MAKERERE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATION SCIENCES

DATA ANALYSIS REPORT ON PLE RESULTS 2010 - 2015

GROUP N

NAME	REGISTRATION NUMBER	STUDENT NUMBER	EMAIL
RASHIDAH MAGEZI TUMUKUNDE	19/U/8741/EVE	1900708741	tumukunderashidahmagez@gmail.com
KIBALAMA TIMOTHY	19/U/8373/EVE	1900708373	timkibalama@gmail.com
MWESIGWA JOSHUA	19/U/8883/EVE	1900708883	joshuamj59@gmail.com
DDAMBA MAHAD	19/U/8192/EVE	1900708192	mahdmccall03@gmail.com
ASIIMWE MARIA	19/U/8980/EVE	1900708980	mariaasiimwe59@gnail.com

Table of Contents

1. INTRODUCTION

The data set provided indicates results of candidates for PLE of years 2010 to 2015. The results are listed by year, school and district.

It shows the performance percentage of pupils that passed in a particular division with in the years 2010 to 2015. The data set also shows schools arranged by districts and the performance divided into gender for each school in each year.

We used a number of tools to critically analyze the data including Jupyter Notebook and python package manager, PIP to install the necessary libraries like Pandas, Matplotlib and numpy.

1.1. A BRIEF INTRODUCTION ABOUT COLUMNS IN THE DATASET

1.1.1. District

This column contains the districts in which schools are found. It was used to sort the data as it was a suitable candidate for sorting. Also districts with the highest performance and those with the lowest were obtained using this column.

1.1.2. School

Names of schools were obtained from this column especially those with great performance over the years.

1.1.3. Gender % DIV columns such as FEMALE % DIV 1

These columns hold the percentages per grade of each gender. This helped identify the performance of boys and girls in each school per year.

1.1.4. Total candidates

This column showed the total number of candidates in a school in a particular year. This column enabled us get the average number of candidates that sat for exams in a specific year.

1.1.5. Male candidates

Shows the number of male candidates that sat for the exams per school in a year.

1.1.6. Female candidates

Shows the number of female candidates that sat for the exams per school in a year.

1.1.7. % DIV columns such as % DIV 1

The percentage of candidates that passed in a given division represented as a partition of the total candidates of a given school. These columns show the portion of candidates, male or female that passed in a given division. These were instrumental in determining the performance of candidates in a particular school and that of males and females in all the districts.

1.1.8. Summary of the dataset showing columns and their datatypes.

COLUMN	DATA TYPE		COLUMN	DATA TYPE
YEAR	int64		FEMALE % DIV2	float64
DISTRICT	object		FEMALE % DIV3	float64
SCHOOL	object		FEMALE % DIV4	float64
TOTAL CANDIDATES	int64		FEMALE % U (%)	float64
% DIV 1	float64		FEMALE % X	float64
% DIV 2	float64		MALE CANDIDATES	float64
% DIV 3	float64		MALE % DIV1	float64
% DIV 4	float64		MALE % DIV2	float64
% U	float64		MALE % DIV3	float64
% X	float64		MALE % DIV4	float64
FEMALE CANDIDATES	float64		MALE % U	float64
FEMALE % DIV1	float64		MALE % X	float64

2. OBJECTIVES

1) To get total number of candidates, males, females, schools and districts in a given year
2) To get average number of students in schools
3) To get average number of students in districts
4) To get number of students in a particular grade in a particular year
5) To get number of males and females per grade in a particular year
6) To get top schools in a particular year
7) To get last schools in a particular year
8) To get top districts in particular year
9) To get last districts in a particular year
10) To compare performance between males and females in a particular year
11) To compare performance per grade in a particular year
12) To get which grade most candidates got in a particular year
13) To get which grade fewest candidates got in a particular year
14) To get which grade most males got in a particular year
15) To get which grade fewest males got in a particular year
16) To get which grade most females got in a particular year
17) To get which grade fewest females got in a particular year

18) To relate students who got division 1 with the total
19) To compare performance from 2010 to 2015.
20) To obtain attendance of candidates who sat over years.
21) To obtain attendance of males and females who sat over the years.
22) To obtain schools that have appeared at least thrice among top 20 from 2010 to 2015.
23) To obtain districts that have appeared at least 5 times among top 20 top 20 from 2010 to 2015.
24) To compare percentage of students who passed and failed from 2010 to 2015.
25) To compare percentage of students who passed and failed from 2010 to 2015 considering gender.
26) To predict the number of candidates who will pass if a certain number of them seat.
27) To obtain statistical representation of entire data.

3. COLUMNS ANALYZED AND REASONS FOR ANALYSIS

3.1. Year

We used the year column to group data by year such that performance and attendance of candidates for each year can be derived.

3.2. District

We used the district column to derive the top and worst performing districts in each year and those that appeared most among the top 20 in all years.

3.3. School

We used the school column to derive the top and worst performing schools in each year. We also used it to get the schools that appeared most among top 20 in all years.

3.4. Total candidates

We used it to plot attendance of candidates from 2010 to 2015.

We also used it to obtain number of candidates who obtained a specific grade by calculating its values with grade percentages.

3.5. Male candidates

We used its values to obtain males who got a specific grade by calculating its values together with male grade percentages.

We also used its values together with female candidates to verify that the total candidates have a true value.

3.6. Female candidates

We used its values to obtain females who got a specific grade by calculating its values together with female grade percentages.

We also used its values together with male candidates to verify that the total candidates have a true value.

3.7. %DIV columns (%DIV1, %DIV2, %DIV3, %DIV4, %U, %X)

These columns were used to determine candidates who got each grade per school per year and thereafter their values used to plot bar graph of candidates who passed in each grade per year.

3.8. GENDER % DIV columns (such as MALE % DIV 1)

These columns were used to determine candidates (gender wise) who got each grade per school per year and thereafter their values used to plot bar graph of candidates who passed in each grade per year.

4. PROCESSES AND TECHNIQUES

4.1. SETTING DATA FOR ANALYSIS

4.1.1. Importing required libraries

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import math
from scipy import stats
plt.style.use('seaborn')
```

4.1.2. Reading the CSV file

We read the file using pandas read_csv() function.

```
ple = pd.read_csv("PleResults2010-2015.csv")
type(ple)
```

```
pandas.core.frame.DataFrame
```

4.1.3. Creating a class to contain common methods for different years

A YearlyData class was created to hold all the methods or functions that's are to be used to analyze all the years performance patterns henceforth particular year instances or objects were generated from this class.

```
ple2010 = YearlyData(2010)
ple2011 = YearlyData(2011)
ple2012 = YearlyData(2012)
ple2013 = YearlyData(2013)
ple2014 = YearlyData(2014)
ple2015 = YearlyData(2015)
```

4.1.4. Creating new columns by using assign method

The div1, div2, div3, div4, divU, divX, female div1, female div2, female div3, female div4, female divU, female divX, male div1, male div2, male div3, male div4, male divU, male divX columns were added to the data set to aid the calculation of total candidates per grade since percentages could not deliver what was required.

```
# adding number of students, females and males per grade as new columns
def add_div_columns(data):
    # for all
    div = ['% DIV 1', '% DIV 2', '% DIV 3', '% DIV 4', '% U', '% X']
    for i in range(len(div)):
        data = data.assign(p=(ple[div[i]]*ple['TOTAL CANDIDATES']/100).round().astype(int))
        if i < 4:
            column_name = 'div'+str(i+1)
        elif i == 4:
            column_name = 'divU'
        else:
            column_name = 'divX'
        data = data.rename(columns={'p': column_name})
    # for female
    female_div = ['FEMALE % DIV1 ', 'FEMALE % DIV2', 'FEMALE % DIV3', 'FEMALE % DIV4', 'FEMALE % U (%)', 'FEMALE % X ']
    for i in range(len(female_div)):
        data = data.assign(p=(ple[female_div[i]]*ple['FEMALE CANDIDATES']/100).round().astype(int))
        if i < 4:
            column_name = 'female div'+str(i+1)
        elif i == 4:
            column_name = 'female divU'
        else:
            column_name = 'female divX'
        data = data.rename(columns={'p': column_name})
```

```
    # for male
    male_div = ['MALE % DIV1 ', 'MALE % DIV2', 'MALE % DIV3', 'MALE % DIV4 ', 'MALE % U ', 'MALE % X']
    for i in range(len(male_div)):
        data = data.assign(p=(ple[male_div[i]]*ple['MALE CANDIDATES']/100).round().astype(int))
        if i < 4:
            column_name = 'male div'+str(i+1)
        elif i == 4:
            column_name = 'male divU'
        else:
            column_name = 'male divX'
        data = data.rename(columns={'p': column_name})
```

From the newly created columns, further critical analysis of the dataset was done and the graphical representation of the data were drawn.

TECHNIQUES

4.2. DATA CLEANING

4.2.1. Checking and removing duplicates

The duplicated() and sum() functions were used to find if there were any duplicate entries in the dataset and the drop_duplicates() function was used to eliminate the duplicated entries in the dataset

```
#checking for rows with duplicate data
print('There are',ple.duplicated().sum(),'duplicate rows.')
```

```
There are 139 duplicate rows.
```

```
# removing the duplicate rows
ple.drop_duplicates(inplace = True)
print('There are now',ple.duplicated().sum(),'duplicate rows.')
```

```
There are now 0 duplicate rows.
```

4.2.2. Checking for null cells and filling empty cells

The isnull() function was used to find and empty entries in the dataset and the fillna() function was used to fill the empty entries with zeros.

Before filling nulls,

```
# checking for number of cells with null values per column
ple.isnull().sum()
```

```
YEAR                    0
DISTRICT                0
SCHOOL                  0
TOTAL CANDIDATES        0
% DIV 1                 0
% DIV 2                 0
% DIV 3                 0
% DIV 4                 0
% U                     0
% X                     0
FEMALE CANDIDATES     155
FEMALE % DIV1         206
FEMALE % DIV2         206
FEMALE % DIV3         206
FEMALE % DIV4         206
FEMALE % U (%)        206
FEMALE % X            206
MALE CANDIDATES       185
MALE % DIV1           256
MALE % DIV2           256
MALE % DIV3           256
MALE % DIV4           256
MALE % U              256
MALE % X              256
dtype: int64
```

After filling nulls,

```
# replacing null values with zero since all columns with null values are numeric
ple.fillna(0, inplace = True)
# checking for number of cells with null values per column
ple.isnull().sum()
```

```
YEAR                  0
DISTRICT              0
SCHOOL                0
TOTAL CANDIDATES      0
% DIV 1               0
% DIV 2               0
% DIV 3               0
% DIV 4               0
% U                   0
% X                   0
FEMALE CANDIDATES     0
FEMALE % DIV1         0
FEMALE % DIV2         0
FEMALE % DIV3         0
FEMALE % DIV4         0
FEMALE % U (%)        0
FEMALE % X            0
MALE CANDIDATES       0
MALE % DIV1           0
MALE % DIV2           0
MALE % DIV3           0
MALE % DIV4           0
MALE % U              0
MALE % X              0
dtype: int64
```

4.2.3. Converting data to numeric

The numeric() method was used to ensure that all the columns that are supposed to hold numeric values and actually contained numerals in the dataset before proceeding with the analysis

```
#converting the data to numeric form
def numeric(*columns):
    for column in columns:
        ple[column] = pd.to_numeric(ple[column], errors='coerce')


numeric('YEAR', 'TOTAL CANDIDATES', '% DIV 1', '% DIV 2', '% DIV 3', '% DIV 4',
        'FEMALE % DIV1 ', 'FEMALE % DIV2', 'FEMALE % DIV3', 'FEMALE % DIV4', 'F
        'MALE CANDIDATES', 'MALE % DIV1 ', 'MALE % DIV2', 'MALE % DIV3', 'MALE
```

4.2.4. Correcting data

To ensure that all the columns and rows in our dataset actually contained the correct consistent values during analysis, the check_consistency() function was used to check for consistency of all the related columns after each correction was made for example when total candidates were not equal to sum of female and male candidates.

```
def check_consistency():
    # getting total number of candidates
    total = ple['TOTAL CANDIDATES'].sum()

    # getting total number of males
    male_total = ple['MALE CANDIDATES'].sum()

    # getting total number of females
    female_total = ple['FEMALE CANDIDATES'].sum()

    # getting total number of candidates in each grade
    div_totals = ple[['div1','div2','div3','div4','divU','divX']].sum()

    # getting total number of males per div
    male_div_totals = ple[['male div1','male div2','male div3','male div4','male divU','male divX']].sum()

    # getting total number of females per div
    female_div_totals = ple[['female div1','female div2','female div3','female div4','female divU','female divX']].sum()

    if (total==(male_total+female_total) and total==div_totals.sum() and male_total==male_div_totals.sum() and female_total==fema
        print('All related columns are now consistent.')
    else:
        print('Is total number of candidates equal to sum of males and females?',total==(male_total+female_total))
        print('Is total number of candidates equal to sum of candidates in grades?',total==div_totals.sum())
        print('Is total number of males equal to sum of males in grades?',male_total==male_div_totals.sum())
        print('Is total number of females equal to sum of females in grades?',female_total==female_div_totals.sum())
```

Correcting male candidate column values and thereafter checking for consistency again

```
# Getting rows where total candidates is not equal to sum of males and females
wrong_total_rows = ple.loc[ple['TOTAL CANDIDATES'] != (ple['MALE CANDIDATES']+ple['FEMALE CANDIDATES'])]

# So, we try to correct the male total and their grade and check again
for index in wrong_total_rows.index:
    ple.at[index,'MALE CANDIDATES'] = ple.at[index,'TOTAL CANDIDATES']-ple.at[index,'FEMALE CANDIDATES']
    ple.at[index,'male div1'] = (ple.at[index,'MALE % DIV1 ']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)
    ple.at[index,'male div2'] = (ple.at[index,'MALE % DIV2']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)
    ple.at[index,'male div3'] = (ple.at[index,'MALE % DIV3']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)
    ple.at[index,'male div4'] = (ple.at[index,'MALE % DIV4 ']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)
    ple.at[index,'male divU'] = (ple.at[index,'MALE % U ']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)
    ple.at[index,'male divX'] = (ple.at[index,'MALE % X']*ple.at[index,'MALE CANDIDATES']/100).round().astype(int)

#We check for consistency again
check_consistency()
```

```
Is total number of candidates equal to sum of males and females? True
Is total number of candidates equal to sum of candidates in grades? True
Is total number of males equal to sum of males in grades? False
Is total number of females equal to sum of females in grades? True
```

5. VISUALIZATIONS

```
#printing the top schools for a given year
ple2010.print_top(5)
```

```
THE TOP 5 SCHOOLS IN 2010 WERE:
```

	YEAR	DISTRICT	SCHOOL	TOTAL CANDIDATES	% DIV 1	% DIV 2	% DIV 3	% DIV 4	% U	% X	...	female div3	female div4	female divU	female divX	male div1	male div2	male div3	male div4	male divU
0	2010	KABAROLE	EXCEL PRIMARY SCHOOL,RWIMI	23	100.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	13	0	0	0	0
1	2010	KALUNGU	SACRED HEART P/S, KYAMUSANSALA	40	100.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	0	0	0
2	2010	KAMPALA	KING FAHAD ISLAMIC PRI. SCHOOL	3	100.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	3	0	0	0	0
3	2010	KAMPALA	MAKINDYE JUNIOR SCHOOL	1	100.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	1	0	0	0	0
4	2010	KAMPALA	WATERFORD P/S NAJJANANKUMBI	19	100.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	12	0	0	0	0

5 rows × 42 columns

Figure 1 Showing top 5 schools in 2010

```
#printing the top  districts
ple2010.print_top_districts(5)
```

```
THE TOP 5 DISTRICTS IN 2010 WERE:
```

DISTRICT	TOTAL CANDIDATES	div1	div2	div3	div4	divU	divX	%div1	%div2	%div3	%div4	%divU	%divX
MASAKA M/C	1894	902	761	123	34	38	36	48.0	40.0	6.0	2.0	2.0	2.0
MBARARA M/C	2375	985	1264	75	19	9	23	41.0	53.0	3.0	1.0	0.0	1.0
KABALE M/C	1474	604	708	98	23	7	34	41.0	48.0	7.0	2.0	0.0	2.0
FORTPORTAL M/C	1334	529	760	20	3	2	20	40.0	57.0	1.0	0.0	0.0	1.0
BUSHENYI M/C	1301	489	607	109	32	45	19	38.0	47.0	8.0	2.0	3.0	1.0

Figure 2 Showing top 5 districts in 2010

```
# plotting chart for performance per gender in particular year
ple2010.draw_gender_grade_chart()
```

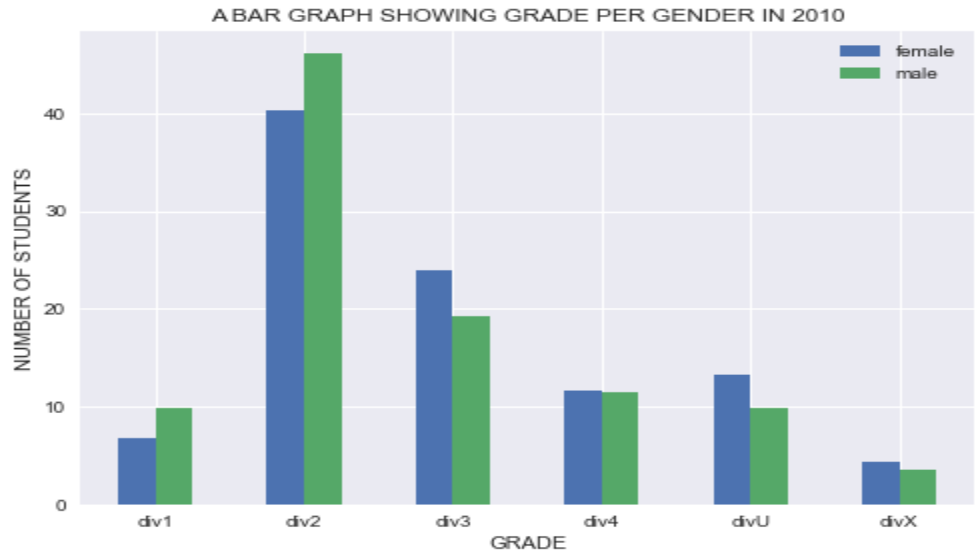

Figure 3 showing grades for both genders in 2010

```
# showing percentage of grades in a particular year
ple2010.draw_grade_chart()
```

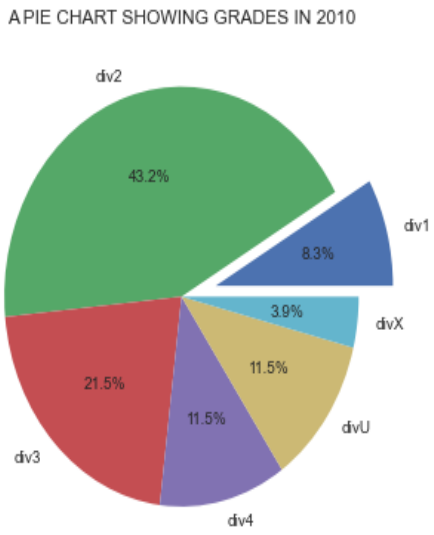

Figure 4 showing performance of candidates in 2010

```
# relate students who got division 1 with the total
ple2010.div1_scatter()
```

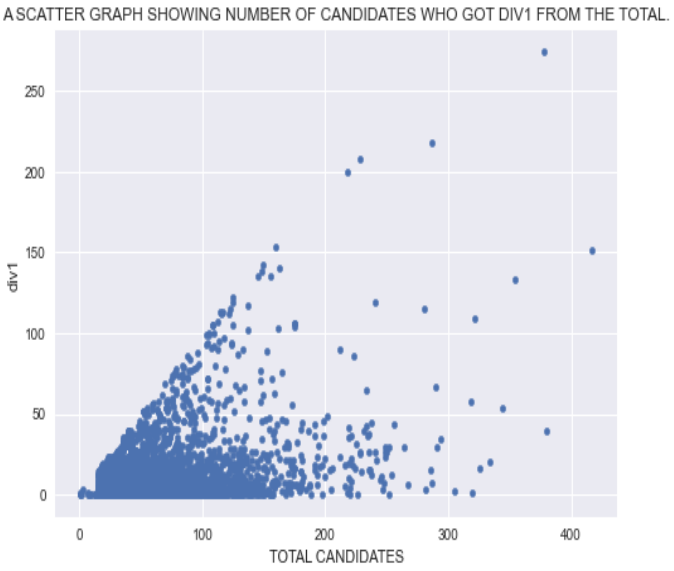

Figure 5 showing candidates who got div1 from the total

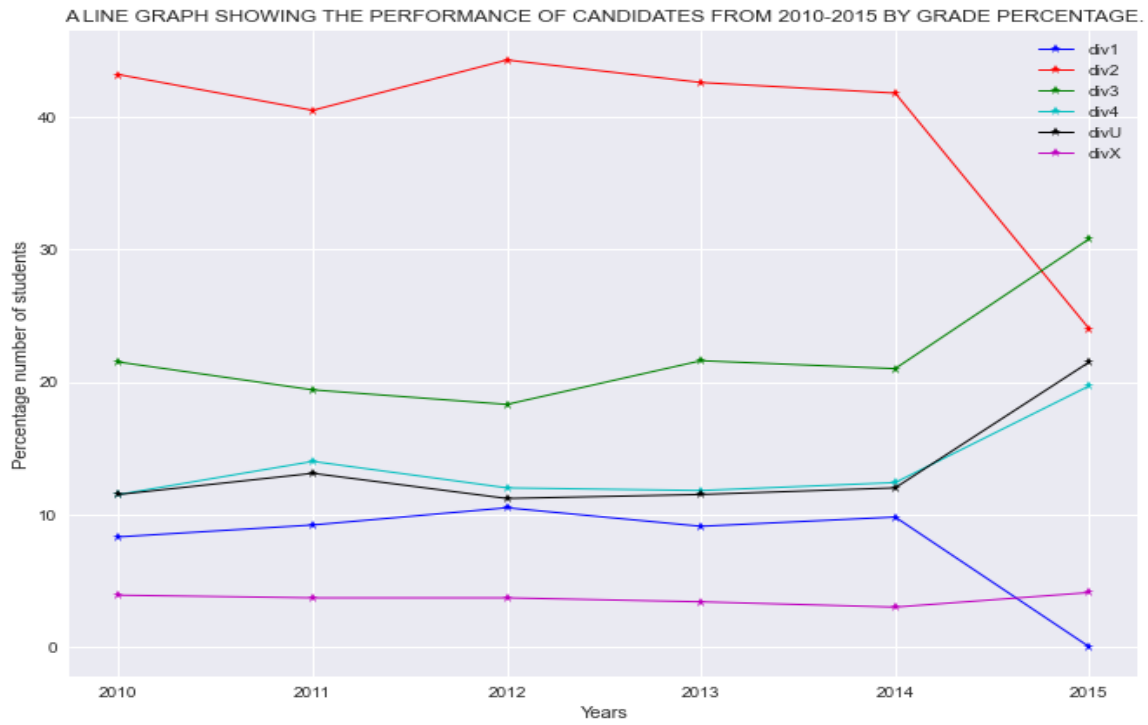

Figure 6 showing performance of students from 2010 to 2015

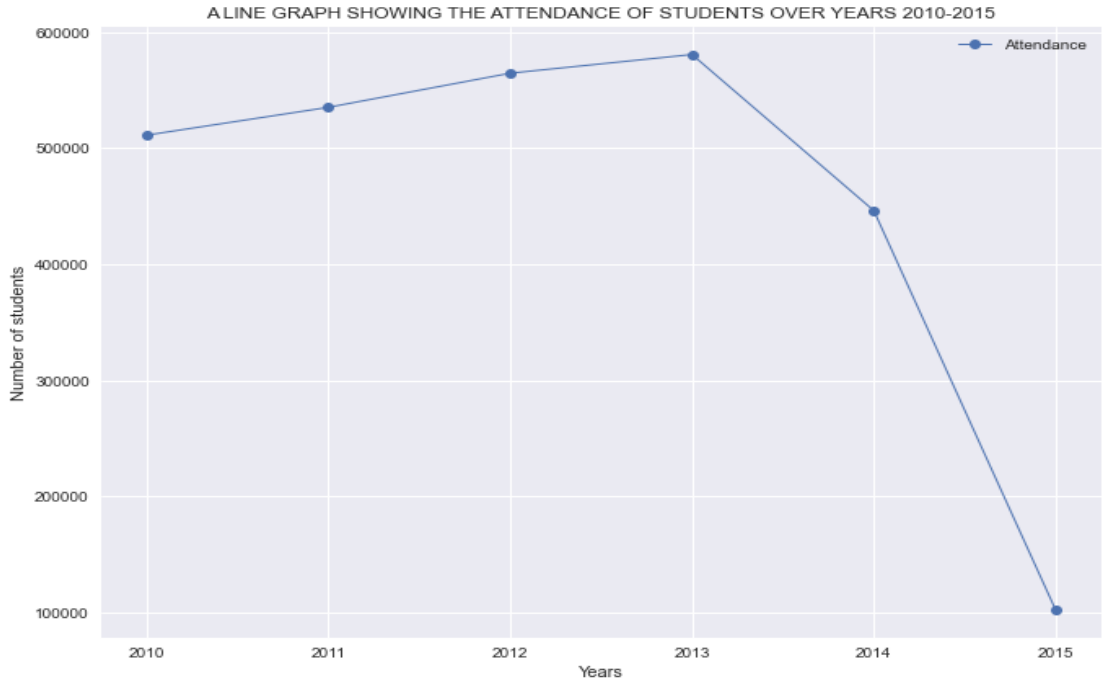

Figure 7 showing attendance of candidates from 2010 to 2015

[41]:

SCHOOL	num
VICTORY LEARNING PRI. SCHOOL	4
K.Y DAY AND BOARDING P/S	3
VILLA ROAD PRIMARY SCHOOL	3
KABOJJA JUNIOR PRIMARY SCHOOL	3
ST.MARYS IMMACULATE VILLA P/S	3
PARENTAL CARE SCHOOL	3
EXCEL PRIMARY SCHOOL,RWIMI	3

Figure 8 top schools and times they appeared from 2010 to 2015

:

DISTRICT	TOTAL CANDIDATES	num
MASINDI M/C	7432	5
SHEEMA	21710	5
MUKONO M/C	17335	5
MBALE M/C	10606	5
MASAKA M/C	8404	5
KAMPALA	133263	5

Figure 9 top districts in all the years

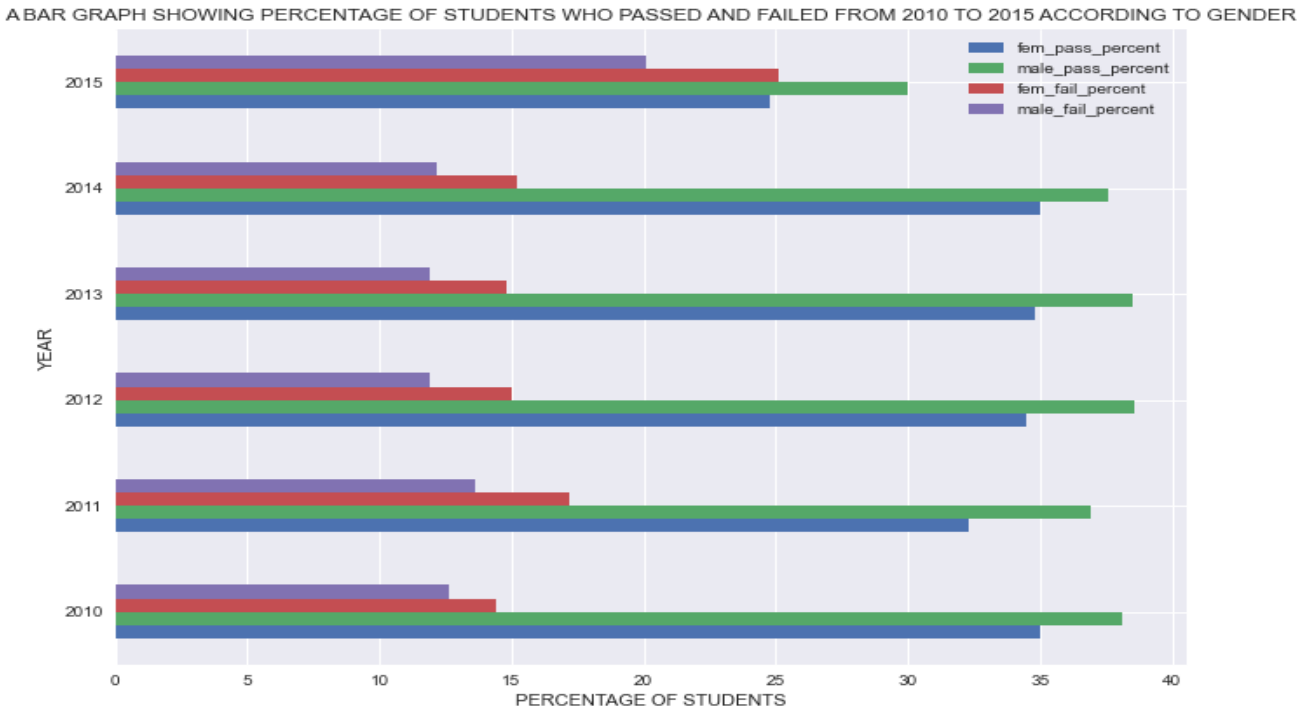

Figure 10 showing students who passed according to gender

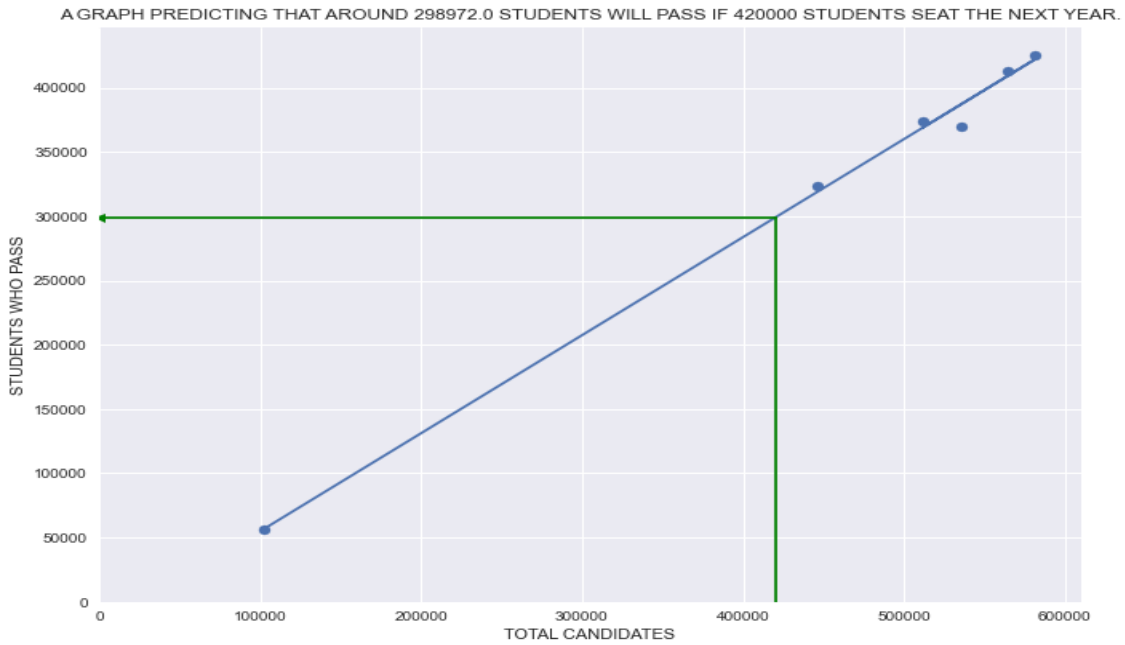

Figure 11 showing prediction of candidates who will pass if a certain number seat

6. DEDUCTIONS

Following the analysis done on the data set of PLE results 2010 to 2015, a number of observations were made and subsequent deductions were noted as follows;

- From the year 2010 to 2013, there was a gradual increase in the number of candidates who sat for PLE.
- From the year 2013 to 2014, there was a gradual decrease in the number of candidates who sat for PLE.
- From 2014 to 2015, there was a rapid decrease in the number of candidates who sat for PLE.
- There were more male candidates compared to female candidates in all the years.
- Most candidates passed in the division 2 over all the years except for 2015 where most candidates passed in division 3.
- There was no correlation between the data from 2015 and the rest of the years.
- There was a higher percentage pass than percentage fail over all the 6 years.
- Boys performed better than girls over all the 6 years.
- In 2015, there wasn't any division 1 and most of candidates got division X.
- Generally, most of the candidates qualified for secondary education in all the 6 years.

7. CONCLUSION

- Boys generally performed more than girls over the years.
- Division 2 was the most obtained grade.
- More boys sat for exams than girls.

8. RECOMMENDATIONS

- The government should sensitize the community to takes their children to school.
- Incentives like bursaries should be put in place to keep students in school who dropout due to lack of finances.
- The government should facilitate the schools that have been performing poorly in the years 2010 to 2015 consecutively.

9. REFERENCES

Python for Data Analysis – Wes McKinney, 2012 O'Reilly Media, Inc.

https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html