

Exploratory Analysis of NYC School's Math Results.

In [873...]

```
import pandas as pd
import numpy as np
from scipy.stats import zscore
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline
```

Uploading all the required files that need to be explored:

- it is the important and first step to clean the data exhaustively in order to get the meaningful information & insights from it.
- Dataset is about the NYC School's Math Test Scores.

Objectives of EDA is following below:

- Uncover the structure and content of the dataset.
- Finding key variables from it.
- Detecting outliers and anomalies.
- Getting maximum insight from it.
- discovering pattern and trends.

First Step: Uploading the datasets.

In [105...]

```
df = pd.read_excel('D:\\Data\\Profile Project\\NYC Math Test Results.xlsx', sheet_name='All_district_student')
df1 = pd.read_excel('D:\\Data\\Profile Project\\NYC Math Test Results.xlsx', sheet_name='2006_-_2012_ethnicity-dist.')
df2 = pd.read_excel('D:\\Data\\Profile Project\\NYC Math Test Results.xlsx', sheet_name='Ethnicity_Disttict')
df3 = pd.read_excel('D:\\Data\\Profile Project\\NYC Math Test Results.xlsx', sheet_name='2006-2012_Dist_allStud_math_res')
```

Second Step: Preparing and Cleaning the datasets for analysis:

- Handling missing data: df0.isnull().sum(), df0.dropna().
- Fixing Structural errors.
- Removing duplicate and irrelevant observations: df0.duplicated(), df0.drop_duplicates().
- Transforming the data.

These datasets are pre-cleaned, so some of the processes of data cleaning will not be performed.

Dataset Number One

In [105...]

```
cols = ['District', 'Grade', 'Year', 'Demographic', 'Number_Tested', 'Mean_Scale_Score', 'Proficient_Level_1', 'Level_1_%', 'Proficient_Level_2', 'Level_2_%', 'Proficient_Level_3', 'Level_3_%', 'Profic
df.columns=cols
df3.columns=cols
df0 = pd.concat([df,df3], axis=0, ignore_index=True)
df0 = df0.round(decimals=2)
```

In [105...]

```
df0 = df0.drop(df0.index[df0['Grade']=='All Grades'])
df0 = df0.reset_index().drop(['index'], axis=1)
```

```
df0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2688 entries, 0 to 2687
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   District         2688 non-null    int64  
 1   Grade            2688 non-null    object  
 2   Year             2688 non-null    int64  
 3   Dempgraphic     2688 non-null    object  
 4   Number_Tested   2688 non-null    int64  
 5   Mean_Scale_Score 2688 non-null    float64 
 6   Proficient_Level_1 2688 non-null    int64  
 7   Level_1_%        2688 non-null    float64 
 8   Proficient_Level_2 2688 non-null    int64  
 9   Level_2_%        2688 non-null    float64 
 10  Proficient_Level_3 2688 non-null    int64  
 11  Level_3_%        2688 non-null    float64 
 12  Proficient_Level_4 2688 non-null    int64  
 13  Level_4_%        2688 non-null    float64 
 14  Proficient_Level_3+level_4 2688 non-null    int64  
 15  Level-3+4%       2688 non-null    float64 
dtypes: float64(6), int64(8), object(2)
memory usage: 336.1+ KB
```

In [105...]

```
df0
```

Out[105...]

	District	Grade	Year	Dempgraphic	Number_Tested	Mean_Scale_Score	Proficient_Level_1	Level_1_%	Proficient_Level_2	Level_2_%	Proficient_Level_3	Level_3_%	Proficient_Level_4	Level_4_%	Proficient_Level_3+level_4	Level-3+4%
0	1	3	2013	All Students	887	306.69	249	28.07	266	29.99	179	20.18	193	21.76	372	41.94
1	1	3	2014	All Students	845	307.95	225	26.63	223	26.39	204	24.14	193	22.84	397	46.98
2	1	3	2015	All Students	765	309.46	221	28.89	175	22.88	168	21.96	201	26.27	369	48.24
3	1	3	2016	All Students	743	313.55	184	24.76	178	23.96	141	18.98	240	32.30	381	51.28
4	1	3	2017	All Students	726	311.88	185	25.48	166	22.87	152	20.94	223	30.72	375	51.65
...	
2683	32	8	2008	All Students	1517	650.00	202	13.30	528	34.80	686	45.20	101	6.70	787	51.90
2684	32	8	2009	All Students	1590	662.00	106	6.70	406	25.50	923	58.10	155	9.70	1078	67.80
2685	32	8	2010	All Students	1600	662.00	314	19.60	698	43.60	462	28.90	126	7.90	588	36.80
2686	32	8	2011	All Students	1541	664.00	215	14.00	708	45.90	498	32.30	120	7.80	618	40.10
2687	32	8	2012	All Students	1295	666.00	184	14.20	564	43.60	428	33.10	119	9.20	547	42.20

2688 rows × 16 columns

Third Step: Filtering and removing outliers.

There are several techniques for detecting outliers such as:

1. Sorting the data points.
2. Data Visualization.

3. Statistical methods.

Interquartile Range method will be used in these datasets.

The Interquartile range method is used in this dataset.

```
In [106...]: df_outlier = df0.astype({'District':'str', 'Year':'int', 'Grade':'str', 'Dempgraphic':'str'})  
category = df_outlier.iloc[:, 0:4]  
numeric=df_outlier.iloc[:,4:17]  
Q1 = numeric.quantile(0.25)  
Q3 = numeric.quantile(0.75)  
IQR = Q3-Q1  
print(IQR)
```

```
Number_Tested      1520.0000  
Mean_Scale_Score   366.3300  
Proficient_Level_1 462.0000  
Level_1_%          28.8950  
Proficient_Level_2 485.2500  
Level_2_%          12.1400  
Proficient_Level_3 645.5000  
Level_3_%          22.9725  
Proficient_Level_4 522.2500  
Level_4_%          18.2825  
Proficient_Level_3+level_4 1195.2500  
Level-3+4%         36.2000  
dtype: float64
```

```
In [106...]: # All the outliers in the dataset:  
outliers = numeric[((numeric<(Q1-1.5*IQR)) | (numeric>(Q3+1.5*IQR)))]  
outliers.count()
```

```
Out[106...]: Number_Tested      0  
Mean_Scale_Score     0  
Proficient_Level_1   107  
Level_1_%            0  
Proficient_Level_2   39  
Level_2_%            23  
Proficient_Level_3   73  
Level_3_%            0  
Proficient_Level_4   33  
Level_4_%            33  
Proficient_Level_3+level_4 18  
Level-3+4%           0  
dtype: int64
```

```
In [106...]: print(outliers.max())  
print(outliers.min())
```

```
Number_Tested      NaN  
Mean_Scale_Score   NaN  
Proficient_Level_1 2298.0  
Level_1_%          NaN  
Proficient_Level_2 1885.0  
Level_2_%          53.5  
Proficient_Level_3 2913.0  
Level_3_%          NaN
```

```
Proficient_Level_4      1842.0
Level_4_%                72.0
Proficient_Level_3+level_4 4207.0
Level-3+4%               NaN
dtype: float64
Number_Tested            NaN
Mean_Scale_Score          NaN
Proficient_Level_1        1318.00
Level_1_%                 NaN
Proficient_Level_2        1549.00
Level_2_%                 0.60
Proficient_Level_3        1911.00
Level_3_%                 NaN
Proficient_Level_4        1425.00
Level_4_%                 54.04
Proficient_Level_3+level_4 3441.00
Level-3+4%               NaN
dtype: float64
```

```
In [106...]: # Removing outliers from dataset
outlier = numeric[~((numeric < (Q1-1.5*IQR)) | (numeric > (Q3+1.5*IQR)))]
outlier.count()
```

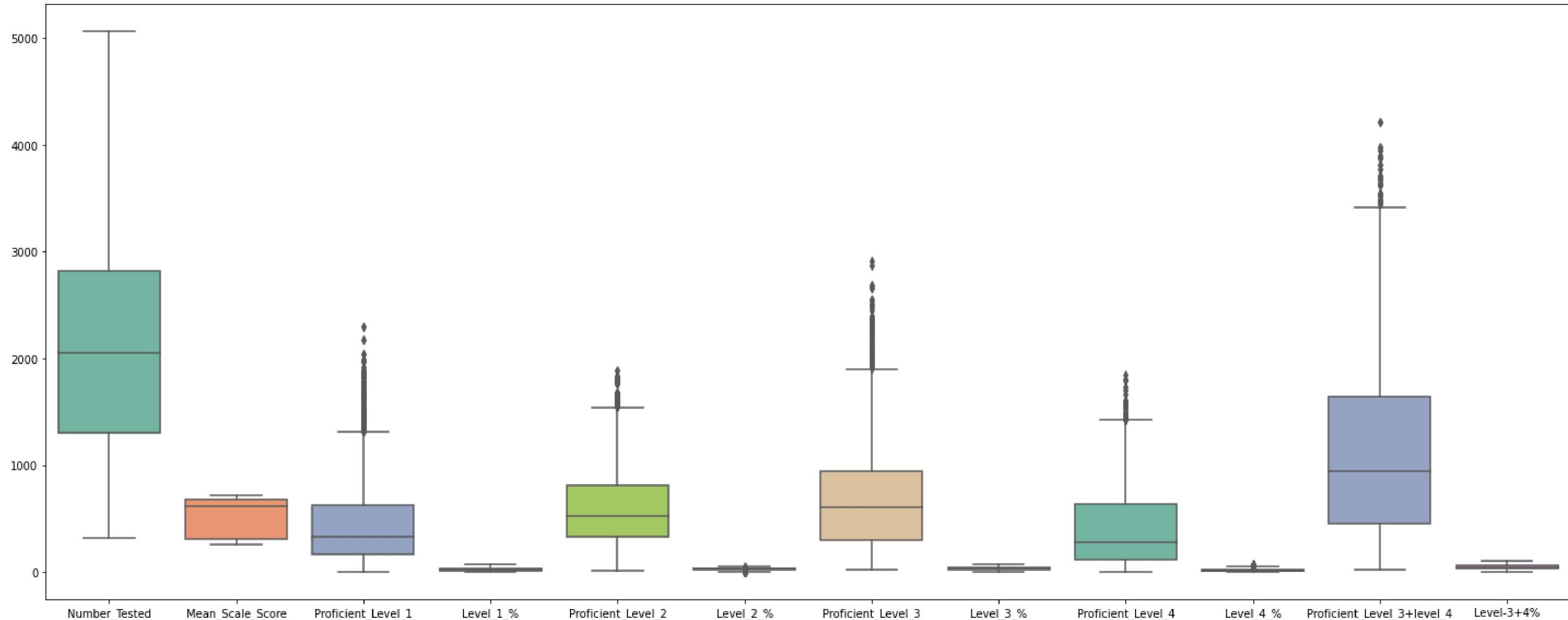
```
Out[106...]: Number_Tested      2688
Mean_Scale_Score          2688
Proficient_Level_1        2581
Level_1_%                 2688
Proficient_Level_2        2649
Level_2_%                 2665
Proficient_Level_3        2615
Level_3_%                 2688
Proficient_Level_4        2655
Level_4_%                 2655
Proficient_Level_3+level_4 2670
Level-3+4%               2688
dtype: int64
```

```
In [106...]: print(outlier.max())
print(outlier.min())
```

```
Number_Tested      5061.00
Mean_Scale_Score   723.00
Proficient_Level_1 1317.00
Level_1_%          71.67
Proficient_Level_2 1538.00
Level_2_%          53.10
Proficient_Level_3 1900.00
Level_3_%          72.70
Proficient_Level_4 1422.00
Level_4_%          53.53
Proficient_Level_3+level_4 3413.00
Level-3+4%         99.30
dtype: float64
Number_Tested      322.00
Mean_Scale_Score   262.20
Proficient_Level_1  1.00
Level_1_%          0.10
Proficient_Level_2  11.00
Level_2_%          4.60
```

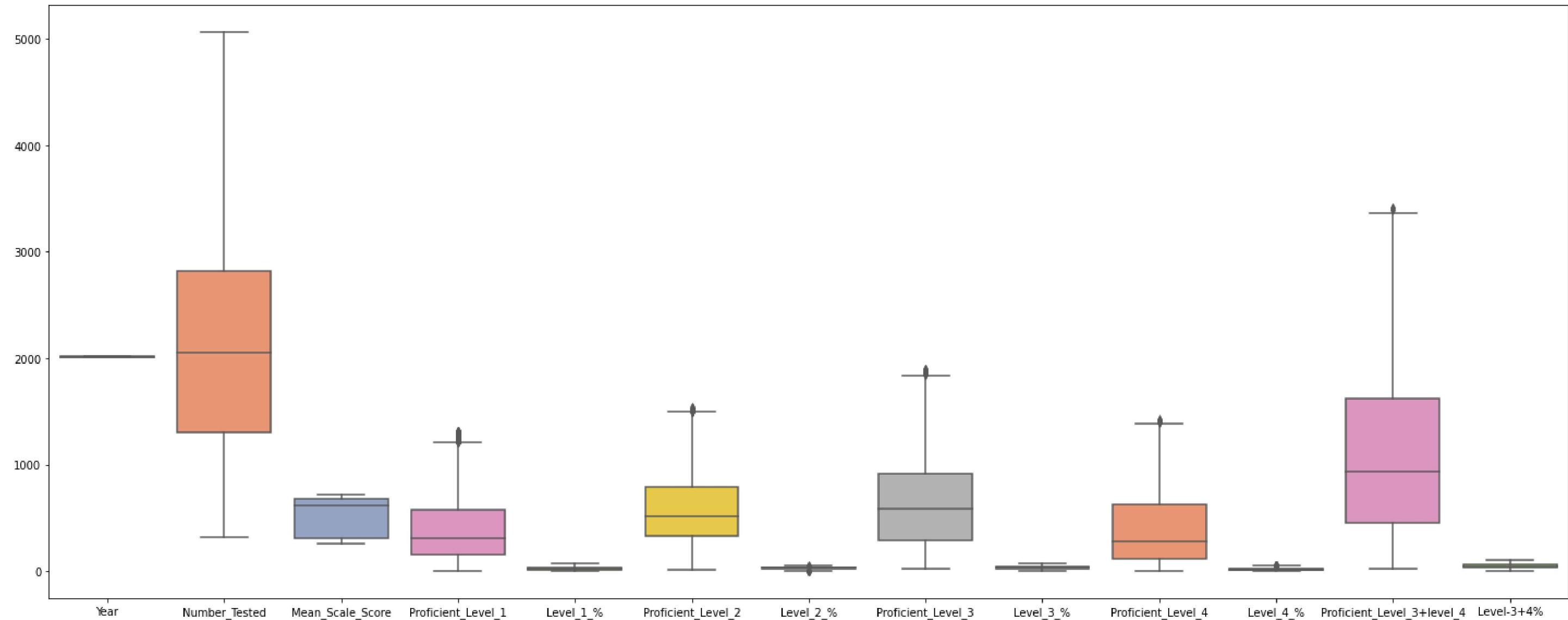
```
Proficient_Level_3      19.00
Level_3_%                3.46
Proficient_Level_4      1.00
Level_4_%                0.16
Proficient_Level_3+level_4 24.00
Level-3+4%              3.76
dtype: float64
```

```
In [106]: plt.figure(figsize=(25,10))
sns.boxplot(data=numeric, palette="Set2");
```



```
In [106]: outlier_removed_data = pd.concat([category, outlier], axis=1)
```

```
In [106]: plt.figure(figsize=(25,10))
sns.boxplot(data=outlier_removed_data, palette="Set2");
```



Dataset Number Two

Similar procedures have been performed on this dataset as in the previous dataset.

```
In [106]: df1.columns = cols
df2.columns=cols
df7 = pd.concat([df1, df2], axis=0, ignore_index=True)
```

```
In [106]: df7 = df7.drop(df7.index[df7['Grade']=='All Grades'])
df7 = df7.reset_index().drop(['index'], axis=1)
df7 = df7.replace('s',0)
df7 = df7.round(decimals=2)
```

```
In [107]: df7 = df7.drop(df7.index[df7['Mean_Scale_Score']==0])
df7.reset_index()
df7.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10608 entries, 0 to 10751
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype 

```

```
---  
0 District          10608 non-null int64  
1 Grade             10608 non-null int64  
2 Year              10608 non-null int64  
3 Demographic       10608 non-null object  
4 Number_Tested     10608 non-null int64  
5 Mean_Scale_Score  10608 non-null float64  
6 Proficient_Level_1 10608 non-null int64  
7 Level_1_%         10608 non-null float64  
8 Proficient_Level_2 10608 non-null int64  
9 Level_2_%         10608 non-null float64  
10 Proficient_Level_3 10608 non-null int64  
11 Level_3_%         10608 non-null float64  
12 Proficient_Level_4 10608 non-null int64  
13 Level_4_%         10608 non-null float64  
14 Proficient_Level_3+level_4 10608 non-null int64  
15 Level-3+4%        10608 non-null float64  
dtypes: float64(6), int64(9), object(1)  
memory usage: 1.4+ MB
```

In [107...]

df7

Out[107...]

	District	Grade	Year	Demographic	Number_Tested	Mean_Scale_Score	Proficient_Level_1	Level_1_%	Proficient_Level_2	Level_2_%	Proficient_Level_3	Level_3_%	Proficient_Level_4	Level_4_%	Proficient_Level_3+level_4	Level-3+4%
0	1	3	2006	Asian	155	704.00	4	2.60	4	2.60	68	43.90	79	51.00	147	94.80
1	1	3	2006	Black	185	659.00	21	11.40	47	25.40	104	56.20	13	7.00	117	63.20
2	1	3	2006	Hispanic	494	664.00	46	9.30	111	22.50	264	53.40	73	14.80	337	68.20
3	1	3	2006	White	94	702.00	1	1.10	5	5.30	45	47.90	43	45.70	88	93.60
4	1	3	2007	Asian	131	703.00	2	1.50	4	3.10	58	44.30	67	51.10	125	95.40
...
10747	32	8	2015	White	30	285.57	14	46.67	13	43.33	2	6.67	1	3.33	3	10.00
10748	32	8	2016	White	12	286.75	6	50.00	4	33.33	2	16.67	0	0.00	2	16.67
10749	32	8	2017	White	13	279.15	7	53.85	4	30.77	2	15.38	0	0.00	2	15.38
10750	32	8	2018	White	17	603.35	5	29.41	5	29.41	5	29.41	2	11.76	7	41.18
10751	32	8	2019	White	13	591.92	5	38.46	5	38.46	2	15.38	1	7.69	3	23.08

10608 rows × 16 columns

In [107...]

```
categorical = df7.iloc[:, 0:4]  
numerical = df7.iloc[:, 4:17]
```

In [107...]

```
numerical.plot(kind="box", subplots=True, layout=(4,3), figsize=(15,10))
```

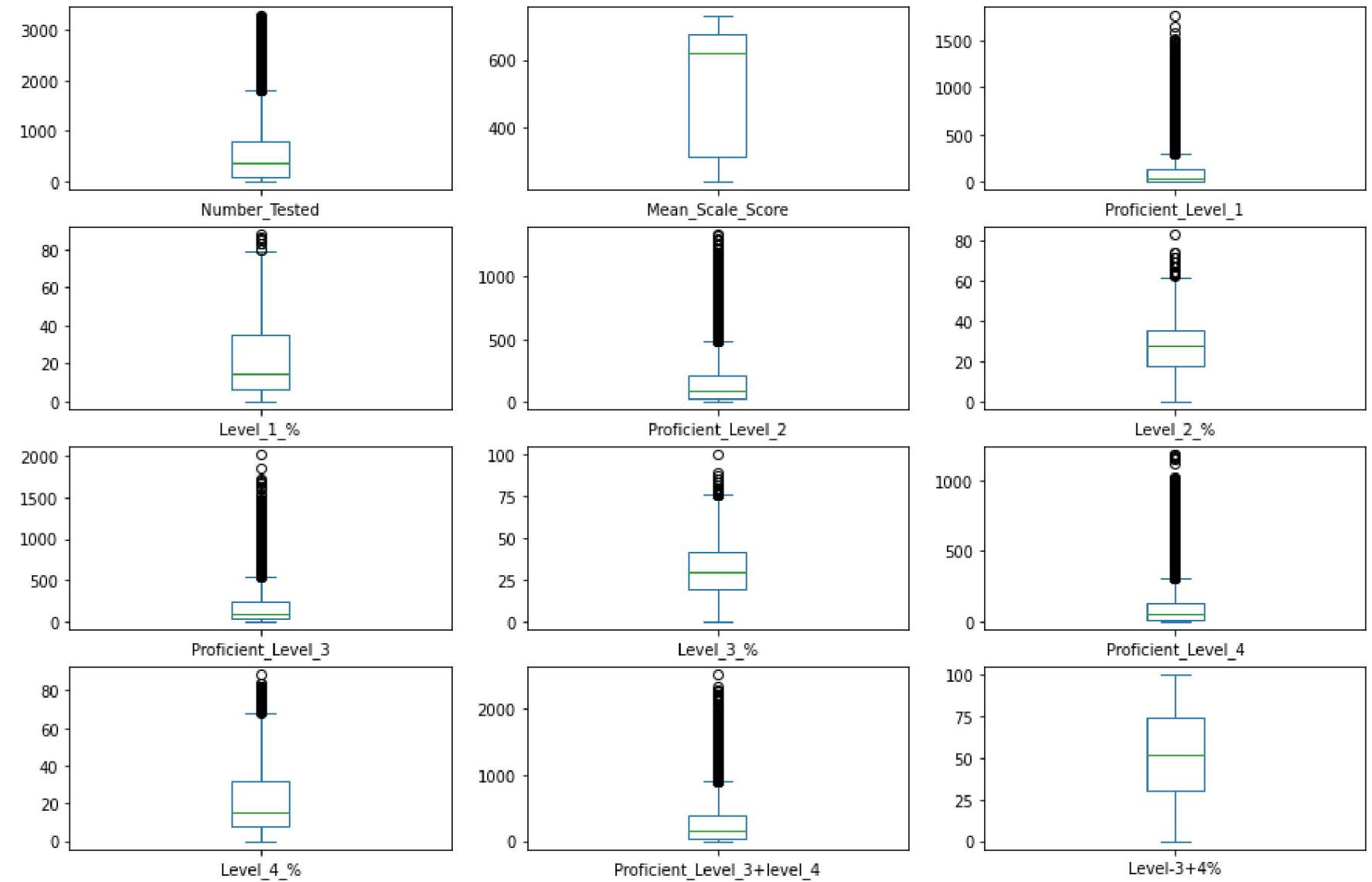
Out[107...]

```
Number_Tested           AxesSubplot(0.125,0.71587;0.227941x0.16413)  
Mean_Scale_Score         AxesSubplot(0.398529,0.71587;0.227941x0.16413)  
Proficient_Level_1       AxesSubplot(0.672059,0.71587;0.227941x0.16413)  
Level_1_%                AxesSubplot(0.125,0.518913;0.227941x0.16413)
```

```

Proficient_Level_2          AxesSubplot(0.398529,0.518913;0.227941x0.16413)
Level_2_%                   AxesSubplot(0.672059,0.518913;0.227941x0.16413)
Proficient_Level_3          AxesSubplot(0.125,0.321957;0.227941x0.16413)
Level_3_%                   AxesSubplot(0.398529,0.321957;0.227941x0.16413)
Proficient_Level_4          AxesSubplot(0.672059,0.321957;0.227941x0.16413)
Level_4_%                   AxesSubplot(0.125,0.125;0.227941x0.16413)
Proficient_Level_3+level_4  AxesSubplot(0.398529,0.125;0.227941x0.16413)
Level-3+4%                  AxesSubplot(0.672059,0.125;0.227941x0.16413)
dtype: object

```



In [107...]

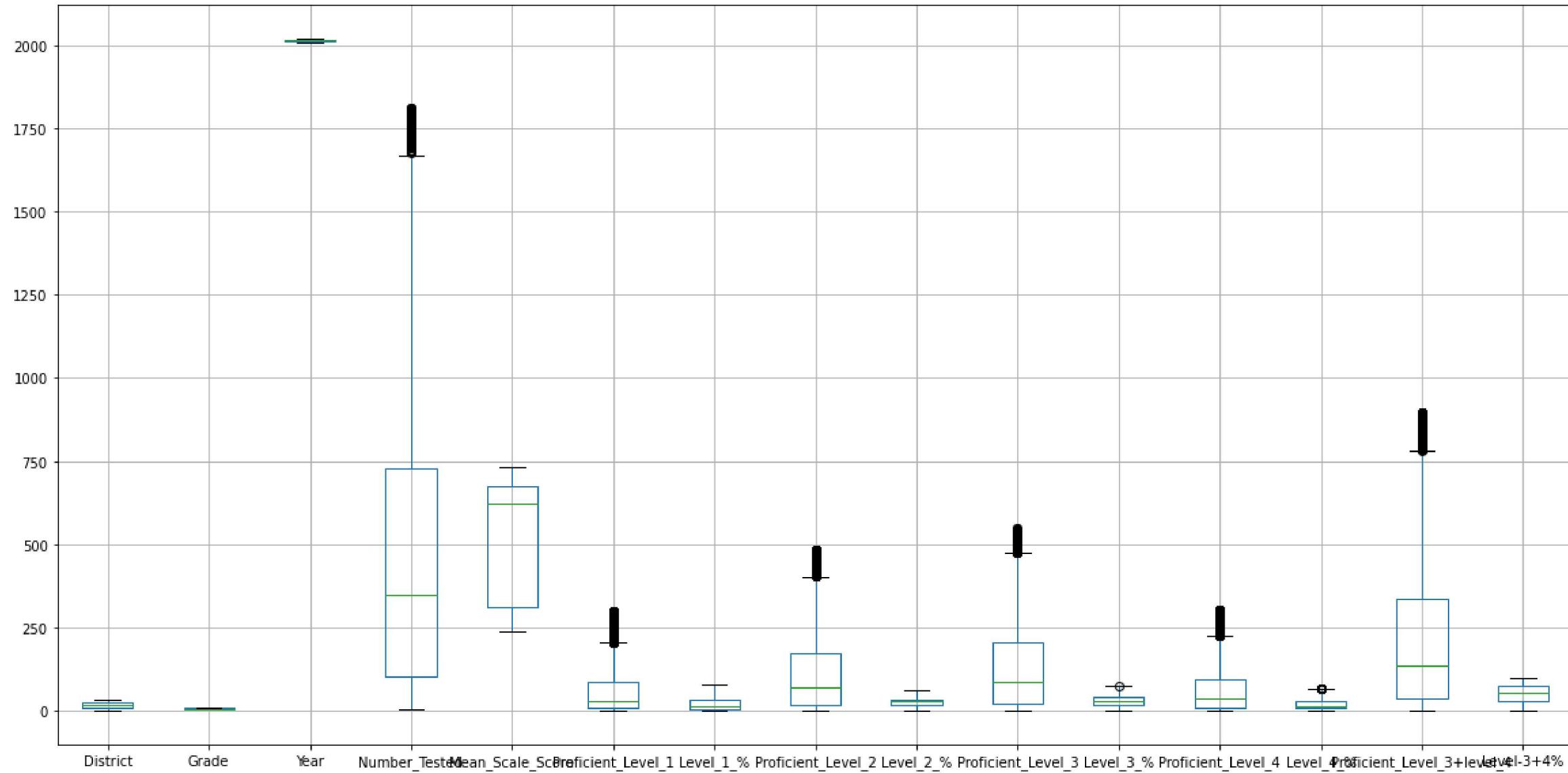
```

q1= numerical.quantile(0.25)
q3=numerical.quantile(0.75)
iqr=q3-q1
cleandff = numerical[~((numerical<(q1-1.5*iqr)) | (numerical>(q3+1.5*iqr)))]
clean_data = pd.concat([categorical, cleandff], axis=1)

```

In [107...]

```
clean_data.boxplot(figsize=(20,10));
```



NOTE - Some data points are still reflecting as outliers in the plots in both datasets. But these data points are valid observations and these are validating other variables in the datasets. That's why, these haven't been discarded.

Now both the datasets are cleaned.

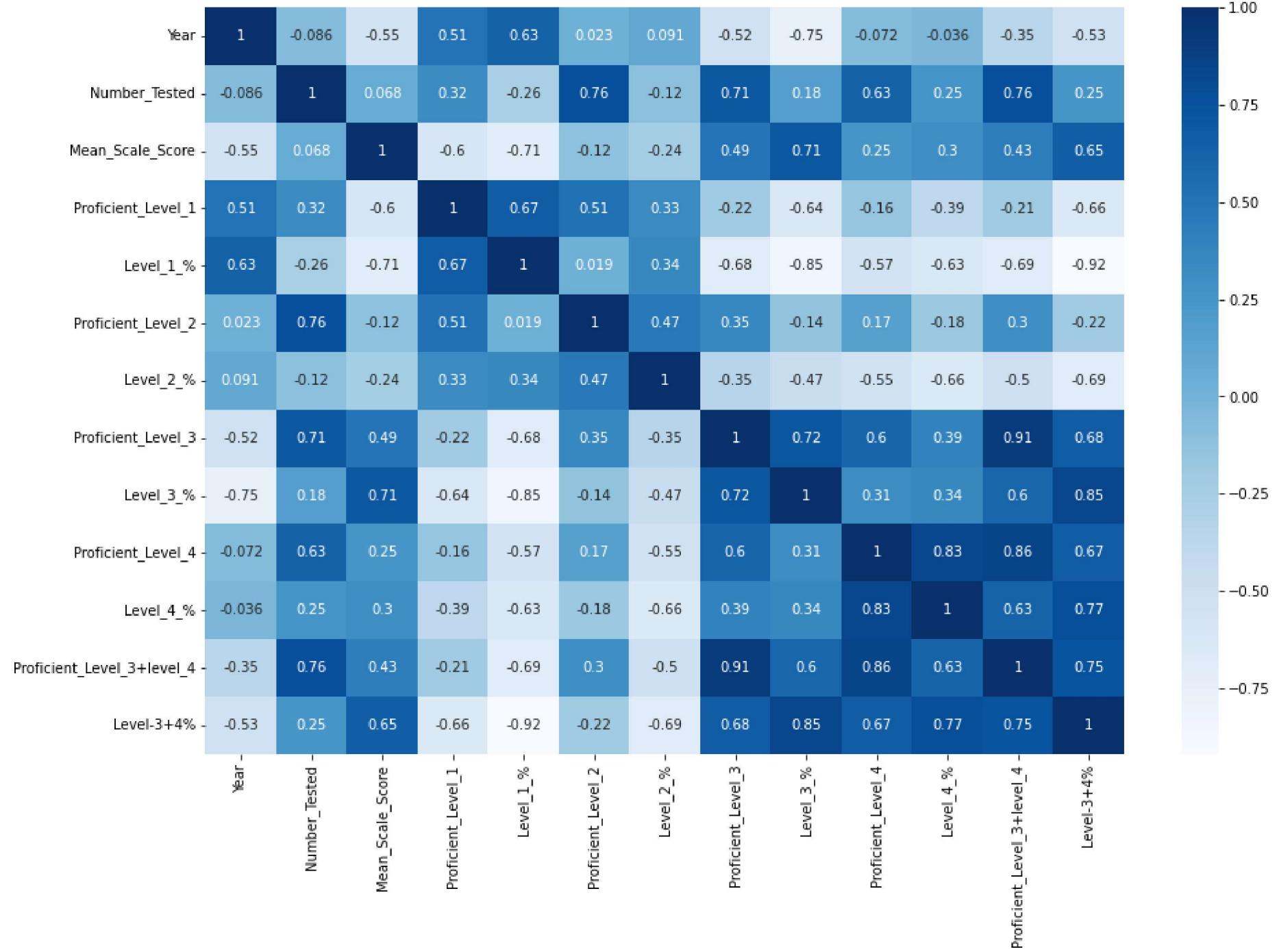
Next step is QA and validate the datasets.

1. Does the data make sense?
2. Does it prove or disprove your theory and expected outcomes.
3. Can you find the trends and patterns in the data that can help you unleash the insights or develop a new theory upon it.
4. if not, what is the issue?
5. is it because of data quality or in other respects.

This can be achieved by asking analytical queries and data visualization.

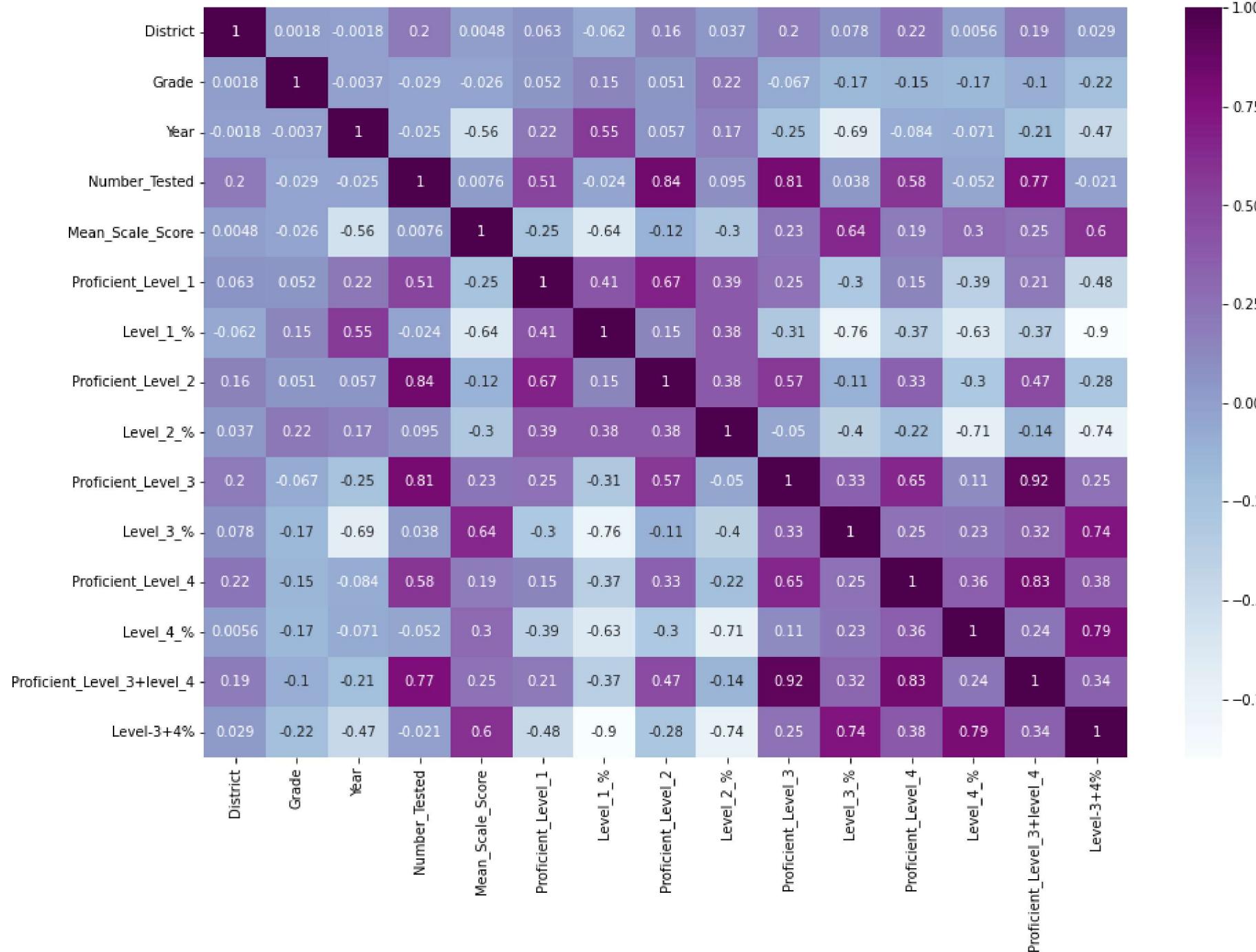
In [107]:

```
# Correlation tool can find the positive and negative relation between the variables.
plt.figure(figsize=(15,10))
sns.heatmap(outlier_removed_data.corr(),cbar=True,annot=True,cmap='Blues');
```



In [107]:

```
plt.figure(figsize=(15,10))
sns.heatmap(clean_data.corr(),cbar=True,annot=True,cmap='BuPu');
```



In [107]:

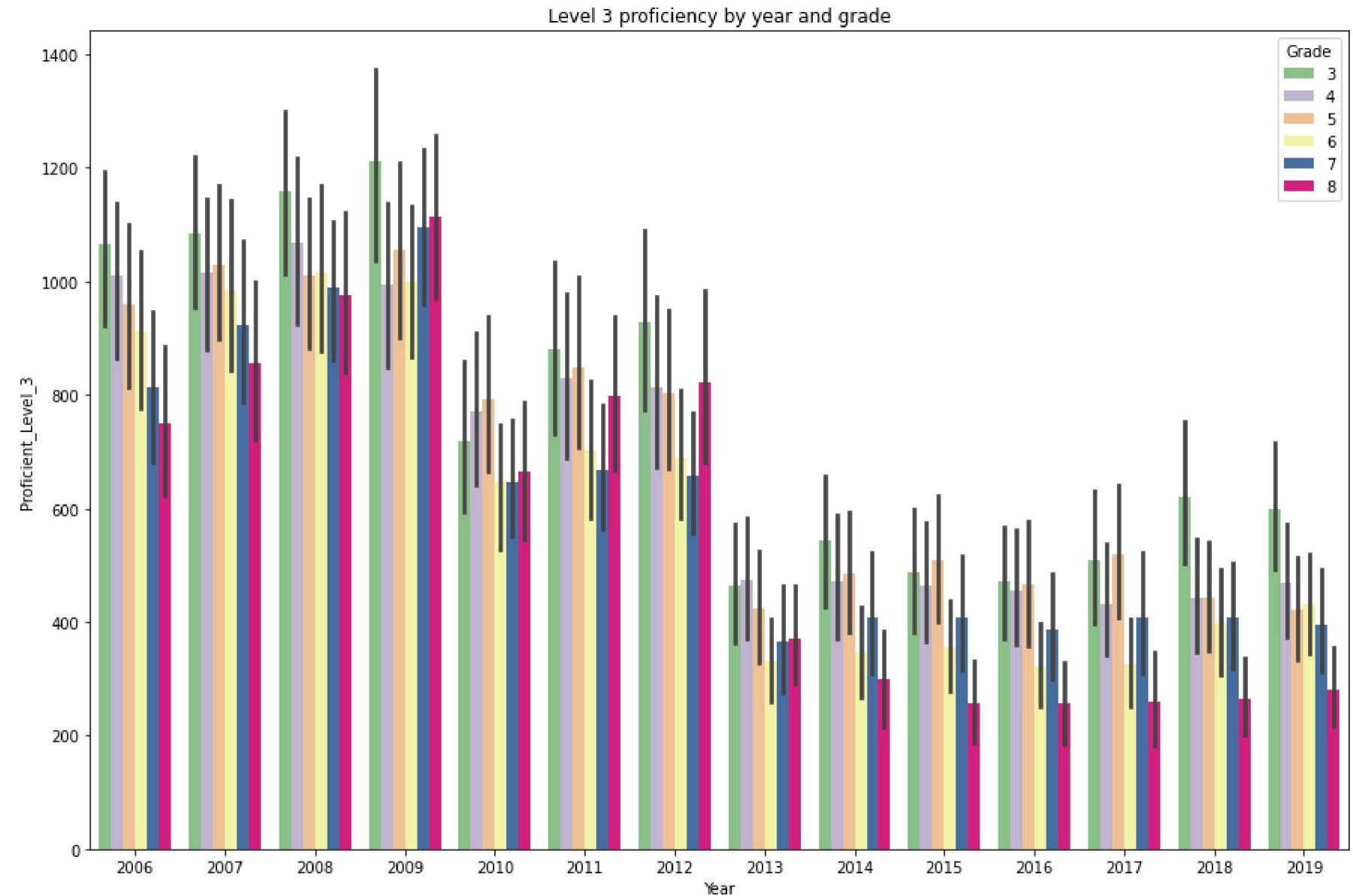
```
# Can create comparison, relationship and correlation charts to extract the insight from the data.
outlier_removed_data.head(2)
```

Out[107]:

	District	Grade	Year	Dempgraphic	Number_Tested	Mean_Scale_Score	Proficient_Level_1	Level_1_%	Proficient_Level_2	Level_2_%	Proficient_Level_3	Level_3_%	Proficient_Level_4	Level_4_%	Proficient_Level_3+level_4	Level-3+4%
0	1	3	2013	All Students	887	306.69	249.0	28.07	266.0	29.99	179.0	20.18	193.0	21.76	372.0	41.94
1	1	3	2014	All Students	845	307.95	225.0	26.63	223.0	26.39	204.0	24.14	193.0	22.84	397.0	46.98

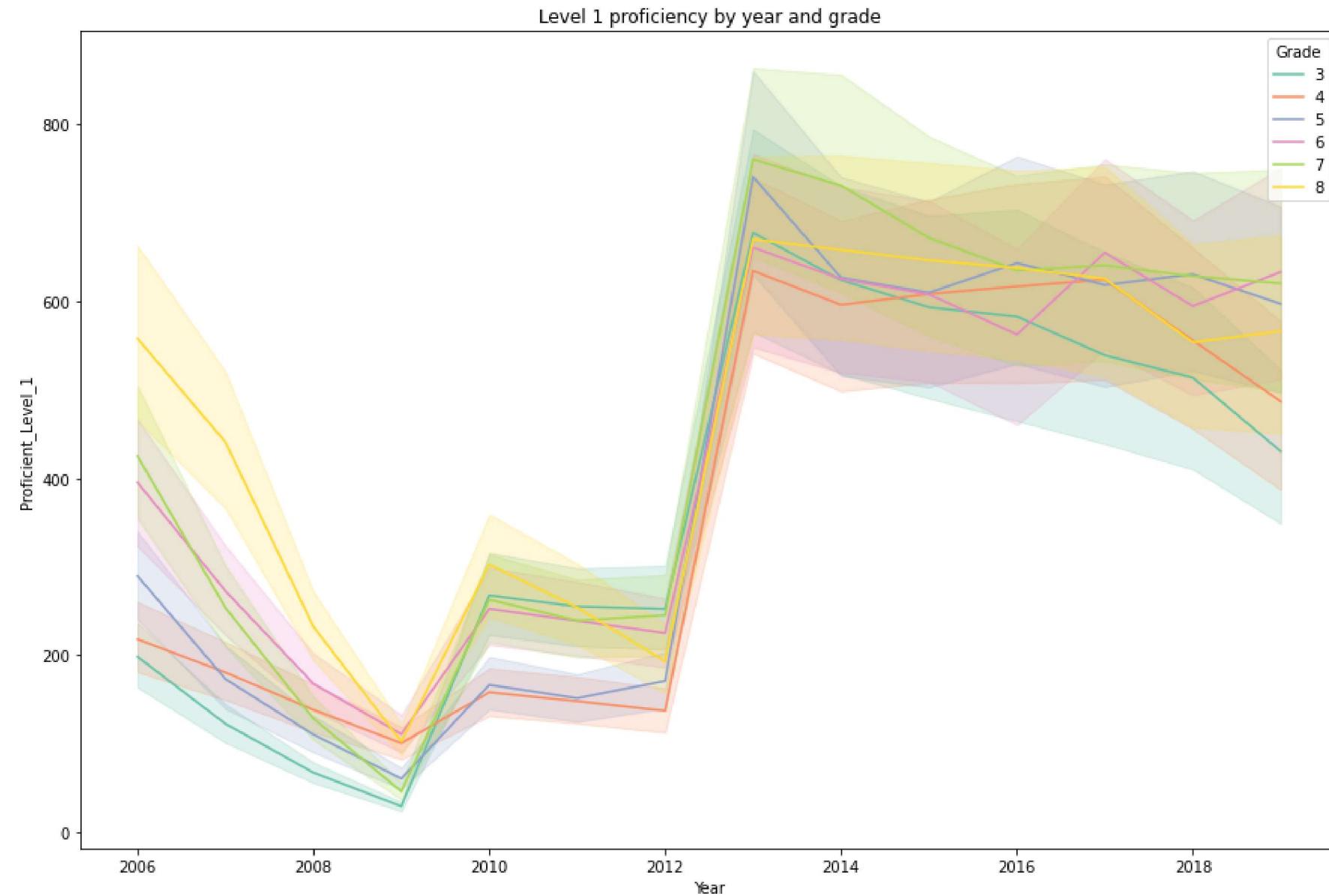
In [110]:

```
plt.figure(figsize=(15,10))
sns.barplot(data=outlier_removed_data, x='Year', y='Proficient_Level_3', hue='Grade', palette="Accent")
plt.title('Level 3 proficiency by year and grade')
plt.show()
```



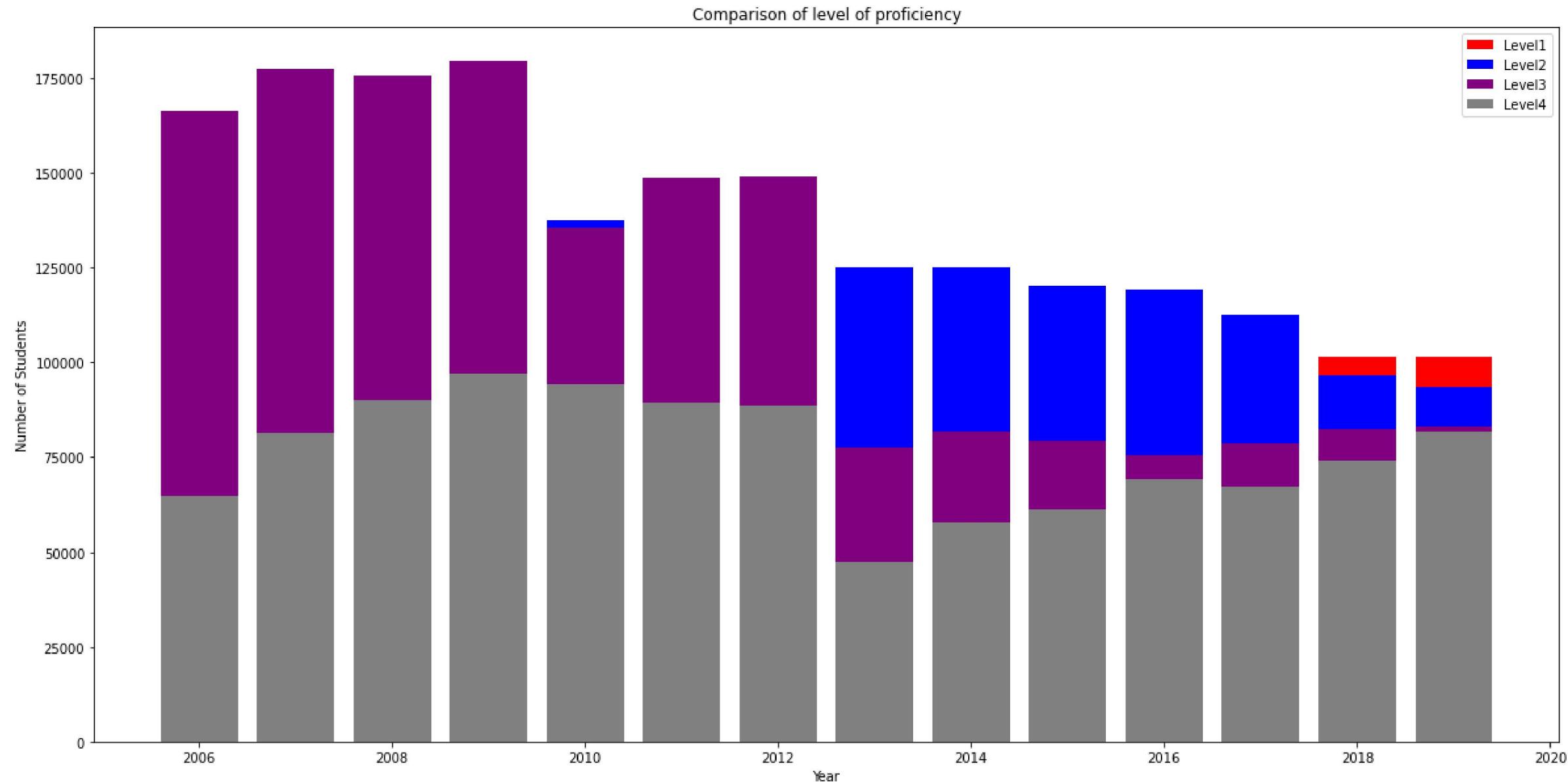
In [108]:

```
plt.figure(figsize=(15,10))
sns.lineplot(data=outlier_removed_data, x='Year', y='Proficient_Level_1', hue='Grade', palette="Set2")
plt.title('Level 1 proficiency by year and grade')
plt.show()
```



```
In [108]: grouped=outlier_removed_data.groupby('Year').sum()
```

```
In [110]: plt.figure(figsize=(20,10))
plt.bar(grouped.index, grouped.Proficient_Level_1, label='Level1', color='red')
plt.bar(grouped.index, grouped.Proficient_Level_2, label='Level2', color='blue')
plt.bar(grouped.index, grouped.Proficient_Level_3, label='Level3', color='purple')
plt.bar(grouped.index, grouped.Proficient_Level_4, label='Level4', color='grey')
plt.xlabel('Year')
plt.ylabel('Number of Students')
plt.title('Comparison of level of proficiency')
plt.legend()
plt.show()
```



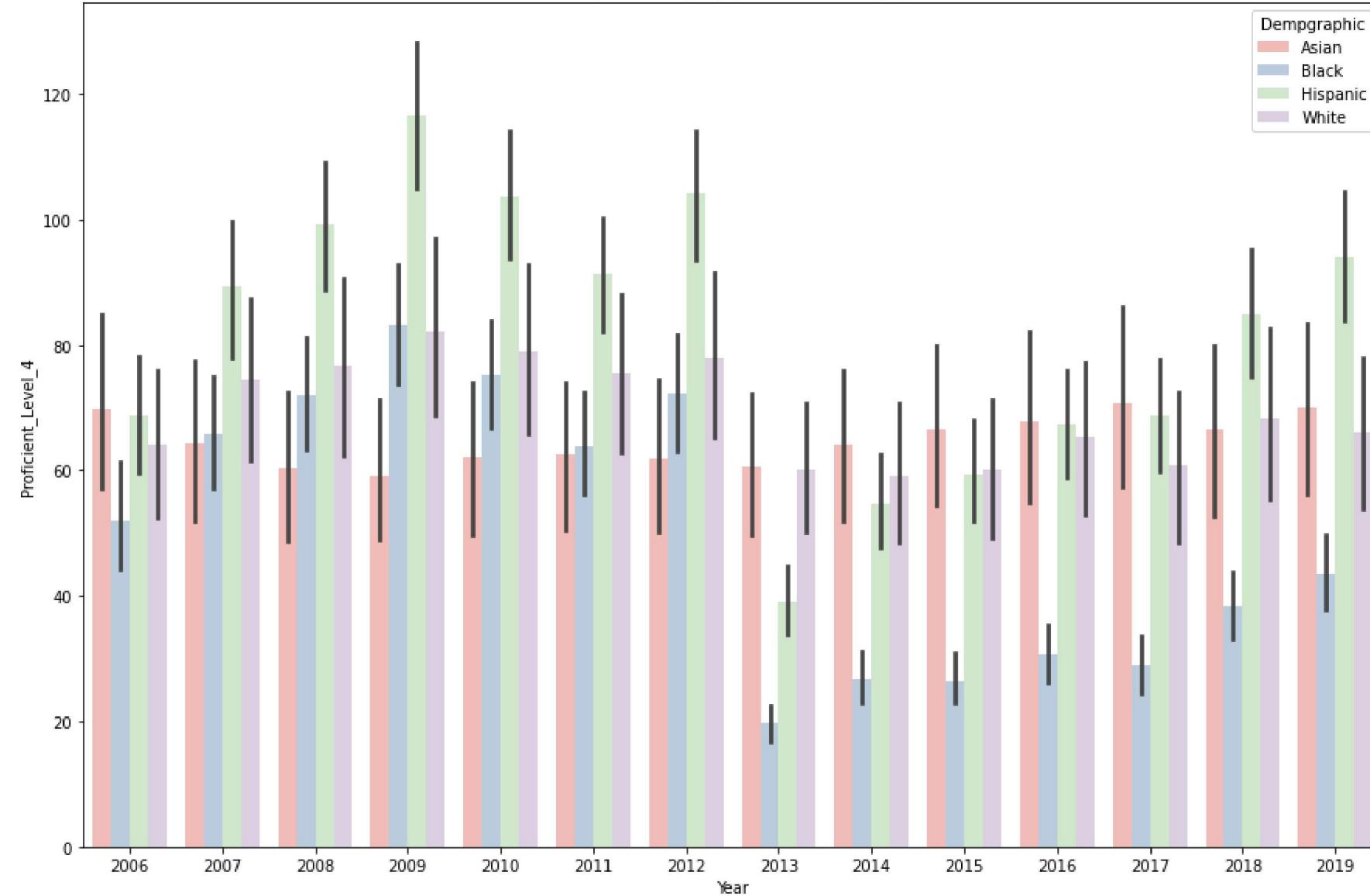
```
In [108]: clean_data['Grade'].astype('str')
clean_data.head(2)
```

```
Out[108]:
```

	District	Grade	Year	Dempgraphic	Number_Tested	Mean_Scale_Score	Proficient_Level_1	Level_1_%	Proficient_Level_2	Level_2_%	Proficient_Level_3	Level_3_%	Proficient_Level_4	Level_4_%	Proficient_Level_3+level_4	Level-3+4%
0	1	3	2006	Asian	155.0	704.0	4.0	2.6	4.0	2.6	68.0	43.9	79.0	51.0	147.0	94.8
1	1	3	2006	Black	185.0	659.0	21.0	11.4	47.0	25.4	104.0	56.2	13.0	7.0	117.0	63.2

```
In [110]: plt.figure(figsize=(15,10))
sns.barplot(data=clean_data, x='Year', y='Proficient_Level_4', hue='Dempgraphic', palette="Pastel1")
plt.title('Level 4 proficiency by year and demographic')
plt.show()
```

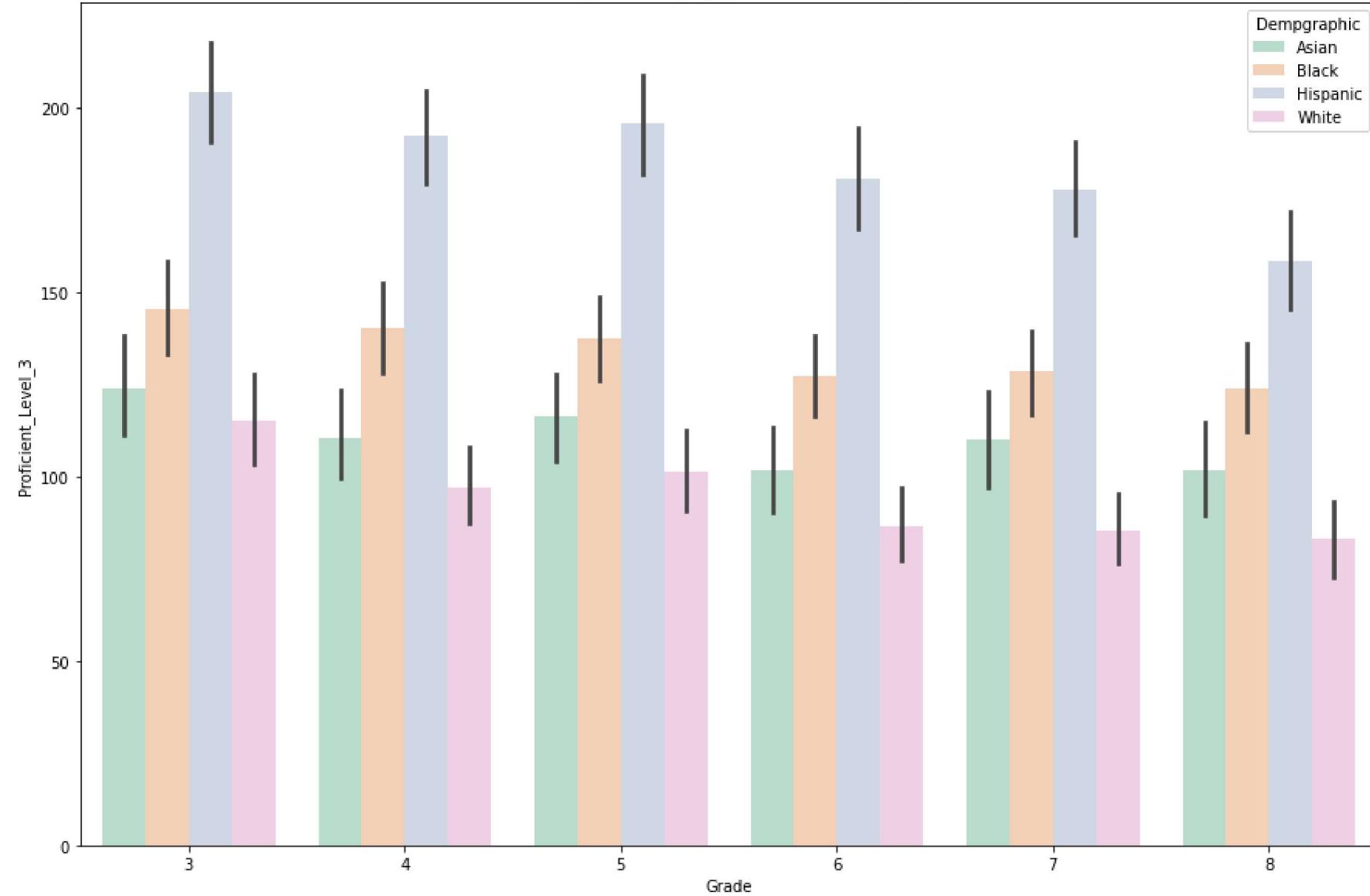
Level 4 proficiency by year and demographic



In [111...]

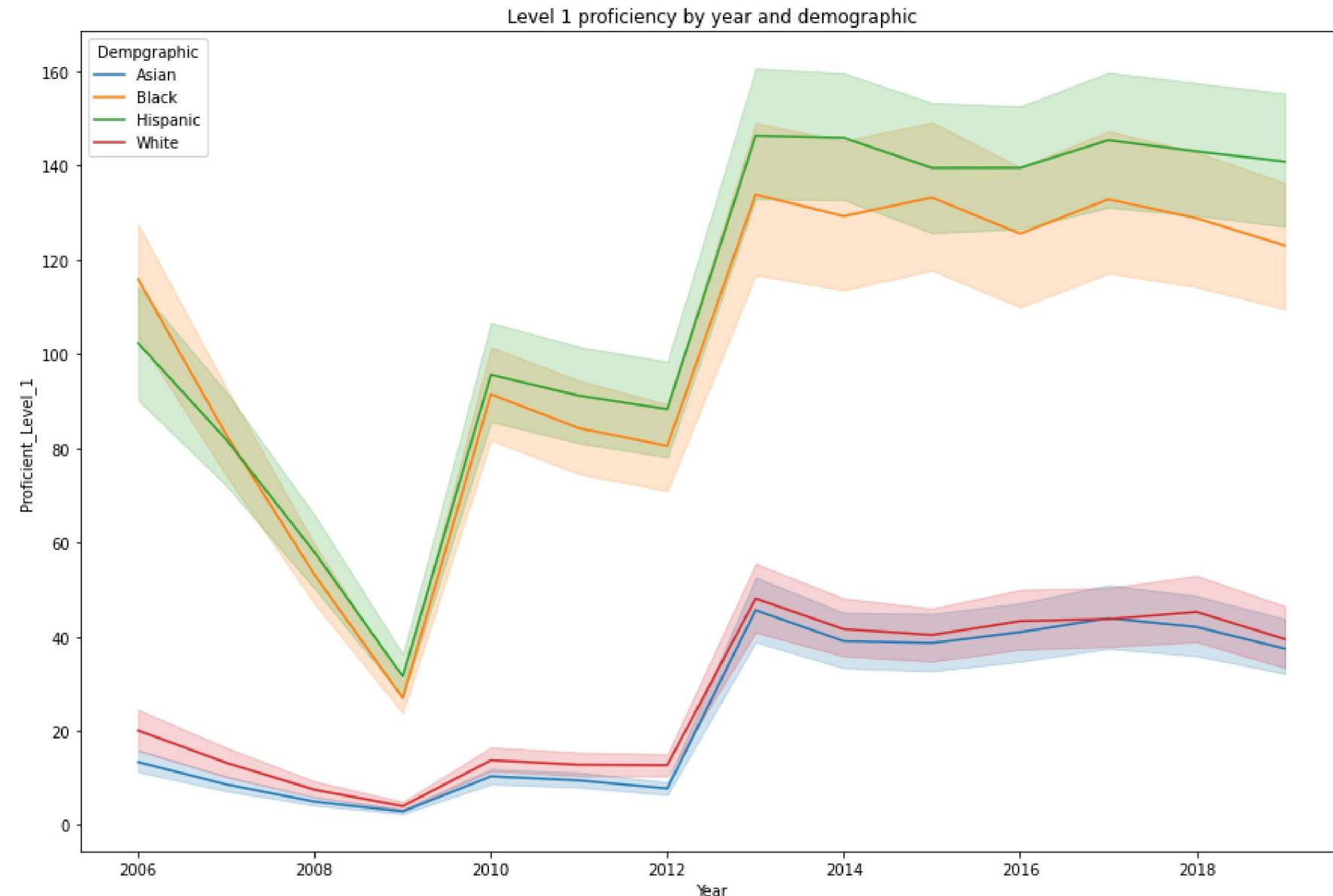
```
plt.figure(figsize=(15,10))
sns.barplot(data=clean_data, x='Grade', y='Proficient_Level_3', hue='Demographic', palette="Pastel2")
plt.title('Level 3 proficiency by grade and demographic')
plt.show()
```

Level 3 proficiency by grade and demographic



In [108]:

```
plt.figure(figsize=(15,10))
sns.lineplot(data=clean_data, x='Year', y='Proficient_Level_1', hue='Demographic')
plt.title('Level 1 proficiency by year and demographic')
plt.show()
```



These datasets are ready for further analysis.

In []: