Student Demographics Analysis

Sagar Limbu

Objectives/Goals Problem 3:

- Test1 is the test students took the first day they began using the software to set their benchmark level
- Comp Score is the overall grade level equivalent of the score the student earned on the progress test
- Hours used is how many hours the student has used the software

- **Note: The dates, scores, and hours used are real data, but all the names of students, schools, and classes are fake. None of this is FERPA-protected data.**

Distributions of Students Data

```
In [4]: df_school.describe()
```

Out[4]:

	Username	Grade Level	Test1_CompScore	mostRecentCompScore	HoursUsed
count	1000.000000	1000.000000	764.000000	803.000000	803.000000
mean	177261.269000	9.756000	5.543194	5.976339	10.243989
std	77093.772978	0.977456	3.078606	2.908529	7.833847
min	100005.000000	9.000000	0.000000	0.000000	0.352500
25%	100844.750000	9.000000	4.000000	4.000000	4.124028
50%	200661.000000	9.000000	6.000000	6.000000	8.620556
75%	204134.250000	10.000000	8.000000	8.000000	14.396528
max	316485.000000	12.000000	11.000000	12.000000	63.882222

Possible Analysis and Findings:

- What are the FEATURES on the data set
- What FEATURES can give insight of the student data
- Types of data
- Exploratory Analysis on Students who has both **Start of Software Date** and **Comp Scores**
- Students with **NO start of Software Date** but have used Software
- Software Starting Date by MONTH
- Performance between Different Grade Levels
- Performance based on Different Schools

Columns/Features:

Username	First Name	Last Name	School Name	Grade Level	Class Name	Test1_Date	Test1_CompScore	mostRecentCompScore	HoursUsed
100204	Gina	Hooper	Schmitt High School	9	Rivera - Period 7	8/5/2020	6.0	6.0	11.445556
100205	Dane	Bates	Schmitt High School	9	Davila - Period 7	8/5/2020	6.0	6.0	10.396944
100206	Autumn	Flowers	Cherry Early College High School	9	Acosta - Period 1	8/5/2020	3.0	6.0	13.669722
100207	Terrell	Arnold	Freeman Career Technical School	9	Wu - Period 7	8/5/2020	9.0	9.0	11.121389

Exploring Data:

Data TYPES and Information

```
In [9]: df_school.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Username             1000 non-null   int64  
 1   First Name           1000 non-null   object 
 2   Last Name            1000 non-null   object 
 3   School Name          1000 non-null   object 
 4   Grade Level          1000 non-null   int64  
 5   Class Name           1000 non-null   object 
 6   Test1_Date           764 non-null    object 
 7   Test1_CompScore      764 non-null    float64
 8   mostRecentCompScore  803 non-null    float64
 9   HoursUsed            803 non-null    float64
dtypes: float64(3), int64(2), object(5)
memory usage: 78.2+ KB
```

MISSING VALUES in INDIVIDUAL FEATURE (SUM)

```
: for i in range(0,len (df_school.columns)):
      print(df_school.columns[i],":",df_school[df_school.columns[i]].isnull().sum())
```

```
Username : 0
First Name : 0
Last Name : 0
School Name : 0
Grade Level : 0
Class Name : 0
Test1_Date : 236
Test1_CompScore : 236
mostRecentCompScore : 197
HoursUsed : 197
```

POSSIBLE REPLACEMENT OF NULL VALUES:

- Assigning ZERO value with SCORE
- Assigning missing DATE with most COMMON Start DATE
- Droping the rows that has missing values

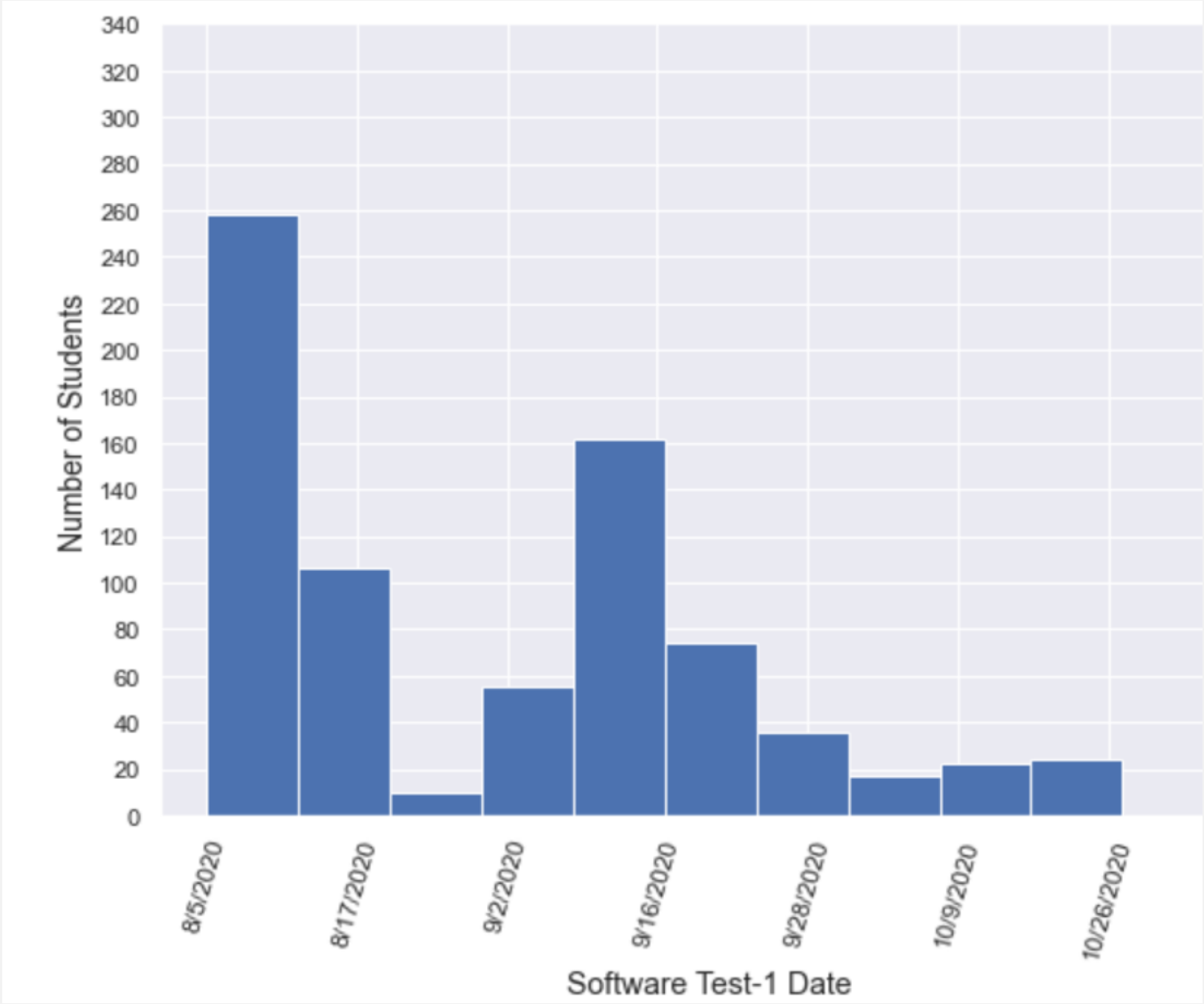
Approach:

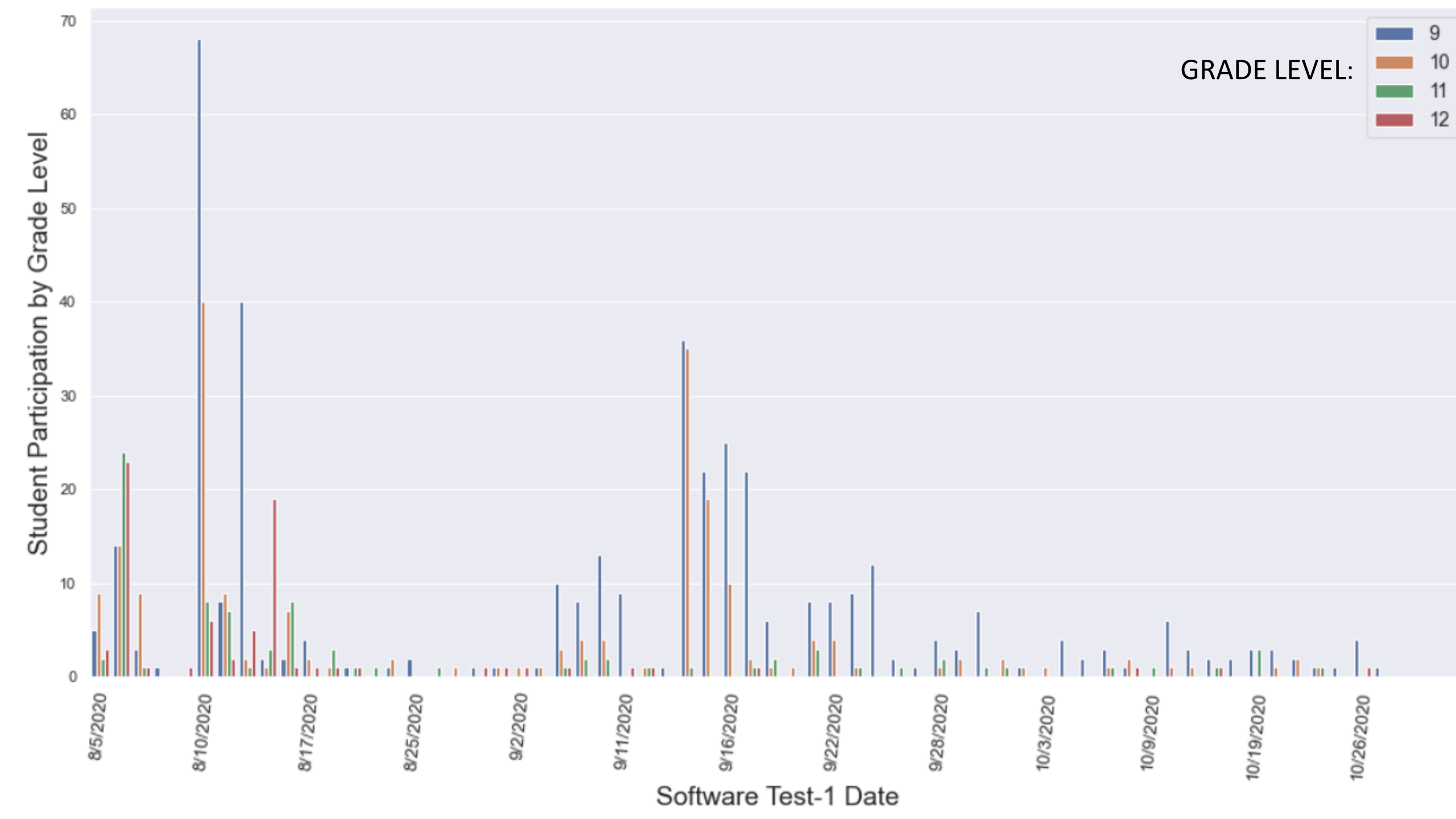
- Upcoming slides show data analysis from students who has the **Test1_Date Recorded** .i.e. Test1_Date= NOT NULL
- The **three ending slides** will show analysis on the **missing data** for students **i.e.** Test1_Date and Test1_CompScore is not recorded or **missing**

Total Number of Students starting Software Test-1 Date (month-day-year)

- Total Students by Grade Level:

- Grade 9= 539
- Grade 10= 256
- Grade 11=115
- Grade 12= 90



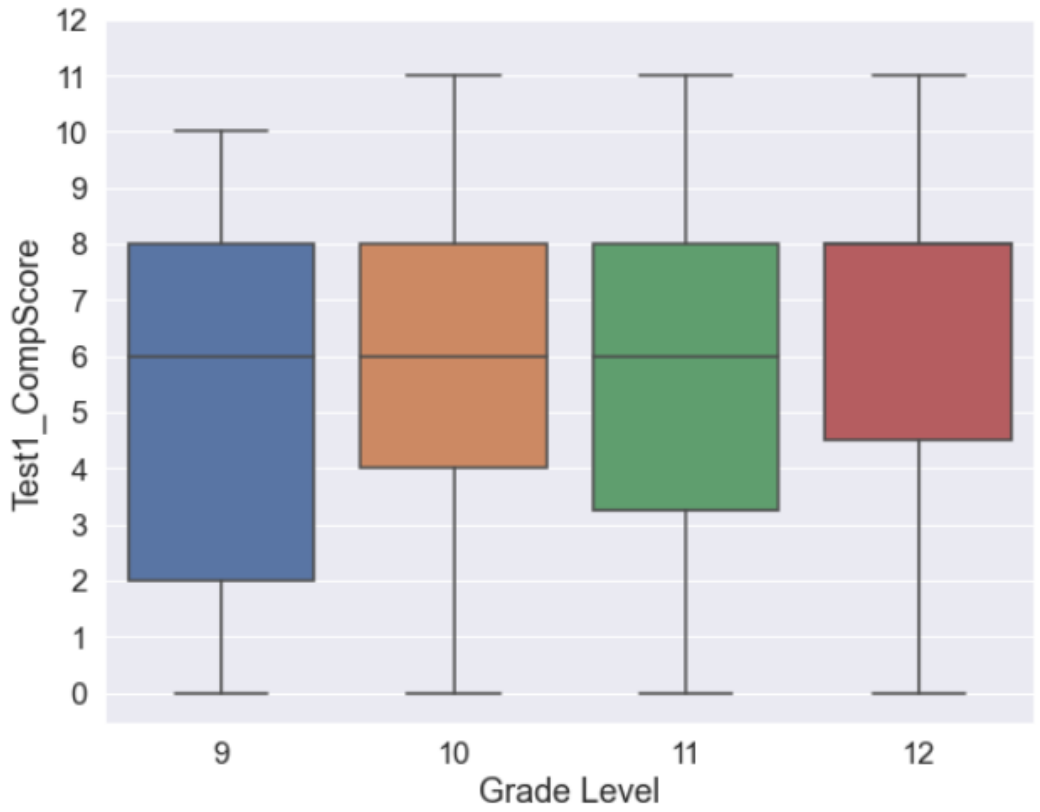

GRADE LEVEL:
9
10
11
12
Student Participation by Grade Level
0
10
20
30
40
50
60
70
8/5/2020
8/10/2020
8/17/2020
8/25/2020
9/2/2020
9/11/2020
9/16/2020
9/22/2020
9/28/2020
10/3/2020
10/9/2020
10/19/2020
10/26/2020
Software Test-1 Date

Result/Findings:

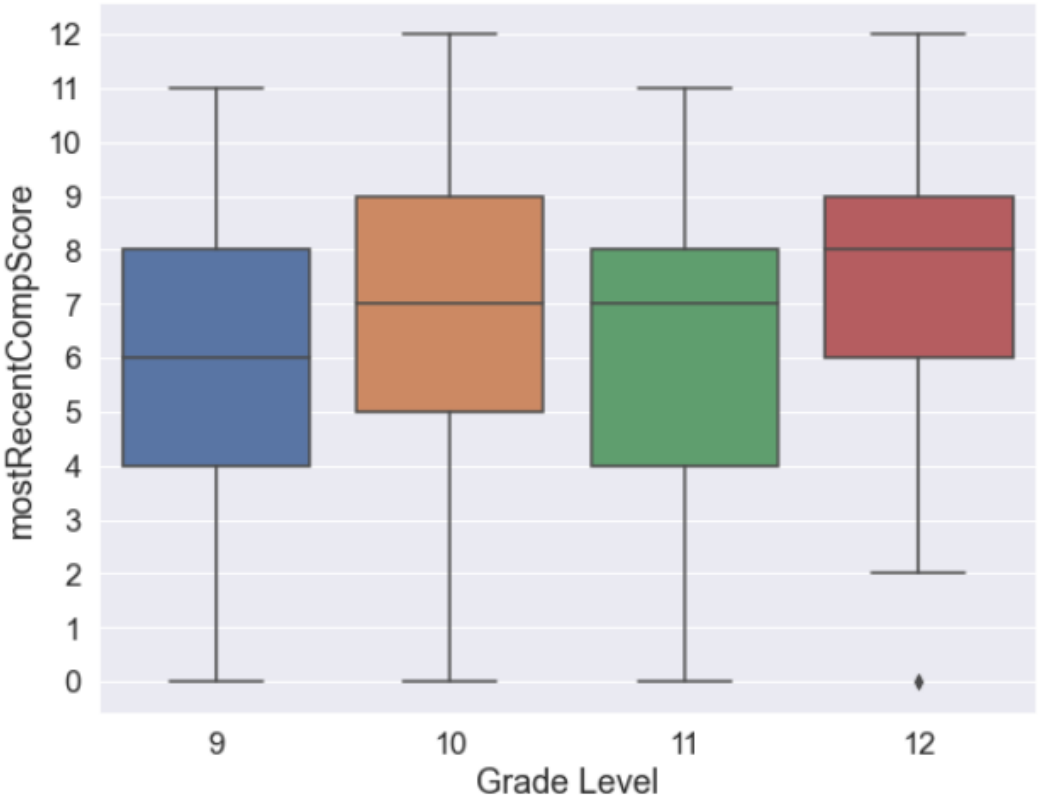
- Previous Graph shows that large numbers of students from GRADE 9 started Early using the software than rest students from different grade Levels
- In the next Box Plot Graph, we can see the Scores of students on First Test-1 CompScore and Recent Comp Score

Box Plot: First Test Score and Most Recent Test Score by different Grade Level

First Test Score



Most Recent Test Score

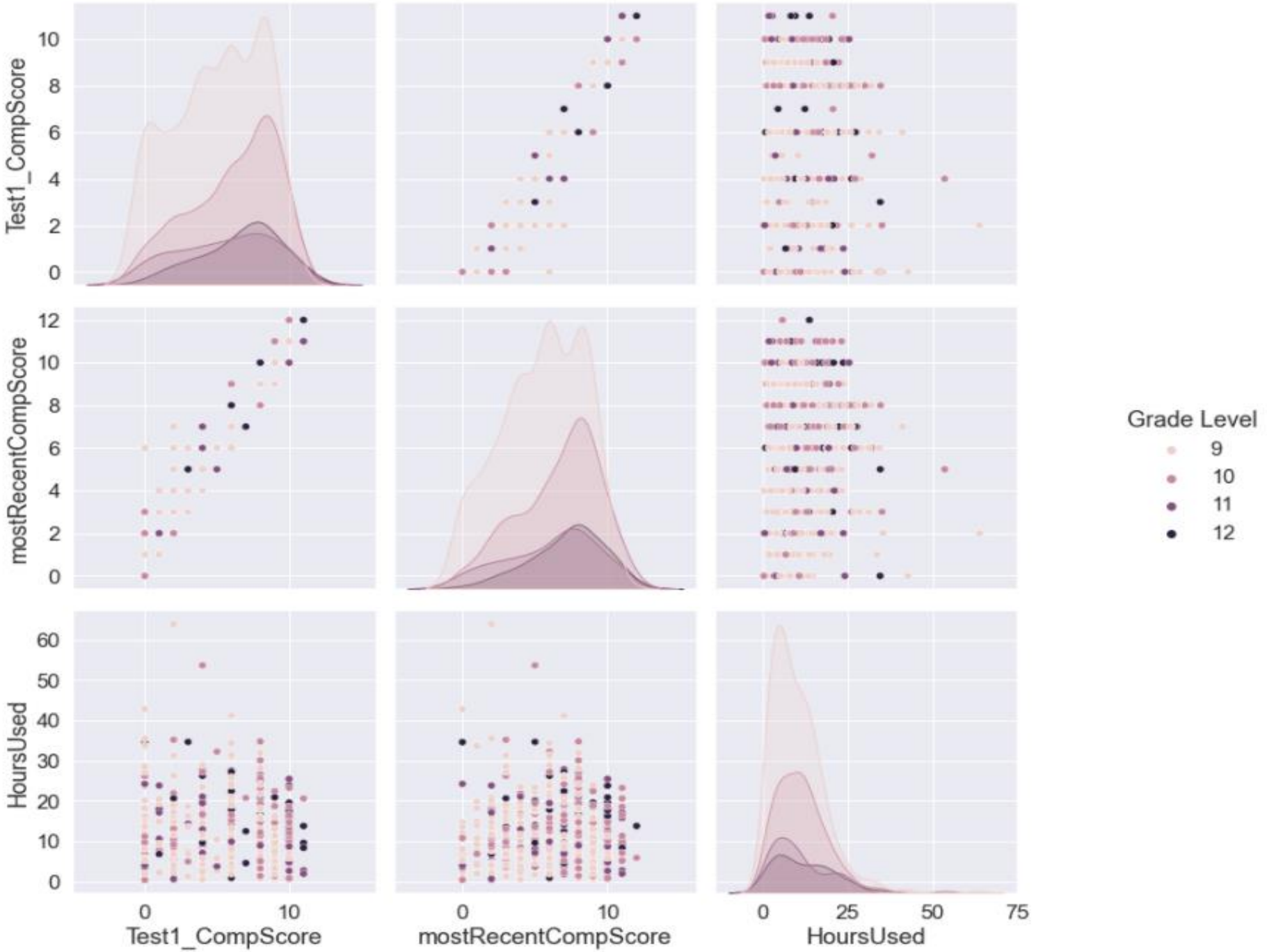


Students **Performance Metrics** based on **Grade Levels**:

- **By Test1_CompScore**
- **Most Recent Comp Score**
- **Software Usage (Hrs)**

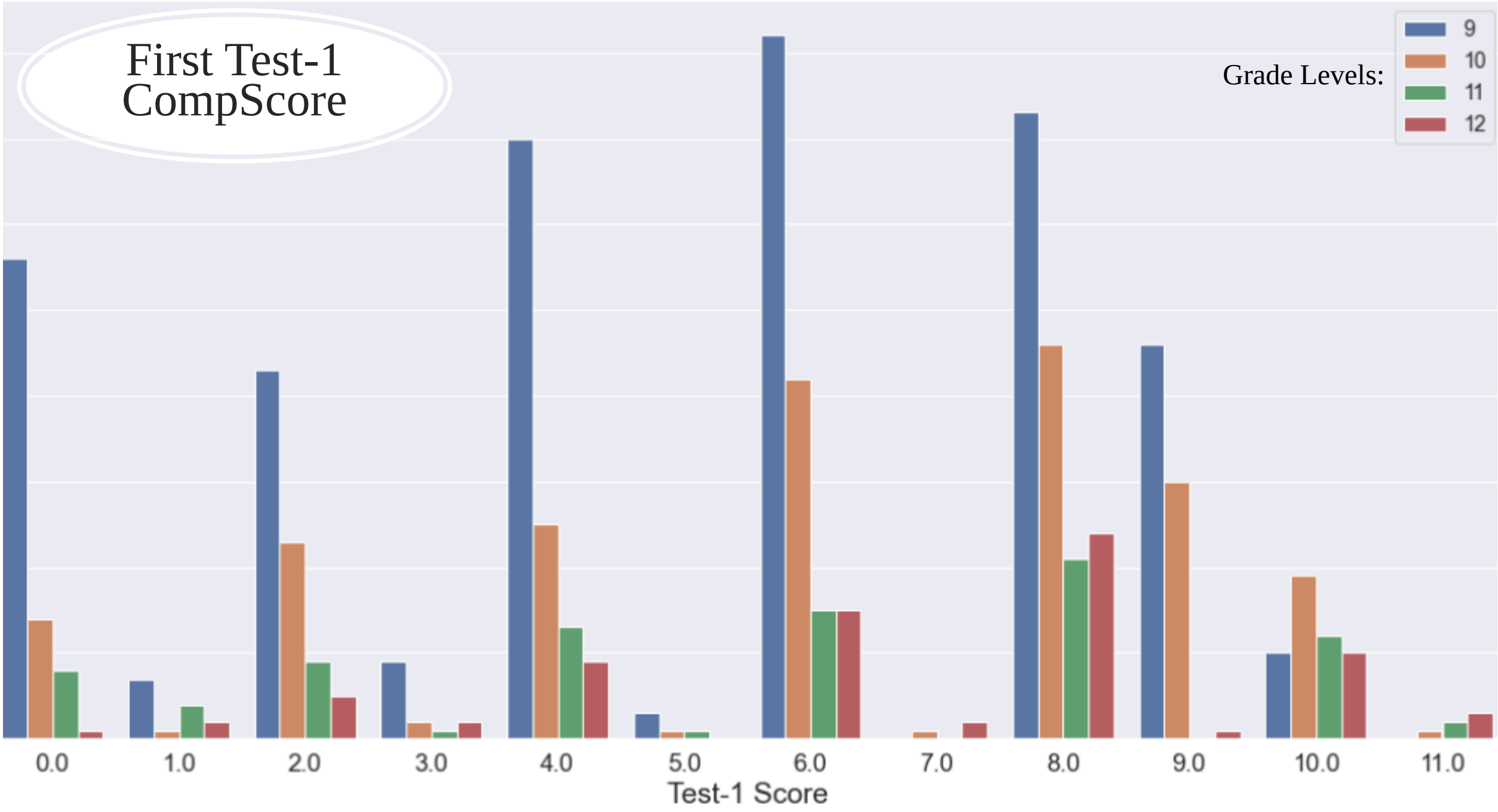
Findings:

EG: we can see the Software usage by hours and the Scores level at the Test-1 and Most Recent Score



TEST Score difference:

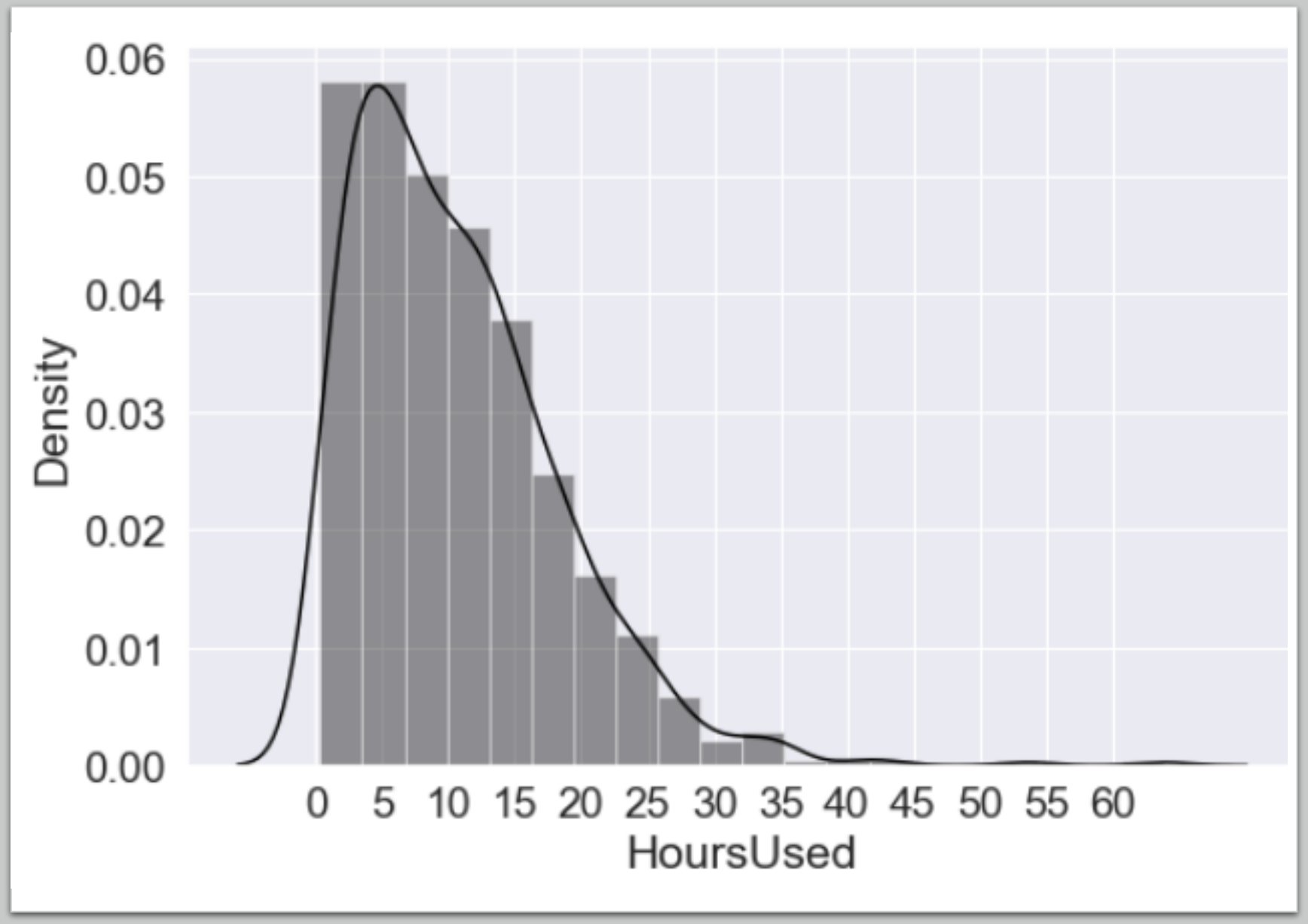
First Test1_CompScore and Most Recent CompScore


First Test-1 CompScore
Grade Levels:
9
10
11
12
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
Test-1 Score

Frequency Distribution of Software Usage:

Students using the Software Usage by hours

Findings:
- Not Normally distributed

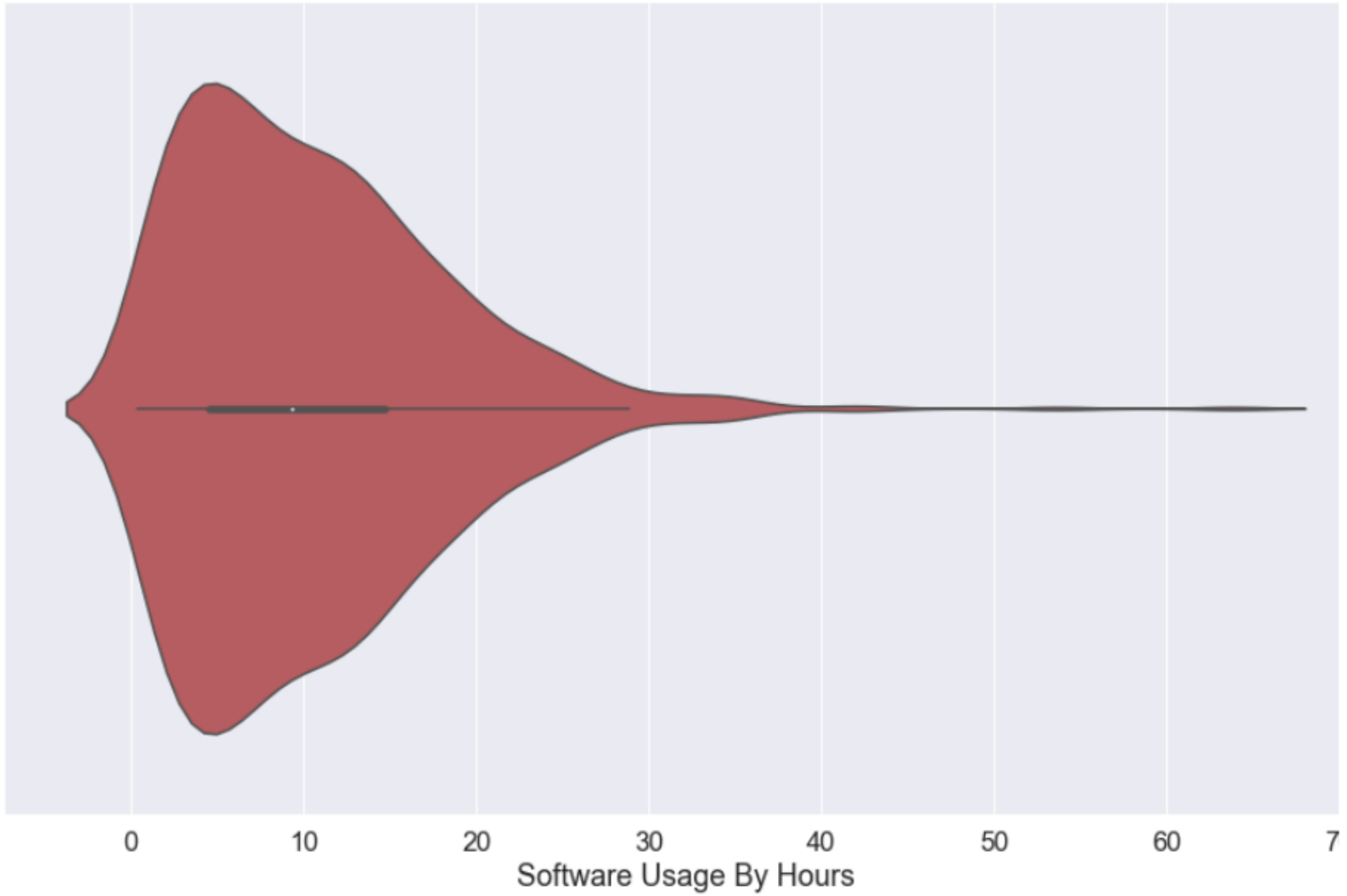


Frequency Distribution of Software Usage:

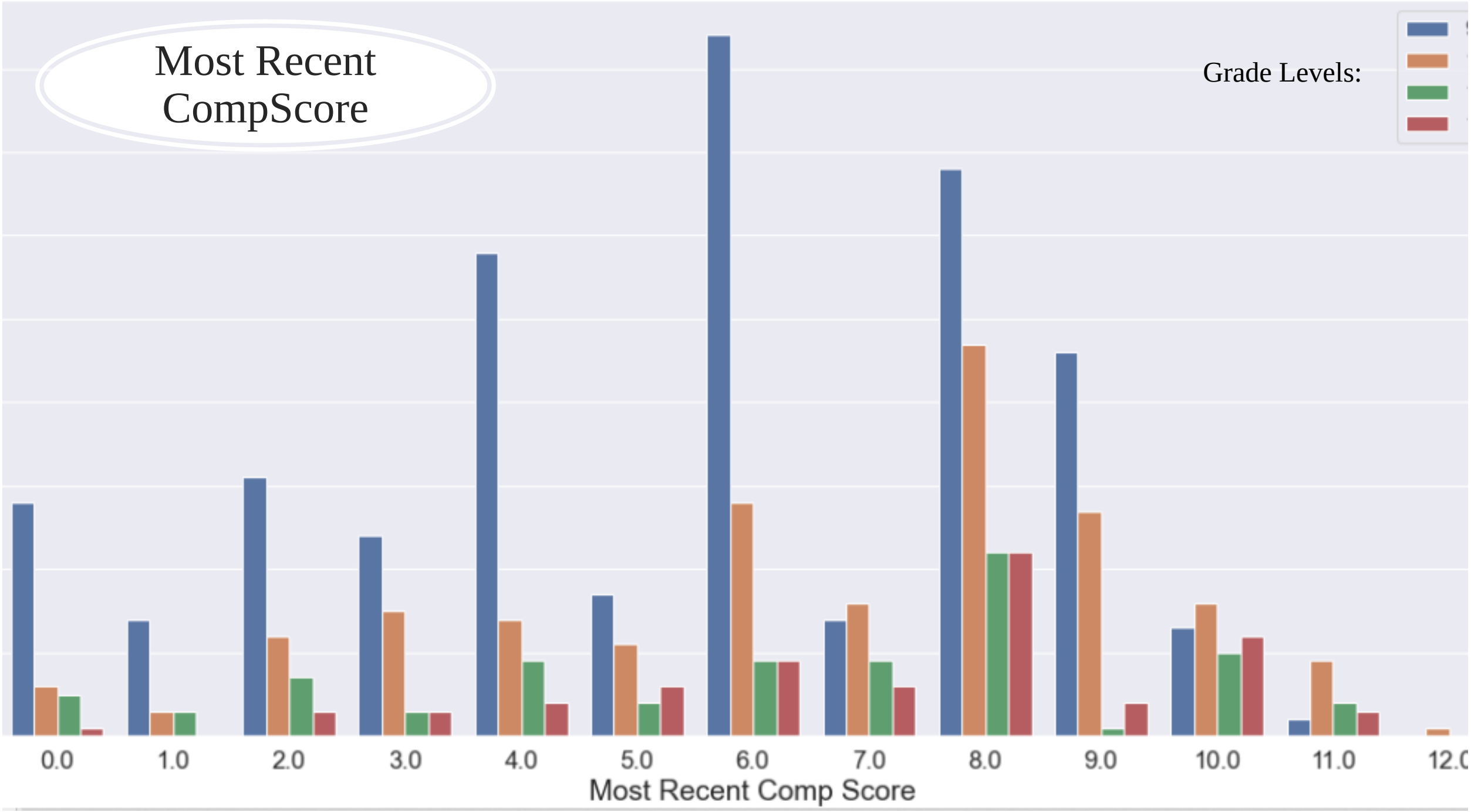
Total Students using the Software Usage by hours

USING violin plot

```
sns.violinplot(x='HoursUsed', data= students_test_rec, color='r')
plt.xlabel('Software Usage By Hours', fontsize=18)
plt.gcf().set_size_inches(15,9)
```



Most Recent CompScore



EXTRA TESTS:

To Analyze the Scores based on different School Names

STEPS:

- Using Label Encoder() function

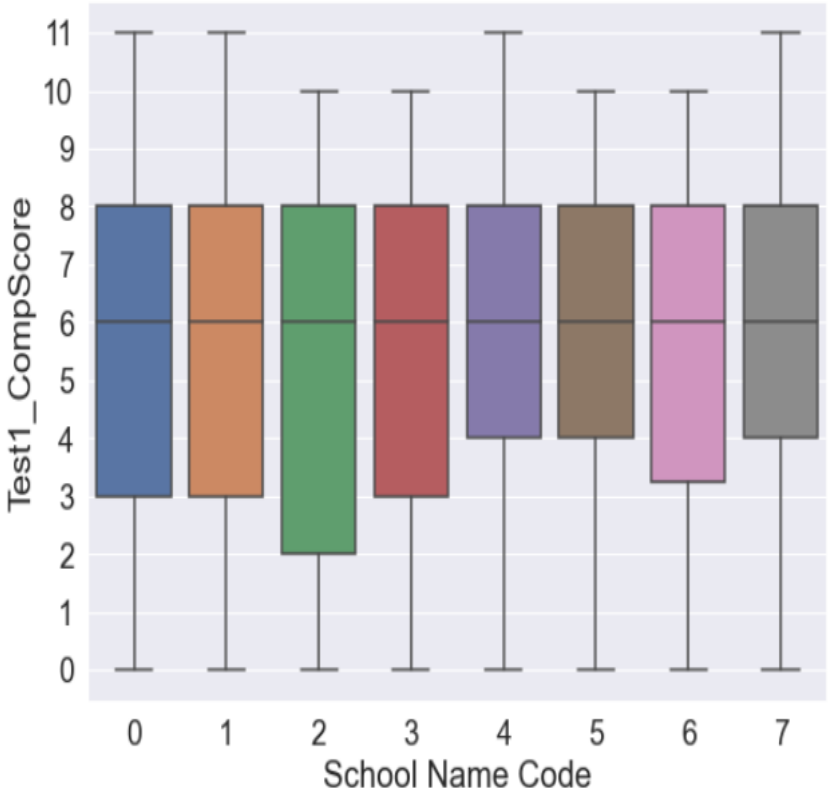
STEPS:

- Using the ‘sklearn’ preprocessing Library to **ENCODE** each School Names. Since the code requires numerical values and not strings. Here, I have assigned a UNIQUE CODE to each school name.
- By passing the school names to **Label Encoder** function which transforms the Strings into Unique Code

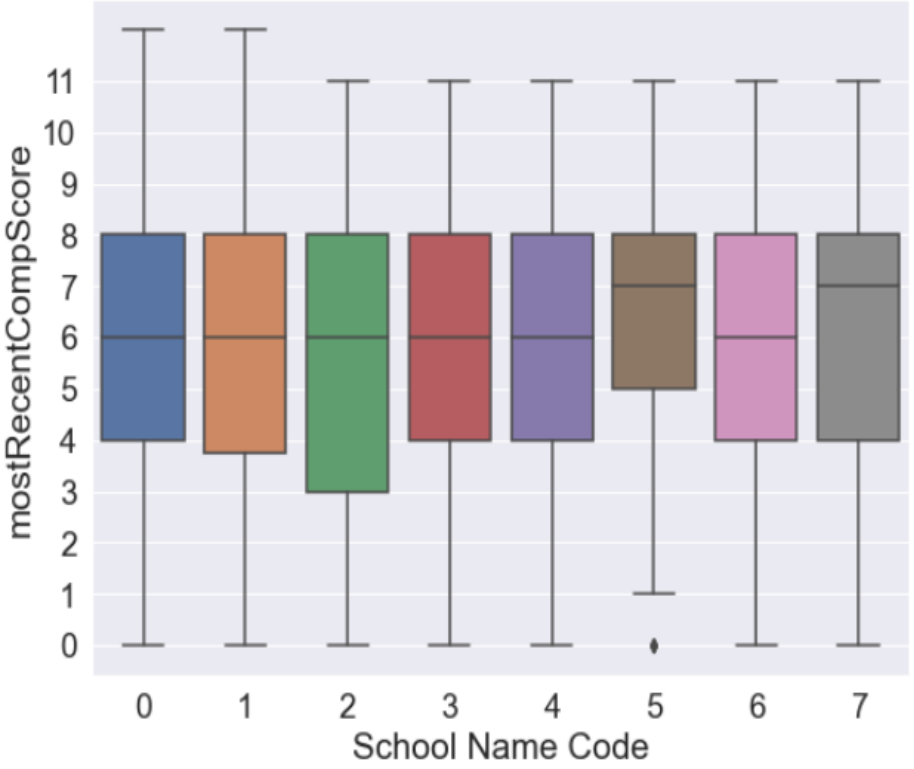
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
df_school_name= df_school.drop(columns=['First Name', 'Last Name'])
school_names= le.fit_transform(df_school_name['School Name'])
df_school_name['School Name Code']= school_names
```

School Name	School Code
Brooks High School	0
Cherry Early College High School	1
Douglas High School	2
Foley Arts Focus High School	3
Freeman Career Technical School	4
Jacobs High School	5
Lawson High School	6
Schmitt High School	7

FINDINGS: Most Recent SCORE Level show improvements of Students from each School.



TEST-1_CompScore



Most Recent CompScore

School Name	School Code
Brooks High School	0
Cherry Early College High School	1
Douglas High School	2
Foley Arts Focus High School	3
Freeman Career Technical School	4
Jacobs High School	5
Lawson High School	6
Schmitt High School	7

THINGS to CONSIDER

Students with NO ‘Starting Software Date (**Test Date**)’ also have’ **NO ‘ComP SCORE**

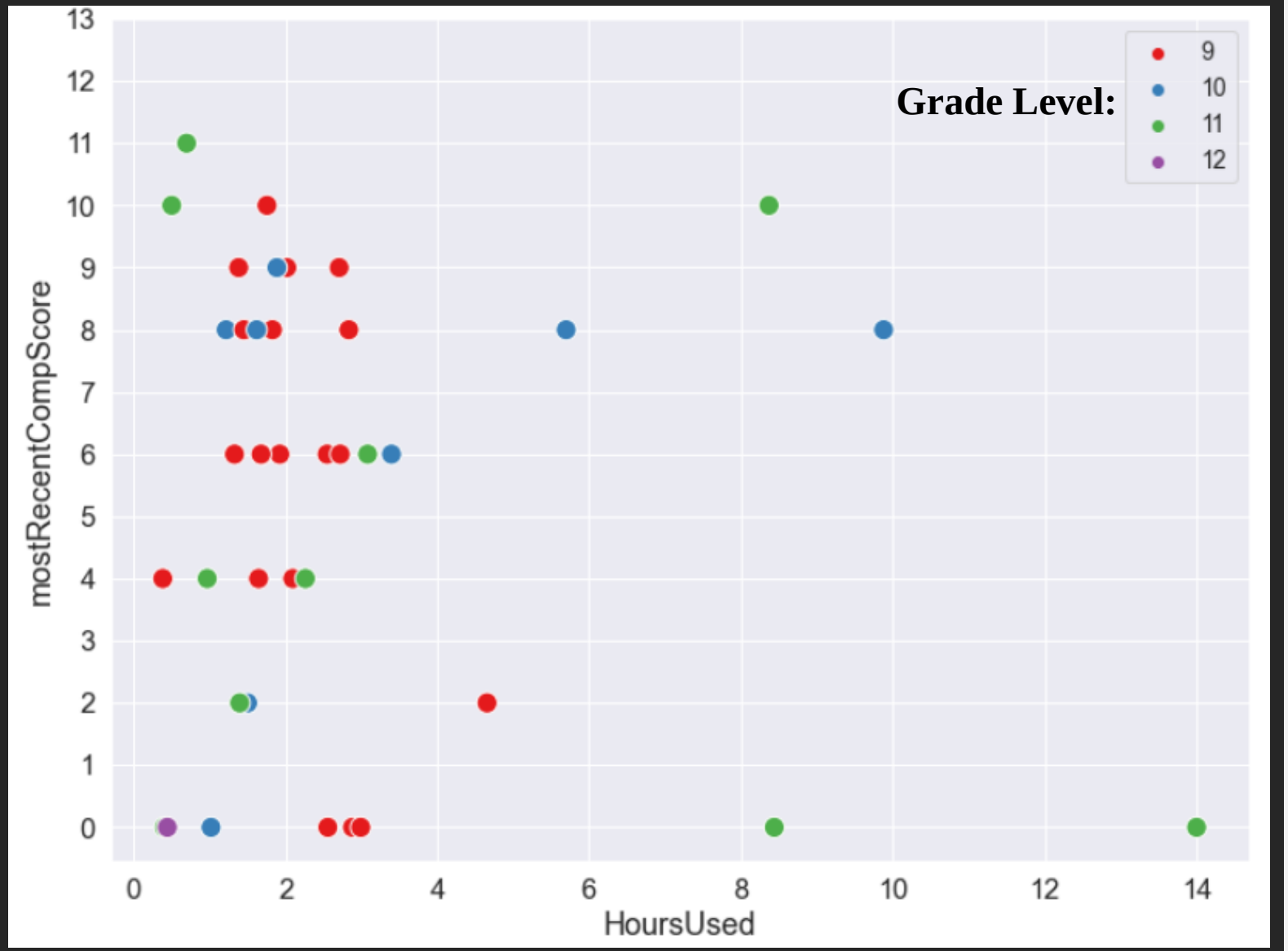
Total Students= 236

FINDINGS:

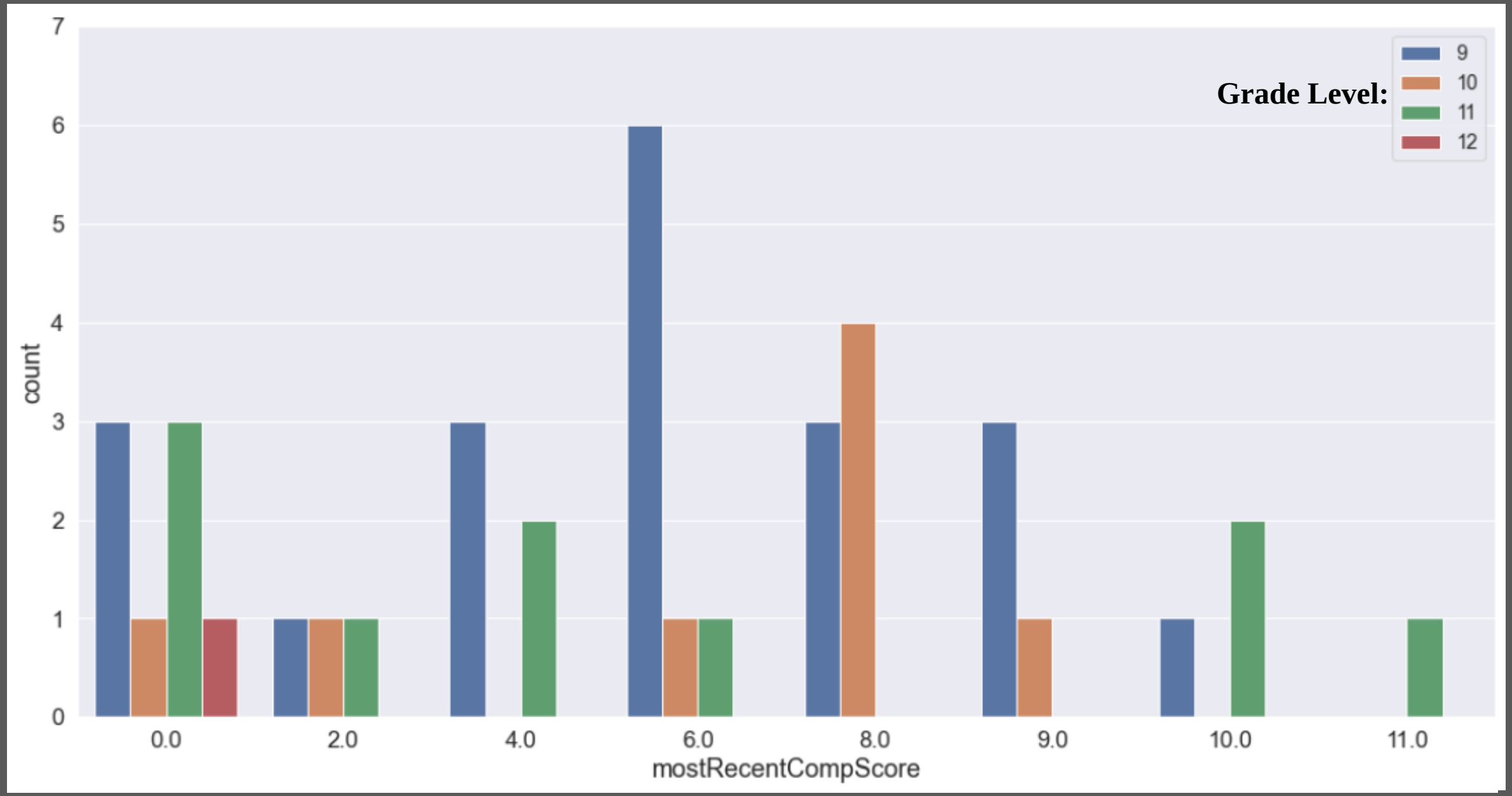
However, a **handful number of students** have SCORES on ‘**MostRecentScore**’ and also have used **SOFTWARE** for several of Hours

Total Students= 39

	mostRecentCompScore	HoursUsed
count	39.000000	39.000000
mean	5.307692	2.817607
std	3.532956	2.841187
min	0.000000	0.382500
25%	2.000000	1.390139
50%	6.000000	1.886111
75%	8.000000	2.858472
max	11.000000	14.005000



- Students with **NO** ‘Starting Software Date (**Test Date**)’ also have’ **NO ‘ComP Score**
- But, **39** Students have **mostRecentCompScore** and have used **Software** for couple of hours
- The next **Bar Graph** shows the Most Recent Scores of same students by Grade Level


Grade Level:
9
10
11
12
count
0
1
2
3
4
5
6
7
0.0
2.0
4.0
6.0
8.0
9.0
10.0
11.0
mostRecentCompScore