and Irrelevant Data

[511]:
```
# Frequency of values in each categorical column

# Generate HTML for each column's value counts
html_output = "<div style='display: flex; flex-wrap: wrap;'>"

cat_cols = ['Gen', 'Eth', 'Fluency', 'SpEd', 'Grade', 'SED', 'Age', 'ChroAbs']
for col in cat_cols:
    # Get the value counts for the column
    counts = schools[col].value_counts()

    # Create an HTML table for the value counts of each column
    html_output += f"""
    <div style='margin-right: 20px; padding: 10px; border: 1px solid #ddd;'>
        <h4>{col}</h4>
        {counts.to_frame().to_html(header=False)}
    </div>
    """

html_output += "</div>"

# Display all counts side by side in HTML format
display(HTML(html_output))
```

```
<IPython.core.display.HTML object>
```

2.1.1 Gender: has one unsual value of 'm'. Modify it to "M"

[ ]:
```
# Gender has one unsual value of 'm'. Modify it to "M"
schools['Gen'] = schools['Gen'].replace('m', 'M')
schools['Gen'].value_counts()
```

[ ]:
```
Gen
M    11857
F    10113
N       24
Name: count, dtype: int64
```

2.1.2 Fluency:

There are 100 records labeled as `TBD` and 6 as`Unknown` in the Fluency column. Upon examining records with `TBD`, we find that these are primarily new enrollees. For students who reappear in subsequent years, their Fluency values are recorded in later years. Based on this insight, I fill Unknown or `TBD` entries in `Fluency` with matching values from other records with the same `ANON_ID`. After this process, 68 `TBD` and 2 `Unknown` entries remain, representing students who appear only once in the dataset—likely indicating that they did not re-enroll. Even though these students account for only 3% of the dataset, they still provide valuable information in other variables, such as Attendance Rate, SED, SpEd, etc. Therefore, I've decided to retain these records rather than dropping them. However, to simplify our model, I will consolidate TBD and Unknown into a single label, Unknown, in the Fluency column. This will help reduce model complexity while preserving the information these records offer.

[290]:
```
# Look at the records that have Unknown in Fluency
schools[schools['Fluency'] == 'Unknown']
```

[290]:

	ANON_ID	Birthdate	Gen	Eth	Fluency	SpEd	Grade	\
4662	78297	2004-11-24	M	Latino	Unknown	Not Special Ed	10	
4663	78298	2004-11-24	M	Latino	Unknown	Not Special Ed	10	
5159	44057	2008-01-15	F	Latino	Unknown	Not Special Ed	9	
12481	5911	2009-10-17	F	Latino	Unknown	Not Special Ed	7	
15905	50958	2014-03-23	M	White	Unknown	Not Special Ed	1	
20077	31810	2017-02-25	M	Not Reported	Unknown	Not Special Ed	-1	

	AttRate	DaysEnr	DaysAbs	Susp	CurrWeightedTotGPA	SED	\
4662	0.9892	93.0	1.0	NaN	3.63	Not SED	
4663	0.9892	93.0	1.0	NaN	3.63	Not SED	
5159	0.5122	41.0	20.0	NaN	0.00	Unknown	
12481	0.7385	65.0	17.0	NaN	3.00	Unknown	
15905	0.9063	64.0	6.0	NaN	NaN	Not SED	
20077	0.0833	12.0	11.0	NaN	NaN	SED	

	School	Year	Age	ChroAbs
4662	Castlemont	21-22	17	0
4663	Castlemont	21-22	17	0

```
5159        Castlemont  22-23   14        1
12481  Elmhurst United  22-23   13        1
15905              MLK  21-22    7        0
20077         Prescott  21-22    4        1
```

[291]:
```
# Examine one student
schools[schools['ANON_ID'] == 78297]
```

[291]:
```
      ANON_ID  Birthdate Gen     Eth  Fluency            SpEd  Grade  AttRate  \
4662    78297 2004-11-24   M  Latino  Unknown  Not Special Ed     10   0.9892
5571    78297 2004-11-24   M  Latino       EL  Not Special Ed     11   0.9056
6442    78297 2004-11-24   M  Latino       EL  Not Special Ed     12   0.8833

      DaysEnr  DaysAbs  Susp  CurrWeightedTotGPA      SED      School   Year  \
4662     93.0      1.0   NaN                3.63  Not SED  Castlemont  21-22
5571    180.0     17.0   NaN                3.43      SED  Castlemont  22-23
6442    180.0     21.0   NaN                2.40      SED  Castlemont  23-24

      Age  ChroAbs
4662   17        0
5571   18        0
6442   19        1
```

[292]:
```
# Function to fill 'Unknown' or 'TBD' in 'Fluency' with matching 'ANON_ID' values
def fill_fluency(row):
    if row['Fluency'] in ['Unknown', 'TBD']:
        # Find the first non-null value of 'Fluency' for the same 'ANON_ID'
        matching_value = schools_cleaned[(schools_cleaned['ANON_ID'] == row['ANON_ID']) & (schools_cleaned['Fluency'].notna()) & (~schools_cleaned['Fluency'].isin(['Unknown', 'TBD']))]

        # If there is a match, return its 'Fluency' value; otherwise, return the original 'Fluency'
        if not matching_value.empty:
            return matching_value.iloc[0]['Fluency']
    return row['Fluency']

# Apply the function to the 'Fluency' column
schools_cleaned['Fluency'] = schools_cleaned.apply(fill_fluency, axis=1)
```

[293]:
```
schools_cleaned['Fluency'].value_counts()
```

[293]:
```
Fluency
EO         10638
EL          8308
RFEP        2683
```

```
IFEP          266
TBD            68
Unknown         2
Name: count, dtype: int64
```

[514]:
```
print('Percentage of TBD and Unknown left in the dataset: ', 70/schools_cleaned.
↪shape[0])
```

```
Percentage of TBD and Unknown left in the dataset:  0.0031868882312770315
```

[294]:
```
# Examine the students left with TBD values in Fluency
schools_cleaned[schools_cleaned['Fluency'] == 'TBD']
```

[294]:

	ANON_ID	Birthdate	Gen	Eth	Fluency	SpEd	Grade	\
8	870	2000-12-23	M	Latino	TBD	Not Special Ed	9	
52	3933	2002-12-27	F	Latino	TBD	Not Special Ed	9	
53	3936	2002-11-16	M	Latino	TBD	Not Special Ed	9	
201	14335	2000-03-04	M	Latino	TBD	Not Special Ed	9	
251	17703	2001-01-22	M	Latino	TBD	Not Special Ed	9	
…	…	…	..	…	…	…	…	
20067	26375	2016-11-21	F	Latino	TBD	Not Special Ed	-1	
20068	26376	2016-12-21	F	Latino	TBD	Not Special Ed	-1	
20548	35013	2006-06-15	M	Not Reported	TBD	Not Special Ed	6	
20549	35015	2003-11-02	F	Not Reported	TBD	Not Special Ed	7	
20904	1962	2007-11-02	M	Asian	TBD	Not Special Ed	6	

	AttRate	DaysEnr	DaysAbs	Susp	CurrWeightedTotGPA	SED	\
8	0.8761	113.0	14.0	0.0	1.82	Unknown	
52	0.9344	122.0	8.0	0.0	2.94	Unknown	
53	0.9426	122.0	7.0	0.0	1.76	Unknown	
201	0.6889	45.0	14.0	0.0	0.00	Unknown	
251	1.0000	9.0	0.0	0.0	0.00	Unknown	
…	…	…	…	…	…	…	
20067	0.8788	33.0	4.0	0.0	NaN	Not SED	
20068	0.8788	33.0	4.0	0.0	NaN	Not SED	
20548	0.6667	6.0	2.0	0.0	0.00	Unknown	
20549	0.6667	6.0	2.0	0.0	0.00	Unknown	
20904	1.0000	5.0	0.0	0.0	0.00	Not SED	

	School	Year	Age	ChroAbs
8	Castlemont	17-18	17	1
52	Castlemont	17-18	15	0
53	Castlemont	17-18	15	0
201	Castlemont	17-18	17	1
251	Castlemont	17-18	16	0
…	…	…	…	
20067	Prescott	21-22	5	1

```
20068    Prescott  21-22    5        1
20548        WOMS  17-18   11        1
20549        WOMS  17-18   14        1
20904        WOMS  19-20   12        0

[68 rows x 17 columns]
```

[295]:
```
# Replace all 'TBD' left as 'Unknown' in the dataset
schools_cleaned['Fluency'] = schools_cleaned['Fluency'].replace('TBD',␣
 ↪'Unknown')
schools_cleaned['Fluency'].value_counts()
```

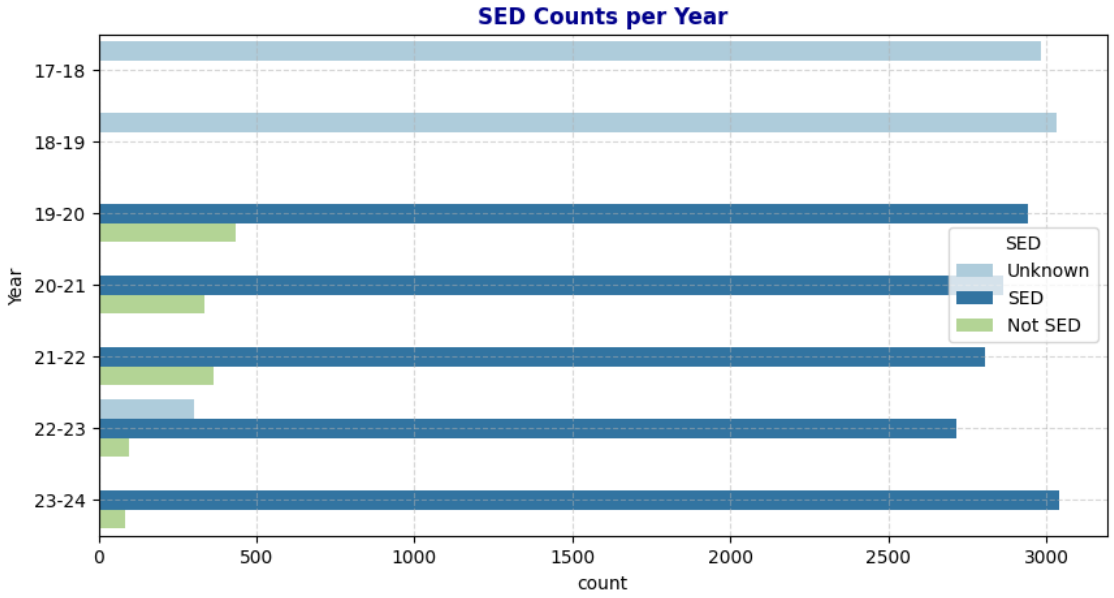
[295]:
```
Fluency
EO         10638
EL          8308
RFEP        2683
IFEP         266
Unknown       70
Name: count, dtype: int64
```

2.1.3 SED

From the Theil's U correlation heatmap in section 2, I observed a strong predictive relationship between Year and SED. In this section, I'll explore the reason behind this relationship. By plotting the distribution of SED values across different years, I found that in the two school years 2017-18 and 2018-19, SED values were consistently labeled as Unknown. This finding aligns with the observed relationship between Year and SED in the heatmap.

Upon further investigation, examining students with Unknown values in SED revealed a pattern: these students often have this column filled with meaningful information in the following school years. Following this pattern, I applied the same process to fill in Unknown values for SED where possible. Although some Unknown values remain, we can retain these records as they still contribute valuable information to other variables.

[522]:
```
# Plot SED counts per Year
plt.figure(figsize=(10, 5))
sns.countplot(data=schools, y='Year', hue='SED', palette='Paired')
plt.title('SED Counts per Year', fontweight='bold', color='darkblue')
plt.grid(True, linestyle='--', alpha=0.5)
```

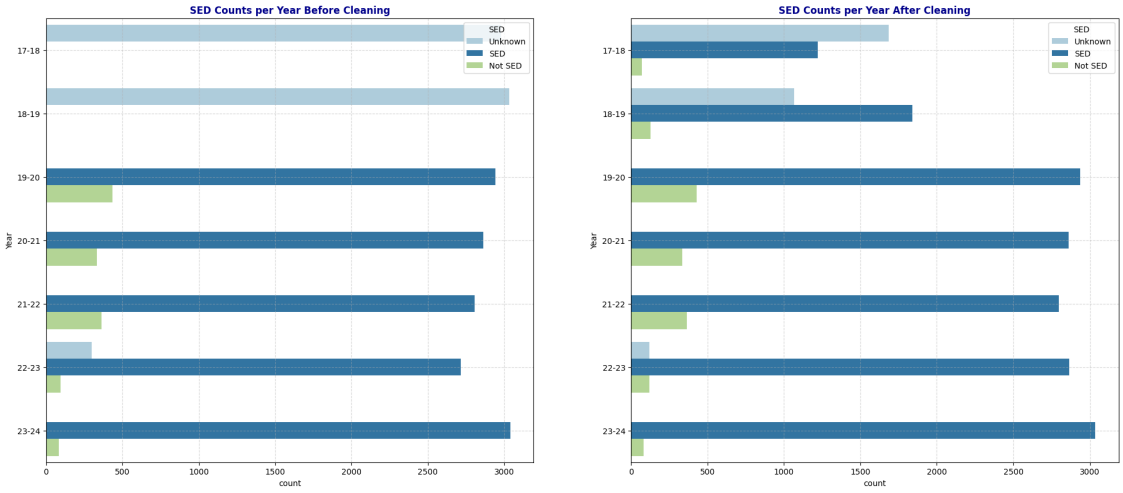


[297]:
```
# Function to fill 'Unknown' 'SED' with matching 'ANON_ID' values
def fill_SED(row):
    if row['SED'] == 'Unknown':
        # Find the first non-null value of 'SED' for the same 'ANON_ID'
        matching_value = schools_cleaned[(schools_cleaned['ANON_ID'] ==␣
 ↪row['ANON_ID']) & (schools_cleaned['SED'].notna()) &␣
 ↪(~schools_cleaned['SED'].isin(['Unknown']))]

        # If there is a match, return its 'SED' value; otherwise, return the␣
 ↪original 'SED'
        if not matching_value.empty:
            return matching_value.iloc[0]['SED']
    return row['SED']

# Apply the function to the 'SED' column
schools_cleaned['SED'] = schools_cleaned.apply(fill_SED, axis=1)
```

2.1.4 Duplicates and Irrelevant: Before and After

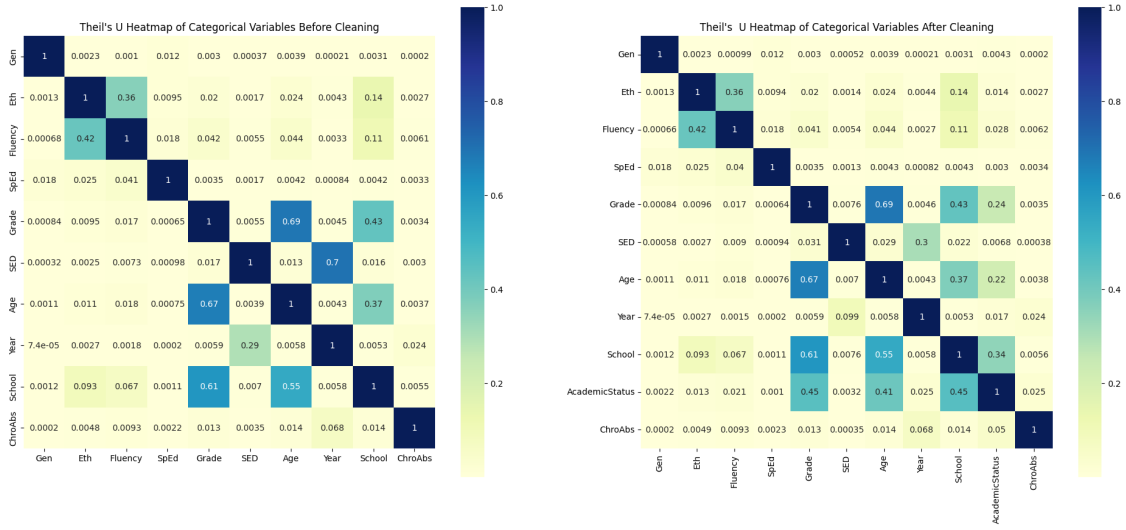
[526]:
```
#SED: Before and after
fig, axs = plt.subplots(
    nrows=1, ncols=2,
    figsize=(24, 10)
)
sns.countplot(data=schools, y='Year', hue='SED', palette='Paired', ax=axs[0])
axs[0].set_title('SED Counts per Year Before Cleaning', fontweight='bold',␣
 ↪color='darkblue')
```

```
axs[0].grid(True, linestyle='--', alpha=0.5)
sns.countplot(data=schools_cleaned, y='Year', hue='SED', palette='Paired',␣
 ↪ax=axs[1])
axs[1].set_title('SED Counts per Year After Cleaning', fontweight='bold',␣
 ↪color='darkblue')
axs[1].grid(True, linestyle='--', alpha=0.5)
plt.show()
```



[527]:
```
# Theil's U Heat Map before and After
cat_cols_cleaned = ['Gen', 'Eth', 'Fluency', 'SpEd', 'Grade', 'SED', 'Age',␣
 ↪'Year', 'School','AcademicStatus', 'ChroAbs']
# Create an empty DataFrame to store Theil's U values
theils_u_matrix_cleaned = pd.DataFrame(index=cat_cols_cleaned,␣
 ↪columns=cat_cols_cleaned)

# Fill the matrix with Theil's U values
for col1 in cat_cols_cleaned:
    for col2 in cat_cols_cleaned:
        if col1 == col2:
            theils_u_matrix_cleaned.loc[col1, col2] = 1  # Perfect association␣
 ↪with itself
        else:
            theils_u_matrix_cleaned.loc[col1, col2] =␣
 ↪theils_u(schools_cleaned[col1], schools_cleaned[col2])

# Convert to float for heatmap compatibility
theils_u_matrix_cleaned = theils_u_matrix_cleaned.astype(float)

fig, axs = plt.subplots(
    nrows=1, ncols=2,
```

```
    figsize=(24, 10)
)

# Plot the heatmap

sns.heatmap(theils_u_matrix_cleaned, annot=True, cmap="YlGnBu", square=True,␣
 ↪ax=axs[1])
axs[1].set_title("Theil's  U Heatmap of Categorical Variables After Cleaning")
sns.heatmap(theils_u_matrix, annot=True, cmap="YlGnBu", square=True, ax=axs[0])
axs[0].set_title("Theil's U Heatmap of Categorical Variables Before Cleaning")
plt.show()
```



Observation: After modifying the Fluency and SED columns, I observed that Fluency retains its predictive power on Chronic Absenteeism, while SED shows a significant decrease in predictive strength (dropping from 0.0035 to 0.00035). One possible reason is that replacing Unknown values in SED diluted the relationship between Year and SED—since school years 2017-18 and 2018-19 no longer consist exclusively of Unknown values, SED no longer inherits the predictive power of Year on Chronic Absenteeism (where Year has the strongest predictive power).

Additionally, after transforming GPA into the categorical variable Academic Status, the heatmap now includes this new variable. Academic Status shows a strong predictive power on Chronic Absenteeism, with a correlation of 0.05, second only to Year. This aligns with the Pearson correlation heatmap observation where GPA and Attendance Rate had a strong correlation. Furthermore, Academic Status exhibits moderate predictive power on School, likely because elementary schools (covering grades -1 to 5) only have Unknown values in Academic Status.

3 4. Preprocessing for Machine Learning

Note on Selecting Variables for Our Machine Learning Model

- Exclude Personal Identifiers: Omit ANON_ID and Birthdate as these are personal identifiers that do not contribute to predicting Chronic Absenteeism.
- Exclude Redundant Attendance Metrics: Exclude AttRate, DaysEnr, and DaysAbs due to their strong dependency on one another, with Chronic Absenteeism (response variable ChroAbs) already capturing this information effectively.
- Encoding for Categorical Variables: Use one-hot encoding for all categorical variables, as they are nominal (non-ordinal). Label encoding would introduce artificial order, which is unnecessary and could mislead the model. Additionally, we'll select a subset of categorical variables with high predictive power and a manageable number of unique categories to reduce model complexity.
- Exclude Weak Predictors: Exclude Gender and SED due to their minimal predictive power on Chronic Absenteeism.
- Address Potential Bias: Exclude Ethnicity to avoid potential racial bias. Retain Fluency, as it has a stronger predictive relationship with Chronic Absenteeism and moderate predictive power on Ethnicity.
- Simplify by Reducing Redundancy: Exclude Grade and Age as they have numerous unique values. Instead, retain School, which is strongly associated with both and provides similar information with fewer categories.
- Final Variable Selection: **AcademicStatus, Year, School, Fluency, and Special Ed** as they each exhibit strong predictive power for Chronic Absenteeism and will support a more interpretable model.

3.1 4a. Encoding Categorical Variables

[530]:
```
schools_cleaned.shape
```

[530]:
```
(21965, 17)
```

[529]:
```
# Categorical used for predictive model
cat_pred_cols = ['Fluency','SpEd', 'Year', 'School', 'AcademicStatus']
df_transformed = pd.get_dummies(schools_cleaned[cat_pred_cols], drop_first=True)
df_transformed.shape
```

[529]:
```
(21965, 20)
```

3.2 4b. Scaling and Normalizing Numerical Data

I don't use any numerical variables for the model; therefore, this section is skipped.

3.3 4c. Data Splitting

[351]:
```
X_col = ['Fluency','SpEd','SED', 'Grade', 'SED', 'Year', 'School',␣
  ↪'AcademicStatus']
y_col = 'ChroAbs'
```

[352]:
```
from sklearn.model_selection import train_test_split

X = df_transformed
y = schools_cleaned[y_col]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[353]:
```
df_train = pd.concat([X_train, y_train], axis=1)
df_test = pd.concat([X_test, y_test], axis=1)
```

[ ]:
```
%cd /content/drive/MyDrive/ONGB -Lan
!pip install nbconvert
!jupyter nbconvert --execute --to html "DataExploration.ipynb"
```

```
/content/drive/MyDrive/ONGB -Lan
Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-
packages (6.5.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (3.0.2)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (0.10.0)
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (24.1)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert) (2.18.0)
```

```
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (1.4.0)
Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dist-
packages (from nbconvert) (5.7.1)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.10/dist-packages (from jupyter-core>=4.7->nbconvert)
(4.3.6)
Requirement already satisfied: jupyter-client>=6.1.12 in
/usr/local/lib/python3.10/dist-packages (from nbclient>=0.5.0->nbconvert)
(6.1.12)
Requirement already satisfied: fastjsonschema>=2.15 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nbconvert) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nbconvert) (4.23.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->nbconvert) (2.6)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.20.0)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (24.0.1)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.10/dist-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.3.3)
[NbConvertApp] Converting notebook DataExploration.ipynb to html
```