Business Report on Advanced Statistics:

Client name: Great Learning.

Objective: To understand the effect of “TWO” ingredients A&B in their Medicine Compound. They had given us the Data and based on that Data they have asked few questions which are related to the ANOVA testing and based on the ANOVA test, they will have more information on their Ingredients.

In this Jupyter Notebook we will be solving the Questions which are asked by the client and based on the Answers we will be updating our Microsoft Word Report (report will only consist of Brief explanation and this notebook will have indetail source code and Outputs.

Lets Begin...

In [1]:

```
# We will be importing Basic Libraries:

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import os
```

executed in 24.4s, finished 02:21:21 2021-01-18

In [2]:

```
# we will pull up the Data set (Fever) given by the client.
# we will check the Data by pulling up top 10 rows.

fever = pd.read_csv (r'E:\Great Learning\Projects\ANOVA_EDA\Fever.csv')
fever.head(10)
```

executed in 268ms, finished 02:21:28 2021-01-18

Out[2]:

	A	B	Volunteer	Relief
0	1	1	1	2.4
1	1	1	2	2.7
2	1	1	3	2.3
3	1	1	4	2.5
4	1	2	1	4.6
5	1	2	2	4.2
6	1	2	3	4.9
7	1	2	4	4.7
8	1	3	1	4.8
9	1	3	2	4.5

In [7]:

```
# will give the number of Rows and Columns
fever.shape
```

executed in 14ms, finished 18:06:45 2021-01-12

Out[7]:

```
(36, 4)
```

In [9]:

```
# Total number of elements present in the Data set,except (columns)
fever.size
```

executed in 10ms, finished 18:07:33 2021-01-12

Out[9]:

```
144
```

In [12]:

```
# will provide Data types, any missing values
fever.info()
```

executed in 19ms, finished 18:08:22 2021-01-12

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 4 columns):
A            36 non-null int64
B            36 non-null int64
Volunteer    36 non-null int64
Relief       36 non-null float64
dtypes: float64(1), int64(3)
memory usage: 1.2 KB
```

In [13]:

```
fever.dtypes.value_counts()
```

executed in 17ms, finished 18:08:44 2021-01-12

Out[13]:

```
int64      3
float64    1
dtype: int64
```

In [14]:

```
# So we have all the Numerical Data in the Data set, no categorical data present.
```

executed in 8ms, finished 18:09:22 2021-01-12

In [15]:

```
# Lets check the Central Tendencies for every column which will give us some basic informat

fever.describe()
```

executed in 88ms, finished 18:10:11 2021-01-12

Out[15]:

	A	B	Volunteer	Relief
count	36.000000	36.000000	36.000000	36.000000
mean	2.000000	2.000000	2.500000	7.183333
std	0.828079	0.828079	1.133893	3.272090
min	1.000000	1.000000	1.000000	2.300000
25%	1.000000	1.000000	1.750000	4.675000
50%	2.000000	2.000000	2.500000	6.000000
75%	3.000000	3.000000	3.250000	9.325000
max	3.000000	3.000000	4.000000	13.500000

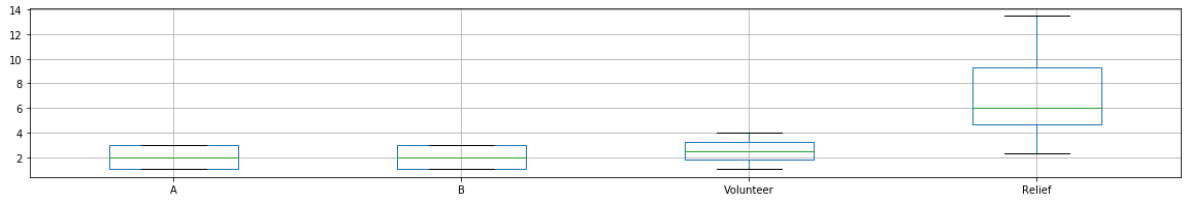
Based on the above measures, we can able to analyse that the Data is ditributed Normally (which we can plot and confirm)

Also there will be no Outlier's (which we can confirm in below plot).

In [16]:

```
fever.boxplot(figsize=(20,3));
```

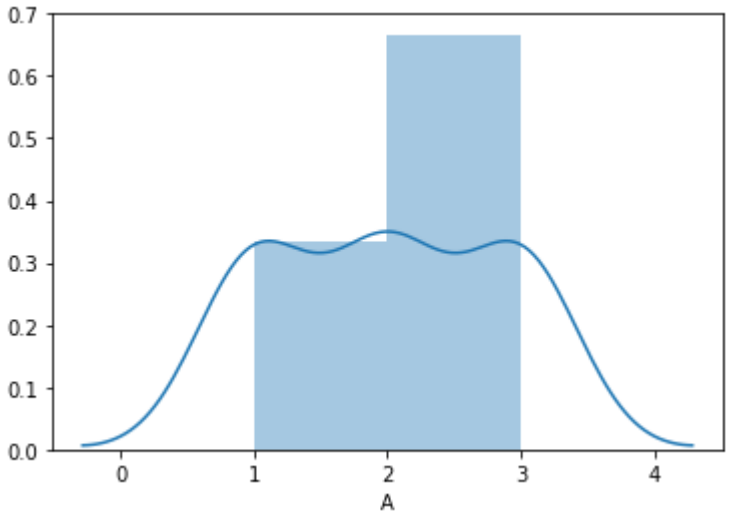
executed in 730ms, finished 18:12:14 2021-01-12



In [21]:

```
sns.distplot(fever['A']);
```

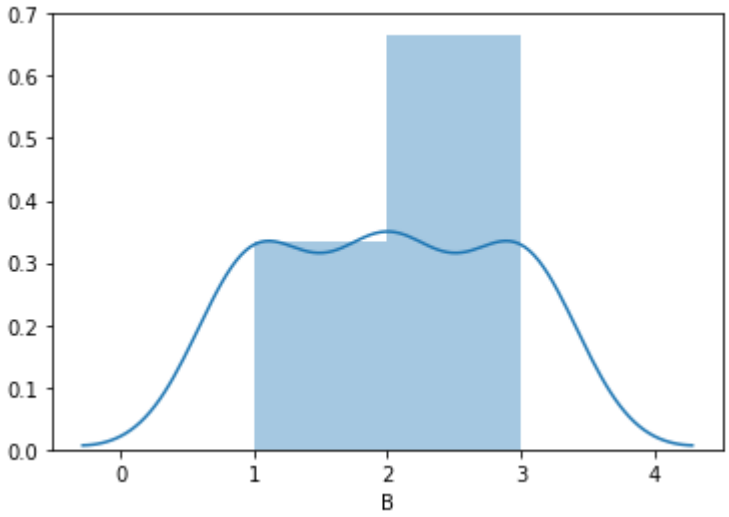
executed in 248ms, finished 18:13:35 2021-01-12



In [22]:

```
sns.distplot(fever['B']);
```

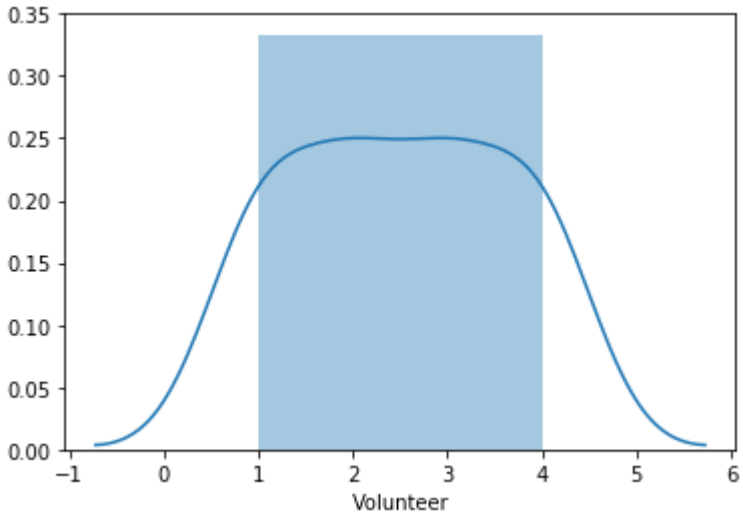
executed in 225ms, finished 18:14:02 2021-01-12



In [23]:

```
sns.distplot(fever['Volunteer']);
```

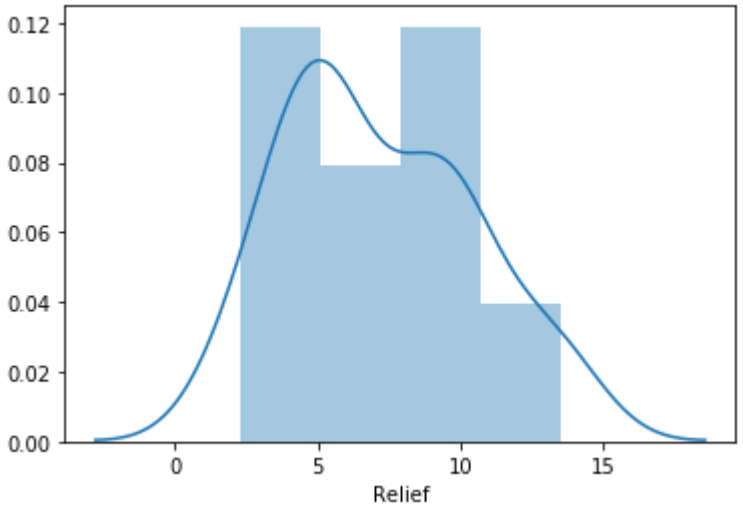
executed in 305ms, finished 18:14:23 2021-01-12



In [24]:

```
sns.distplot(fever['Relief']);
```

executed in 230ms, finished 18:14:46 2021-01-12



So by looking at all the above plots we proved that There are no Outliers and all Data is Normally Distributed.

In [25]:

```
# Lets check if there are any Missing Values and any duplicates.

fever.isna().sum()
```

executed in 12ms, finished 18:19:25 2021-01-12

Out[25]:

```
A            0
B            0
Volunteer    0
Relief       0
dtype: int64
```

In [27]:

```
dupe = fever.duplicated()
print('Number of duplicates in this Data are %d' % dupe.sum())
```

executed in 24ms, finished 18:20:28 2021-01-12

```
Number of duplicates in this Data are 0
```

So there are No Missing values and No Duplicates.

In [34]:

```
# Lets check if there are any non-int values.

fever ['A'].unique()
fever ['B'].unique()
```

executed in 14ms, finished 18:23:42 2021-01-12

Out[34]:

```
array([1, 2, 3], dtype=int64)
```

In [32]:

```
fever ['Volunteer'].unique()
```

executed in 20ms, finished 18:22:37 2021-01-12

Out[32]:

```
array([1, 2, 3, 4], dtype=int64)
```

In [33]:

```
fever ['Relief'].unique()
```

executed in 12ms, finished 18:22:54 2021-01-12

Out[33]:

```
array([ 2.4,  2.7,  2.3,  2.5,  4.6,  4.2,  4.9,  4.7,  4.8,  4.5,  4.4,
        5.8,  5.2,  5.5,  5.3,  8.9,  9.1,  8.7,  9. ,  9.3,  9.4,  6.1,
        5.7,  5.9,  6.2,  9.9, 10.5, 10.6, 10.1, 13.5, 13. , 13.3, 13.2])
```

So now we have analyse the given Data and we know that the Data is Clean and Appropriate for further ANOVA testing which is the Main Objective for the

Client.

In [ ]:

Basic Terminology in ANOVA testing.

#We perform ANOVA based on p-value (0.05) which means level of significance and the Hypothesis Testing will be decided based on this value (we will not go in too Technicality).

If my p value > 0.05 then we Accept Null Hypothesis (H0) and

If my p value < 0.05 then we Reject Null Hypothesis (H1).

For performing ANOVA testing we need to calculate the all the values which are:

1 SSA – Sum of Squares amongst group (xbar – xdoublebar) ^ 2 xdoublebar-Global Mean

2 SSW—Sum of Squares within group (x – xbar) ^ 2 xbar- Mean

3 SST—Sum f total squares (SST) (SSA + SSW) deg (1) – c-1,deg (2)- n-c

4 MSA- Mean Square Variance amongst group (SSA / c-1) c – Number of Columns

5 MSW- Mean Square Variance within group (SSW / n-c) n- Total Elements

6 MST- Mean squared total Variance (MSA +MSW)

7 Fstat – MSA / MSW and then we calculate the p value and check the level

Above terminology can be easily explained by its own labels and I will not go in too detail. This all are only for one time to understand the stake holders about the back-end calculations for doing the ANOVA test.

In [ ]:

Problem 1:

A research laboratory was developing a new compound for the relief of severe cases of hay fever. In an experiment with 36 volunteers, the amounts of the two active ingredients (A & B) in the compound were varied at three levels each. Randomization was used in assigning four volunteers to each of the nine treatments. The data on hours of relief can be found in the following .csv file: Fever.csv

1.1) State the Null and Alternate Hypothesis for conducting one-way ANOVA for both the variables ‘A’ and ‘B’ individually.

HYPOTHESIS FOR “A”

Null Hypothesis (H0): Active ingredient (A) in the compound have the SAME relief time on severe cases of Hay Fever.

Alternate Hypothesis (H1): At-least one of the Active ingredient (A) in the compound has the DIFFERENT/NOT SAME relief time on severe cases of Hay Fever.

HYPOTHESIS FOR "B"

Null Hypothesis (H0): Active ingredient (B) in the compound have the SAME relief time on severe cases of Hay Fever.

Alternate Hypothesis (H1): At-least one of the Active ingredient (B) in the compound has the DIFFERENT/NOT SAME relief time on severe cases of Hay Fever.

In [ ]:

```
```

In [ ]:

```
```

1.2) Perform one-way ANOVA for variable 'A' with respect to the variable 'Relief'. State whether the Null Hypothesis is accepted or rejected based on the ANOVA results.

Lets define the Null and Alternate Hypothesis.

Null Hypothesis (H0) : Ingredient A in the compound gives the SAME relief Time result.

Alternate Hypothesis (H1) :Atleast one of the Relief time result is different due the Ingredient A.

condition : p value > 0.05 then we will fail to reject Null Hypothesis.

In [37]:

```
# Lets start the calculations.
from statsmodels.formula.api import ols
from statsmodels.stats.anova import _get_covariance,anova_lm
```

executed in 1.15s, finished 23:53:03 2021-01-12

In [65]:

```
formula = 'Relief ~ (A)'
model = ols (formula, fever).fit()
aov_table = anova_lm(model)
print (aov_table)
```

executed in 40ms, finished 15:04:15 2021-01-13

```
            df   sum_sq     mean_sq          F        PR(>F)
A          1.0  212.415  212.415000  44.494409  1.175871e-07
Residual  34.0  162.315    4.773971        NaN           NaN
```

As we can see that the P-Value in this scenario is less than 0.05 Hence we can say that we Failed to ACCEPT Null Hypothesis, which means that statement: Ingredient A in the compound gives the SAME relief Time result is INCORRECT which concludes that Atleast one of the Relief time result is Different for Ingredient A in the compound.

In [ ]:

```
```

In [ ]:

```
```

1.3) Perform one-way ANOVA for variable 'B' with respect to the variable 'Relief'. State whether the Null Hypothesis is accepted or rejected based on the ANOVA results.

Lets define the Null and Alternate Hypothesis.

Null Hypothesis (H0) : Ingredient B in the compound gives the SAME relief Time result.

Alternate Hypothesis (H1) :Atleast one of the Relief time result is different due the Ingredient B.

condition : p value > 0.05 then we will fail to reject Hypothesis.

In [66]:

```
formula = 'Relief ~ (B)'
model = ols (formula, fever).fit()
aov_table = anova_lm(model)
print (aov_table)
```

executed in 51ms, finished 18:59:08 2021-01-13

```
            df   sum_sq     mean_sq          F    PR(>F)
B          1.0  113.535  113.535000  14.778958  0.000505
Residual  34.0  261.195    7.682206        NaN       NaN
```

As we can see that the P-Value in this scenario is less than 0.05 Hence we can say that we Failed to Accept Null Hypothesis, which means that statement : Ingredient B in the compound gives the SAME relief Time result is INCORRECT which concludes that Atleast one of the Relief time result is Different for Ingredient B in the compound.

In [ ]:

```
```

In [ ]:

```
```

1.4) Analyse the effects of one variable on another with the help of an interaction plot.

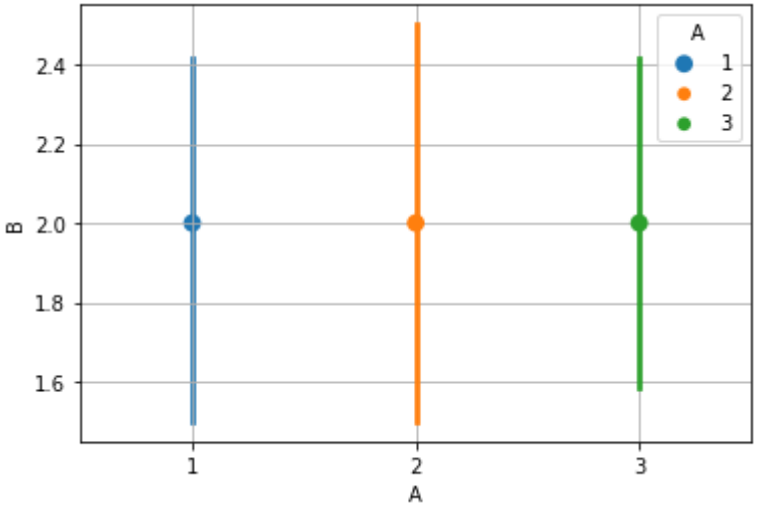
What is an interaction between two treatments? [hint: use the 'pointplot' function from the 'seaborn' function]

Below is the Interaction plot which shows that the Mean is the Same for all the Ingredients which is A and B, but the volume of quantity 2 in ingredient A is higher as compared to the ingredient B.

In [77]:

```
sns.pointplot (x = 'A' , y = 'B' , data=fever , hue='A');
plt.grid()
plt.show()
```

executed in 426ms, finished 23:21:31 2021-01-13



In [ ]:

In [ ]:

1.5) Perform a two-way ANOVA based on the different ingredients (variable 'A' & 'B') with the variable 'Relief' and state your results.

Lets define Null and Alternate Hypothesis.

Null Hypothesis (H0): Both the Ingredients A & B have the same Relief Time.

Alternate Hypothesis (H1): Atleast one of the ingredient (A&B) has the different Relief Time.

In [78]:

```
formula = 'Relief ~ (A) + (B)'
model = ols (formula, fever).fit()
aov_table = anova_lm(model)
print(aov_table)
```

executed in 46ms, finished 23:30:59 2021-01-13

```
            df   sum_sq     mean_sq           F        PR(>F)
A          1.0  212.415  212.415000  143.700185  1.437294e-13
B          1.0  113.535  113.535000   76.807196  3.952700e-10
Residual  33.0   48.780    1.478182         NaN           NaN
```

In [79]:

```
formula = 'Relief ~ C(A) + C(B)'
model = ols (formula, fever).fit()
aov_table = anova_lm(model)
print(aov_table)
```

executed in 76ms, finished 23:31:48 2021-01-13

```
            df  sum_sq     mean_sq           F        PR(>F)
C(A)       2.0  220.02  110.010000  109.832850  8.514029e-15
C(B)       2.0  123.66   61.830000   61.730435  1.546749e-11
Residual  31.0   31.05    1.001613         NaN           NaN
```

Above two results proves that Both the Ingredients not have the same Relief time , hence we FAILED TO ACCEPT NULL HYPOTHESIS (H0).

Even though the MEAN of both the ingredient is same and the quantity is also same but still the result relief time is different.

So Effect of Both the Ingredients (A&B) in the compound can not determine the Relief time and they are independent variables.

Lets try to investigate the other Variable which is Volunteer and see what exactly it is behaving on the Relief time.

In [ ]:

```

```

Lets Prove it by ANOVA TESTING.

Null Hypothesis (H0) : Volunteers for the Ingredients (A&B) are SAME for getting the RELIEF Result.

Alternate Hypothesis (H1) : At-Least one of the Volunteers for the Ingredients (A&B) is different for getting the RELIEF Result.

So we will be Performing ANOVA testing with the Variable Volunteer and Relief.

In [81]:

```
formula = 'Relief ~ Volunteer + (A) + (B)'
model = ols (formula,fever).fit()
aov_table = anova_lm(model)
print(aov_table)
```

executed in 83ms, finished 23:36:33 2021-01-13

```
             df      sum_sq     mean_sq           F        PR(>F)
Volunteer   1.0    0.000222    0.000222    0.000146  9.904416e-01
A           1.0  212.415000  212.415000  139.346268  3.375819e-13
B           1.0  113.535000  113.535000   74.480044  7.317462e-10
Residual   32.0   48.779778    1.524368         NaN           NaN
```

So above results shows that when we TEST both the ingredient with the Volunteer and Releif time, we

are getting the result in the favour of Volunteer and it is giving the result of 0.99 which is greater than 0.05.

NOW VOLUNTEER is the main Variable for the RELIEF TIME.

0.99 > 0.05

Hence we failed to REJECT the NULL Hypothesis (H0) which proves that Volunteers for the Ingredients (A&B) are SAME for getting the Relief time Result.

In [88]:

```
formula = 'Relief ~ Volunteer + (A) + (B)+ (A):(B):(Volunteer)'
model = ols (formula,fever).fit()
aov_table = anova_lm(model)
print(aov_table)
```

executed in 109ms, finished 13:46:32 2021-01-14

```
                 df      sum_sq     mean_sq           F        PR(>F)
Volunteer       1.0    0.000222    0.000222    0.000167  9.897851e-01
A               1.0  212.415000  212.415000  159.227463  9.467354e-14
B               1.0  113.535000  113.535000   85.106466  2.120119e-10
A:B:Volunteer   1.0    7.424694    7.424694    5.565592  2.479730e-02
Residual       31.0   41.355083    1.334035         NaN           NaN
```

So above result shows that Volunteer is the Critical Variable for Predicating the Relief Time,as assigning the number of volunteers for each ingredients is the Valuable input. And when we compare Both ingredeints A:B with Volunteer gives 0.0247973 is greater than 0.05 (0.02 > 0.05) which shows that Volunteering is always a predicatable or critical variable for Relief result.

In [ ]:

```
```

1.6) Mention the business implications of performing ANOVA for this particular case study

The most important factor for this case study is analyzed and which is Volunteer.

Business implications:

The ingredients in the compound are vary from start to end but the total amount in compound is the same which we see by looking at their MEAN's. When we check the Volunteers as soon as it gets assigned to both the ingredients the Relief time started to be predicting. So number of intake in ingredients increases with the Volunteers then the Relief time is also increasing. Lets check:

Per Volunteer Time Taken in Hours:

2.4

1.35

0.7666

0.625

As we can see if there is only ONE volunteer with the same amount of both the ingredients then the Relief time is 2.4 Hours and For TWO time is 1.35 Hours for Three time is 0.76 minutes and for Four is 0.625 minutes.

So if the same amount of quantity is distributed with number of people/volunteer then relief time is decreasing which shows that the same quantity of ingredient have maximum time when taken by one volunteer and same quantity of ingredient have minimum time when taken by four volunteers.

So we need to decrease the QUANTITY for INGREDIENTS A and B in the compound (like for making quantity of total 1 they are putting the calculations to make it 1 and now they have to reduce the calculation to make it less than 1) which will give the SAME relief time for EVERY Volunteer.

In [ ]:

```
```

In [ ]:

```
```

Problem 2:

The dataset Education - Post 12th Standard.csv is a dataset which contains the names of various colleges. This particular case study is based on various parameters of various institutions. You are expected to do Principal Component Analysis for this case study according to the instructions given in the following rubric. The data dictionary of the 'Education - Post 12th Standard.csv' can be found in the following file: Data Dictionary.xlsx.

In [5]:

```
# Lets define a data set and pull first 10 values to understand the Data.

edu = pd.read_csv(r'E:\Great Learning\Projects\ANOVA_EDA\Education+-+Post+12th+Standard.csv
```

executed in 48ms, finished 03:00:48 2021-01-18

In [6]:

```
edu.head()
```

executed in 32ms, finished 03:00:56 2021-01-18

Out[6]:

	Names	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
0	Abilene Christian University	1660	1232	721	23	52	2885	537	7440
1	Adelphi University	2186	1924	512	16	29	2683	1227	12280
2	Adrian College	1428	1097	336	22	50	1036	99	11250
3	Agnes Scott College	417	349	137	60	89	510	63	12960
4	Alaska Pacific University	193	146	55	16	44	249	869	7560

Lets perform some basic checks.

In [7]:

```
edu.shape
```

executed in 24ms, finished 03:01:13 2021-01-18

Out[7]:

```
(777, 18)
```

In [8]:

```
edu.size
```

executed in 20ms, finished 03:01:24 2021-01-18

Out[8]:

```
13986
```

In [9]:

```
edu.info()
```

executed in 24ms, finished 03:01:32 2021-01-18

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
Names          777 non-null object
Apps           777 non-null int64
Accept         777 non-null int64
Enroll         777 non-null int64
Top10perc      777 non-null int64
Top25perc      777 non-null int64
F.Undergrad    777 non-null int64
P.Undergrad    777 non-null int64
Outstate       777 non-null int64
Room.Board     777 non-null int64
Books          777 non-null int64
Personal       777 non-null int64
PhD            777 non-null int64
Terminal       777 non-null int64
S.F.Ratio      777 non-null float64
perc.alumni    777 non-null int64
Expend         777 non-null int64
Grad.Rate      777 non-null int64
dtypes: float64(1), int64(16), object(1)
memory usage: 109.4+ KB
```

In [10]:

```
edu.describe()
```

executed in 273ms, finished 03:01:49 2021-01-18

Out[10]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Unde
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	777.0(
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	855.2(
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	1522.4(
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	1.0(
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	95.0(
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	353.0(
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	967.0(
max	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31643.000000	21836.0(

In [12]:

```
edu.dtypes.value_counts()
```

executed in 36ms, finished 03:04:25 2021-01-18

Out[12]:

```
int64      16
float64     1
object      1
dtype: int64
```

In [13]:

```
edu.isnull().sum()
```

executed in 22ms, finished 03:07:42 2021-01-18

Out[13]:

```
Names          0
Apps           0
Accept         0
Enroll         0
Top10perc      0
Top25perc      0
F.Undergrad    0
P.Undergrad    0
Outstate       0
Room.Board     0
Books          0
Personal       0
PhD            0
Terminal       0
S.F.Ratio      0
perc.alumni    0
Expend         0
Grad.Rate      0
dtype: int64
```

In [16]:

```
du= edu.duplicated()
du.sum()
print('Number of Duplicates in this data set are %d' % du.sum())
```

executed in 24ms, finished 03:09:57 2021-01-18

```
Number of Duplicates in this data set are 0
```

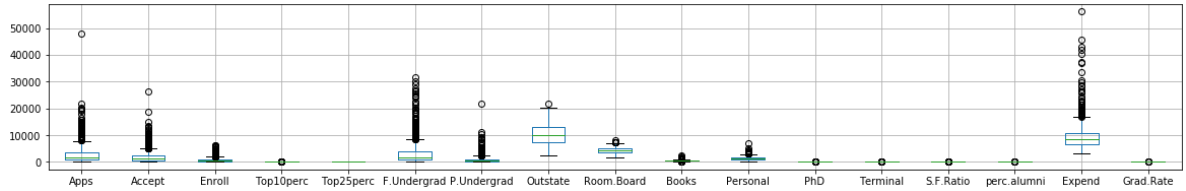
777 Rows and 18 Columns. There are 16 (Integer Values) , 1 (Float) and 1 (Object). There are no Missing values in Entire Data set. Number of Duplicates in this data set are 0.

Now we will check the Outliers:

In [17]:

```
edu.boxplot(figsize=(20,3));
```

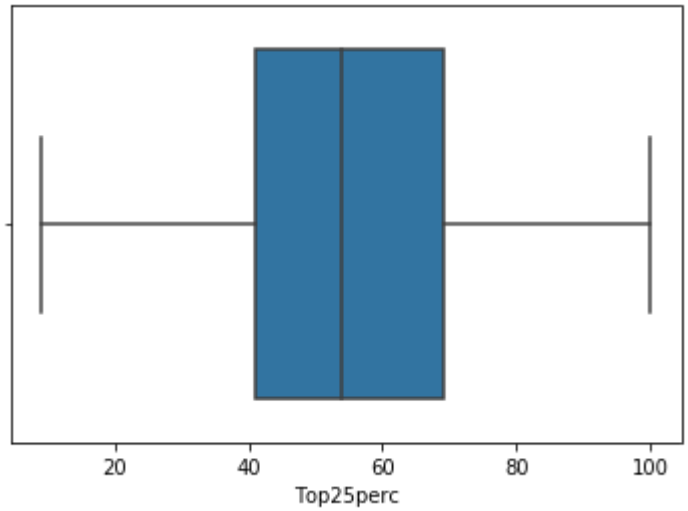
executed in 1.45s, finished 03:11:28 2021-01-18



In [18]:

```
sns.boxplot(edu['Top25perc']);
```

executed in 196ms, finished 03:12:04 2021-01-18



Except Top25perc All variables contains Outliers.

In [ ]:

```
# Here Dependent variable is the Grad.Rate.
```

In [ ]:

```

```

In [ ]:

```

```

2.1) Perform Exploratory Data Analysis [both univariate and multivariate analysis to be performed]. The inferences drawn from this should be properly documented.

First we will perfrom Univariate Analysis (used to study one variable in the data set) on all the Continous (Numerical Variables) except Names which is a (Categorical Column)

#so we will perform on 17 Variables.

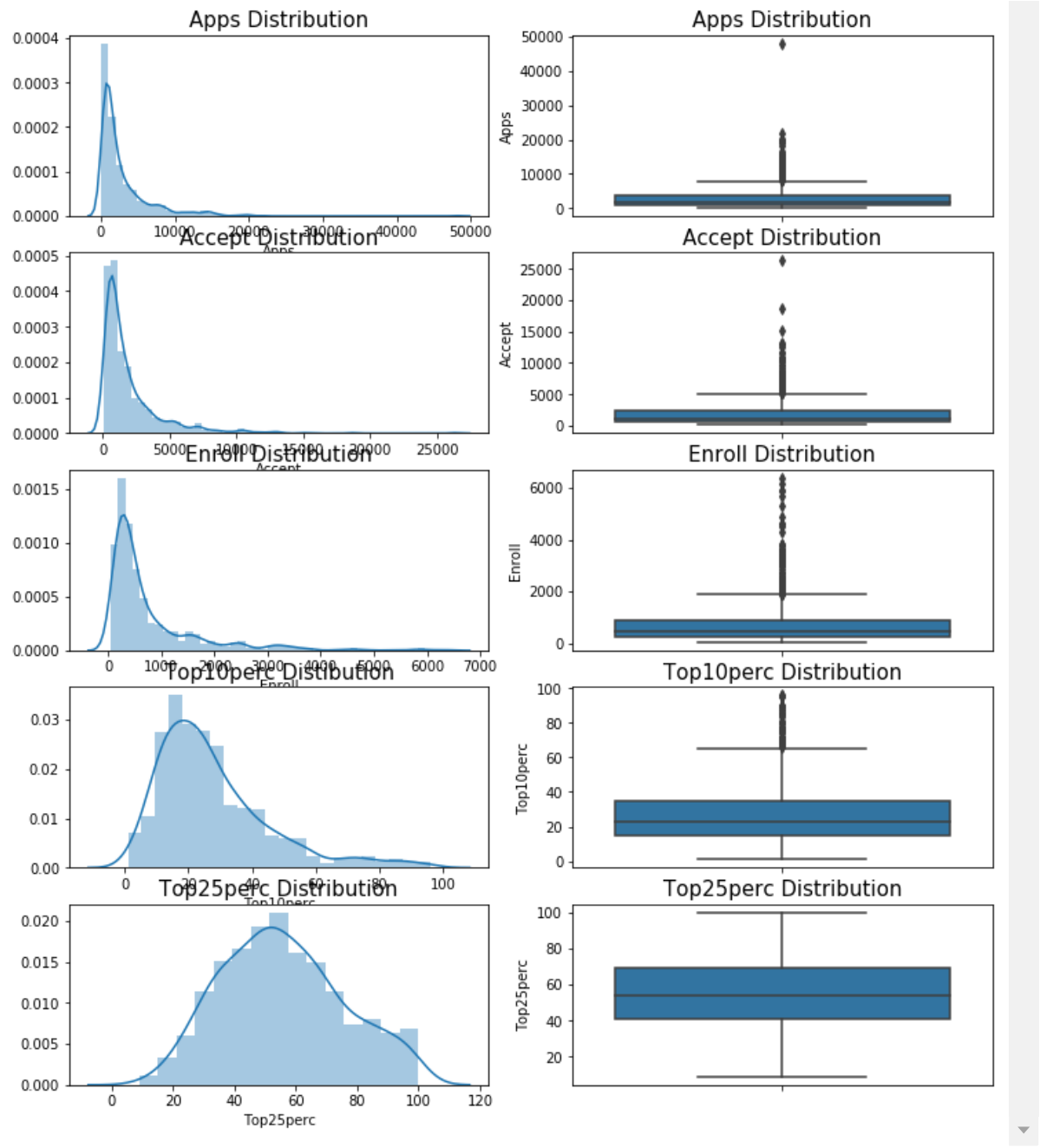
In [53]:

```
fig, axes = plt.subplots (nrows=5 , ncols=2)
fig.set_size_inches (12,14)

r = sns.distplot (edu ['Apps'] , ax=axes [0][0]);
r.set_title ('Apps Distribution',fontsize=15)
r = sns.boxplot (edu ['Apps'] ,orient='v', ax=axes[0][1]);
r.set_title ('Apps Distribution' , fontsize=15)

r = sns.distplot (edu ['Accept'] , ax=axes [1][0]);
r.set_title ('Accept Distribution' , fontsize=15)
r = sns.boxplot (edu ['Accept'] , orient='v', ax=axes[1][1]);
r.set_title ('Accept Distribution' , fontsize=15)

r = sns.distplot (edu ['Enroll'] , ax=axes[2][0]);
r.set_title ('Enroll Distribution' , fontsize = 15)
r = sns.boxplot (edu ['Enroll'] , orient= 'v' , ax=axes [2][1]);
r.set_title ('Enroll Distribution' , fontsize = 15)

r = sns.distplot (edu ['Top10perc'] , ax=axes [3][0]);
r.set_title ('Top10perc Distibution', fontsize=15)
r = sns.boxplot (edu ['Top10perc'], orient='v',ax=axes [3][1])
r.set_title ('Top10perc Distribution',fontsize=15)

r = sns.distplot (edu ['Top25perc'] , ax=axes[4][0]);
r.set_title ('Top25perc Distribution' , fontsize=15)
r = sns.boxplot (edu ['Top25perc'], orient='v', ax=axes [4][1]);
r.set_title ('Top25perc Distribution' ,fontsize=15)
```

executed in 3.38s, finished 16:43:16 2021-01-23

Out[53]:

```
Text(0.5, 1.0, 'Top25perc Distribution')
```



As we can see by above all variables have the right Skewed and have Outliers except Top25perc which is slightly Normally Distributed and No Outliers.

Lets check for next 5 variables.

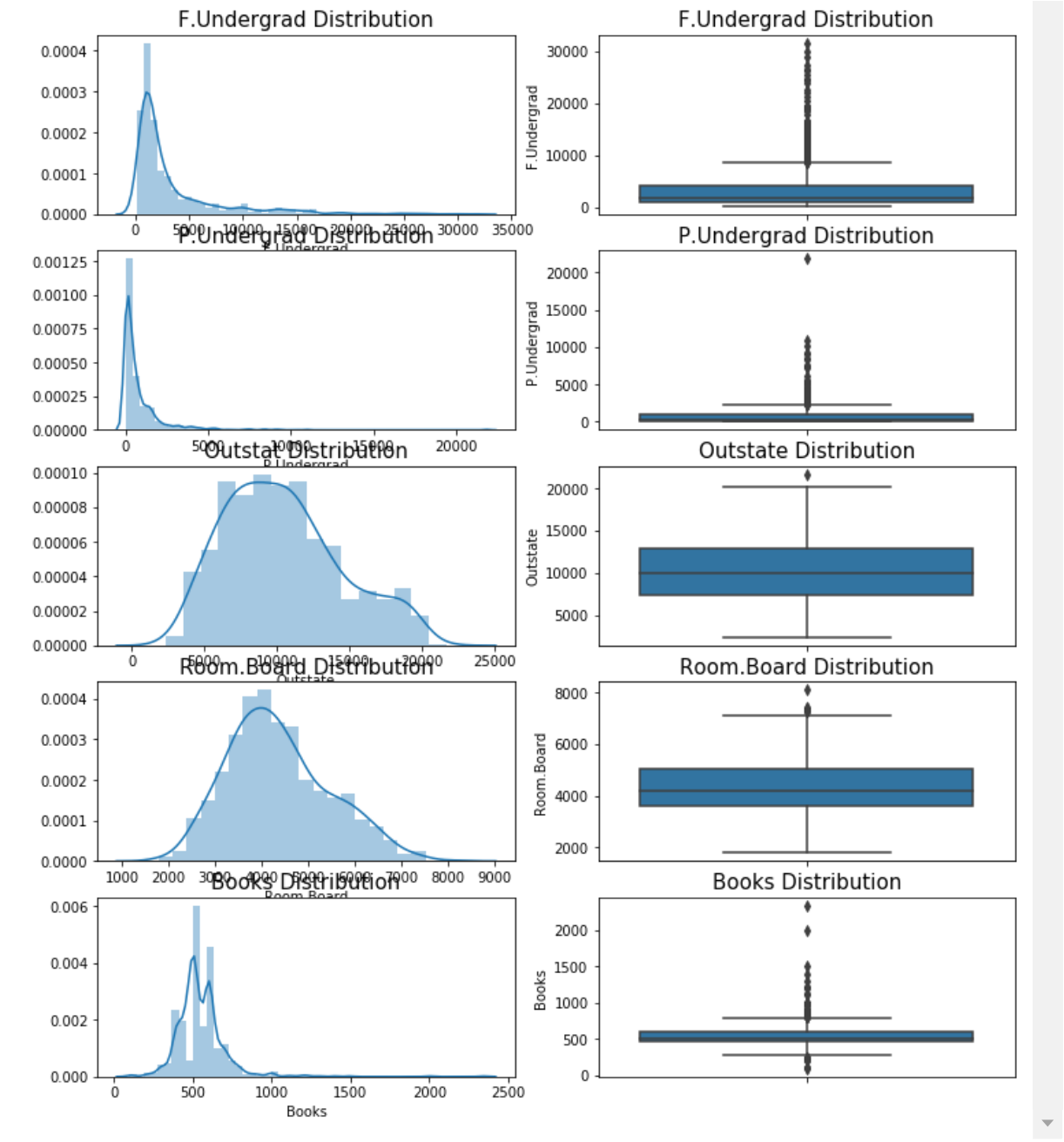
In [56]:

```
fig , axes = plt.subplots (nrows=5 , ncols=2)
fig.set_size_inches (12,14)

r = sns.distplot (edu ['F.Undergrad'] , ax=axes [0][0]);
r.set_title ('F.Undergrad Distribution' ,fontsize=15)
r = sns.boxplot (edu ['F.Undergrad'] , orient='v' , ax=axes [0][1]);
r.set_title ('F.Undergrad Distribution' , fontsize= 15)

r = sns.distplot (edu ['P.Undergrad'] , ax=axes [1][0]);
r.set_title ('P.Undergrad Distribution' , fontsize=15)
r = sns.boxplot (edu ['P.Undergrad'] , orient='v', ax=axes [1][1]);
r.set_title ('P.Undergrad Distribution' , fontsize=15)

r = sns.distplot (edu ['Outstate'] , ax=axes [2][0]);
r.set_title ('Outstat Distribution' , fontsize=15)
r = sns.boxplot (edu ['Outstate'], orient='v' ,ax=axes [2][1]);
r.set_title ('Outstate Distribution' , fontsize=15)

r=sns.distplot (edu ['Room.Board'] , ax=axes[3][0]);
r.set_title ('Room.Board Distribution' , fontsize=15)
r = sns.boxplot (edu ['Room.Board'], orient='v',ax=axes[3][1]);
r.set_title ('Room.Board Distribution', fontsize=15)

r = sns.distplot (edu ['Books'] , ax=axes [4][0]);
r.set_title ('Books Distribution' , fontsize=15)
r = sns.boxplot (edu ['Books'] , orient='v' , ax=axes[4][1]);
r.set_title ('Books Distribution' , fontsize=15)
```

executed in 3.09s, finished 16:48:28 2021-01-23

Out[56]:

```
Text(0.5, 1.0, 'Books Distribution')
```



As we can say from above that Room.Board and Outstate are Slightly Normally Distributed with minimum Outliers but P.Undergrad and F.Undergrad are Right Skewed and also Books have the Outliers.

Lets check for next 5 variables. Personal

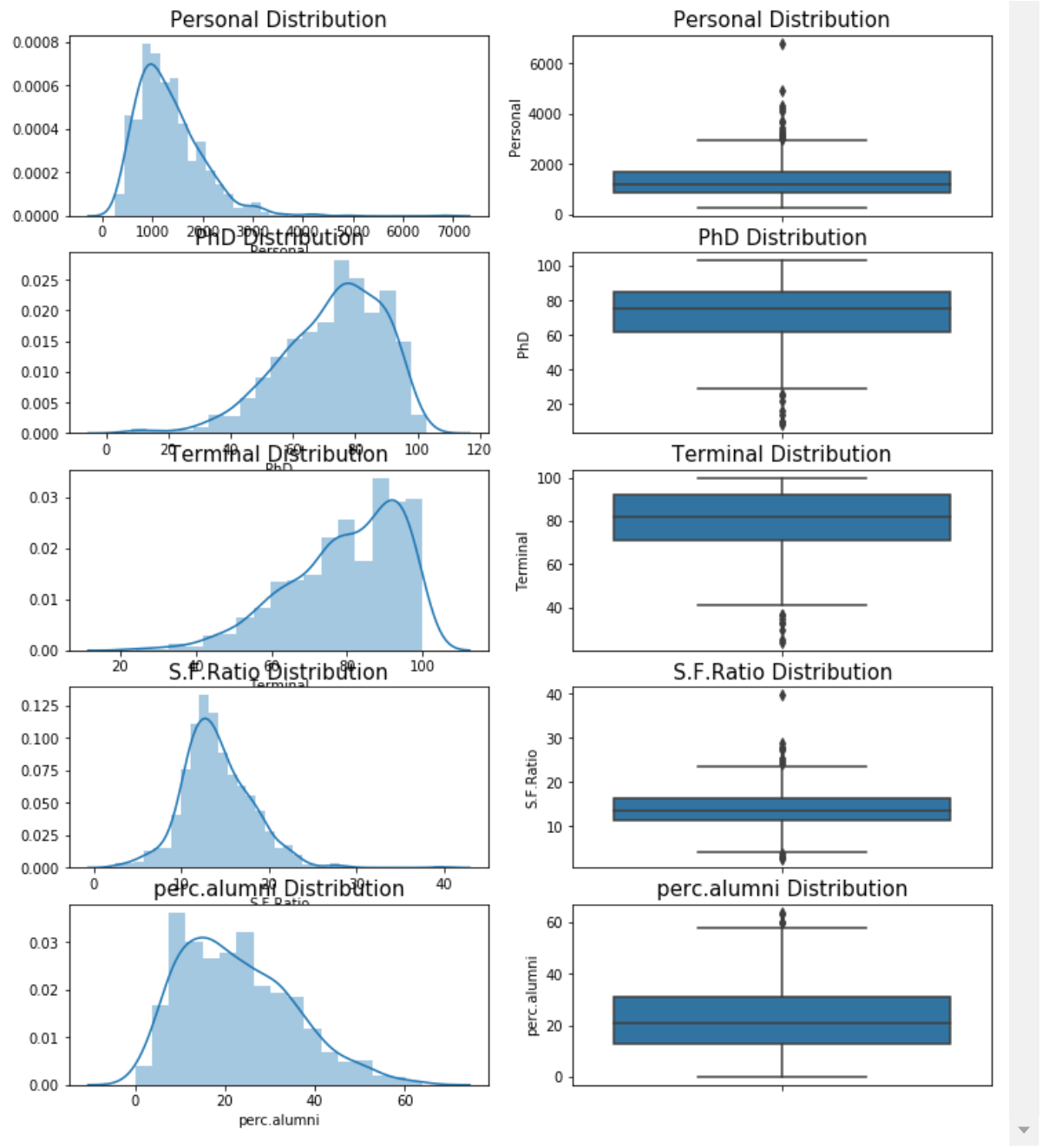
In [64]:

```
fig , axes = plt.subplots (nrows=5 , ncols=2)
fig.set_size_inches (12,14)

r = sns.distplot (edu ['Personal'] , ax=axes [0][0]);
r.set_title ('Personal Distribution' , fontsize=15)
r = sns.boxplot (edu ['Personal'] , orient='v' , ax=axes [0][1]);
r.set_title ('Personal Distribution' , fontsize=15)

r = sns.distplot (edu ['PhD'] , ax=axes[1][0]);
r.set_title ('PhD Distribution' , fontsize=15)
r = sns.boxplot (edu ['PhD'] , orient='v', ax=axes [1][1]);
r.set_title ('PhD Distribution', fontsize=15)

r = sns.distplot (edu ['Terminal'] , ax=axes [2][0]);
r.set_title ('Terminal Distribution' , fontsize=15)
r = sns.boxplot (edu ['Terminal'] , orient='v', ax=axes[2][1]);
r.set_title ('Terminal Distribution' , fontsize=15)

r = sns.distplot (edu ['S.F.Ratio'] , ax=axes [3][0]);
r.set_title ('S.F.Ratio Distribution', fontsize=15)
r = sns.boxplot (edu ['S.F.Ratio'], orient='v',ax=axes[3][1]);
r.set_title ('S.F.Ratio Distribution',fontsize=15)

r = sns.distplot (edu ['perc.alumni'] , ax=axes [4][0]);
r.set_title ('perc.alumni Distribution' , fontsize=15)
r = sns.boxplot (edu ['perc.alumni'] , orient='v' , ax=axes [4][1]);
r.set_title ('perc.alumni Distribution',fontsize=15)
```

executed in 1.99s, finished 17:09:35 2021-01-23

Out[64]:

```
Text(0.5, 1.0, 'perc.alumni Distribution')
```



As we can see from the above figure that Personal,S.F.Ratio and perc.alumni are Right skewed with Outliers and PhD and Terminal are the Left skewed with Outliers.

In [ ]:

In [ ]:

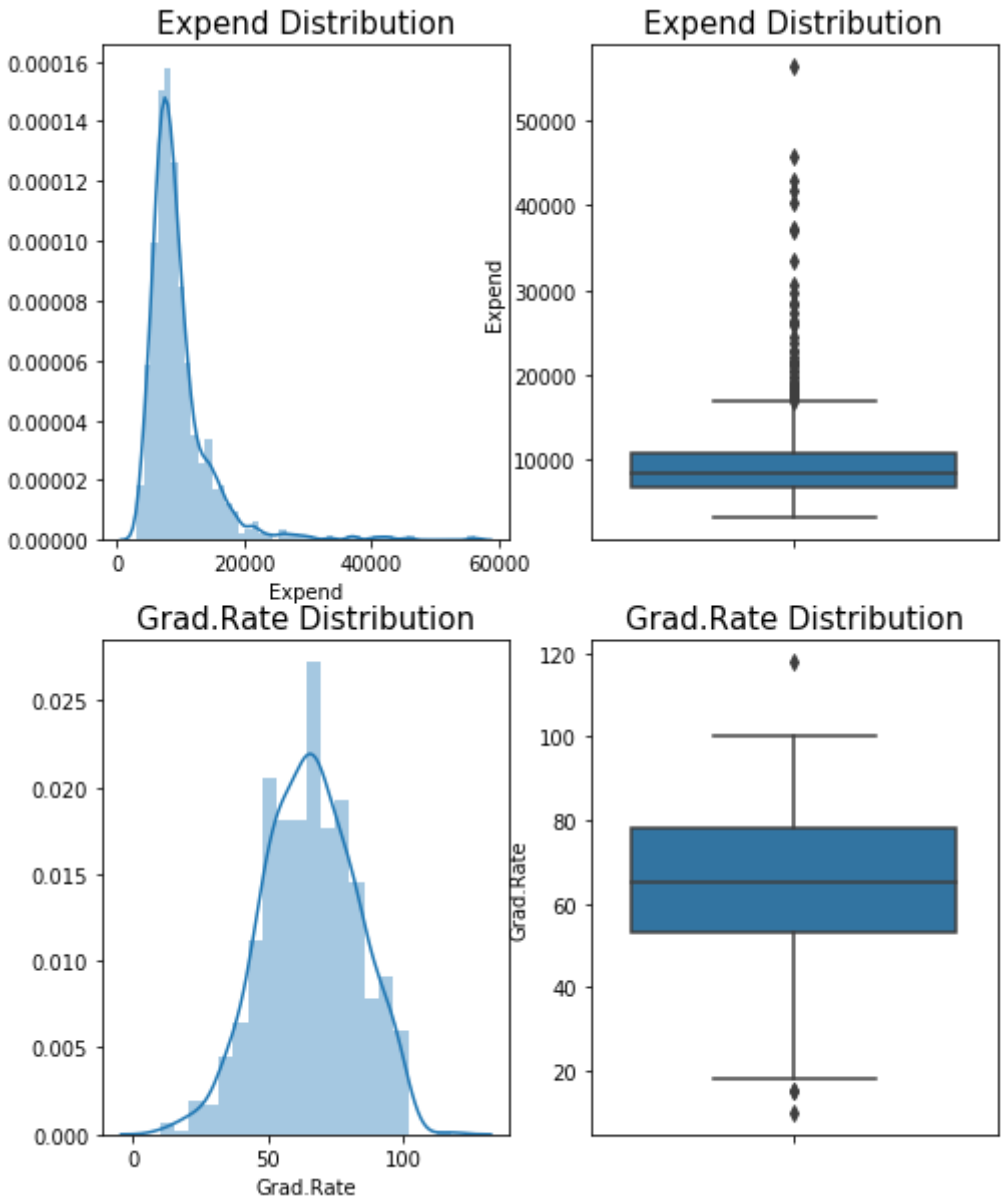
In [70]:

```
fig , axes = plt.subplots (nrows=2, ncols=2)
fig.set_size_inches (8,10)

r = sns.distplot (edu ['Expend'] , ax=axes [0][0]);
r.set_title ('Expend Distribution' , fontsize=15)
r = sns.boxplot (edu ['Expend'] , orient='v',ax=axes[0][1]);
r.set_title ('Expend Distribution' , fontsize=15)

r = sns.distplot (edu ['Grad.Rate'] , ax=axes[1][0]);
r.set_title ('Grad.Rate Distribution', fontsize=15)
r = sns.boxplot (edu ['Grad.Rate'], orient='v',ax=axes [1][1]);
r.set_title ('Grad.Rate Distribution' , fontsize=15)
```

executed in 832ms, finished 17:11:27 2021-01-23

Out[70]:

```
Text(0.5, 1.0, 'Grad.Rate Distribution')
```



Expend is Right Skewed and Grad.Rate is Slightly Left skewed with both having Outliers.

In [ ]:

In [ ]:

Inferences from Univariate Analysis :

All the variables (int,float / continous) in the Given DataSet have Outliers which means some of them are Rightly Skewed and few of them are Left Skewed.

Hence we need to treat the Outliers means cleaning the Data before going further for Model Building.

Top25perc (variable) is slightly Normally Distributed with No Outliers.

In [ ]:

In [ ]:

Multi-Variate Analysis:

It is used to find out the Interactions between different variables more than 2 in the data set. We can use :

corr method

pairplot

heatmap

Corr gives us the comparison / relationship between 2 variables in the data set.

Pair-plot is the graphical representation between 2 variables.

Heatmap is the graphical representation of correlation.

In [72]:

```
# Correlation
edu.corr (method='pearson')
```

executed in 58ms, finished 17:51:44 2021-01-23

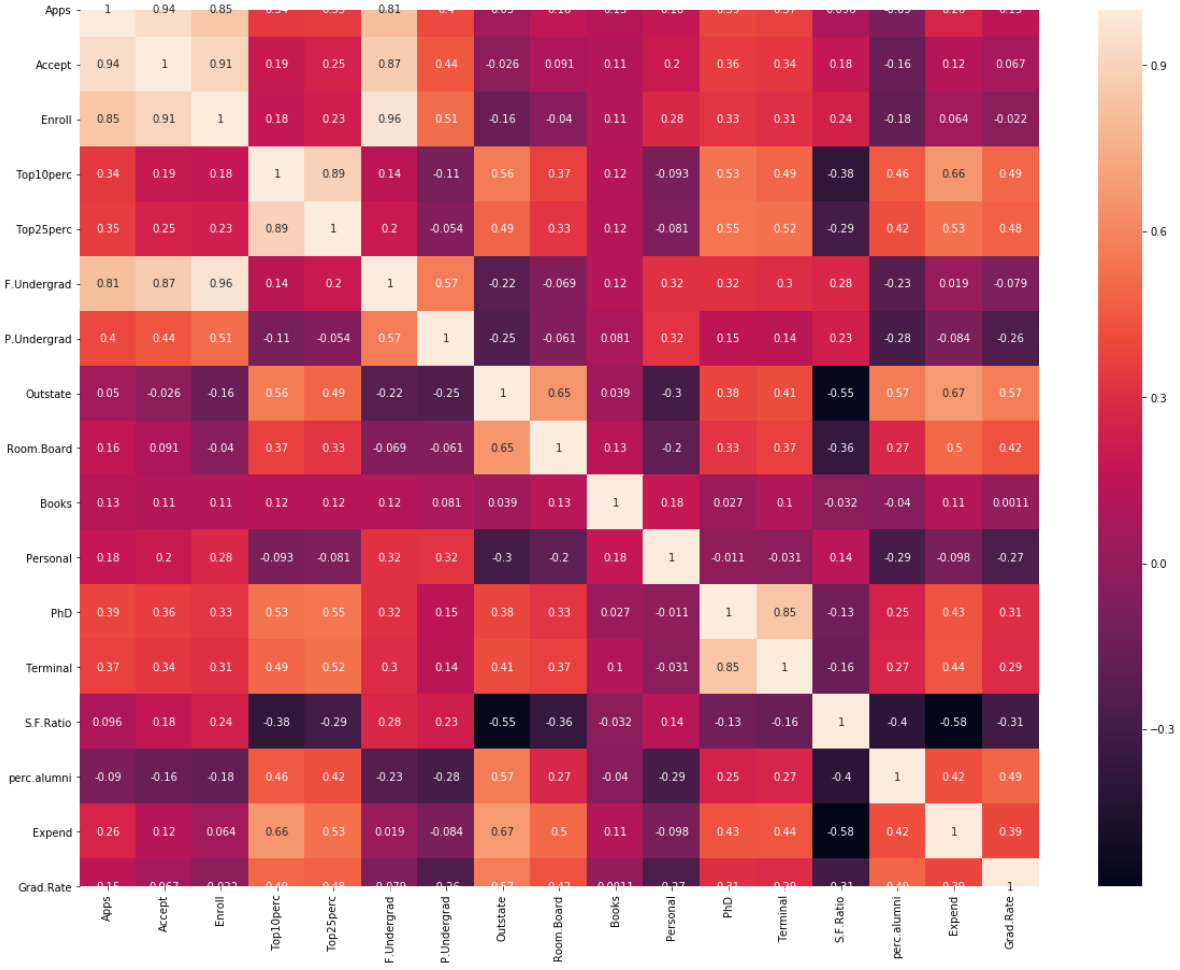
Out[72]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad
Apps	1.000000	0.943451	0.846822	0.338834	0.351640	0.814491	0.398264
Accept	0.943451	1.000000	0.911637	0.192447	0.247476	0.874223	0.441271
Enroll	0.846822	0.911637	1.000000	0.181294	0.226745	0.964640	0.513069
Top10perc	0.338834	0.192447	0.181294	1.000000	0.891995	0.141289	-0.105356
Top25perc	0.351640	0.247476	0.226745	0.891995	1.000000	0.199445	-0.053577
F.Undergrad	0.814491	0.874223	0.964640	0.141289	0.199445	1.000000	0.570512
P.Undergrad	0.398264	0.441271	0.513069	-0.105356	-0.053577	0.570512	1.000000
Outstate	0.050159	-0.025755	-0.155477	0.562331	0.489394	-0.215742	-0.253512
Room.Board	0.164939	0.090899	-0.040232	0.371480	0.331490	-0.068890	-0.061326
Books	0.132559	0.113525	0.112711	0.118858	0.115527	0.115550	0.081200
Personal	0.178731	0.200989	0.280929	-0.093316	-0.080810	0.317200	0.319882
PhD	0.390697	0.355758	0.331469	0.531828	0.545862	0.318337	0.149114
Terminal	0.369491	0.337583	0.308274	0.491135	0.524749	0.300019	0.141904
S.F.Ratio	0.095633	0.176229	0.237271	-0.384875	-0.294629	0.279703	0.232531
perc.alumni	-0.090226	-0.159990	-0.180794	0.455485	0.417864	-0.229462	-0.280792
Expend	0.259592	0.124717	0.064169	0.660913	0.527447	0.018652	-0.083568
Grad.Rate	0.146755	0.067313	-0.022341	0.494989	0.477281	-0.078773	-0.257001

In [76]:

```
# Heat-Map

plt.subplots (figsize=(20,15))
sns.heatmap (edu.corr() , annot=True);
```

executed in 2.44s, finished 17:53:24 2021-01-23



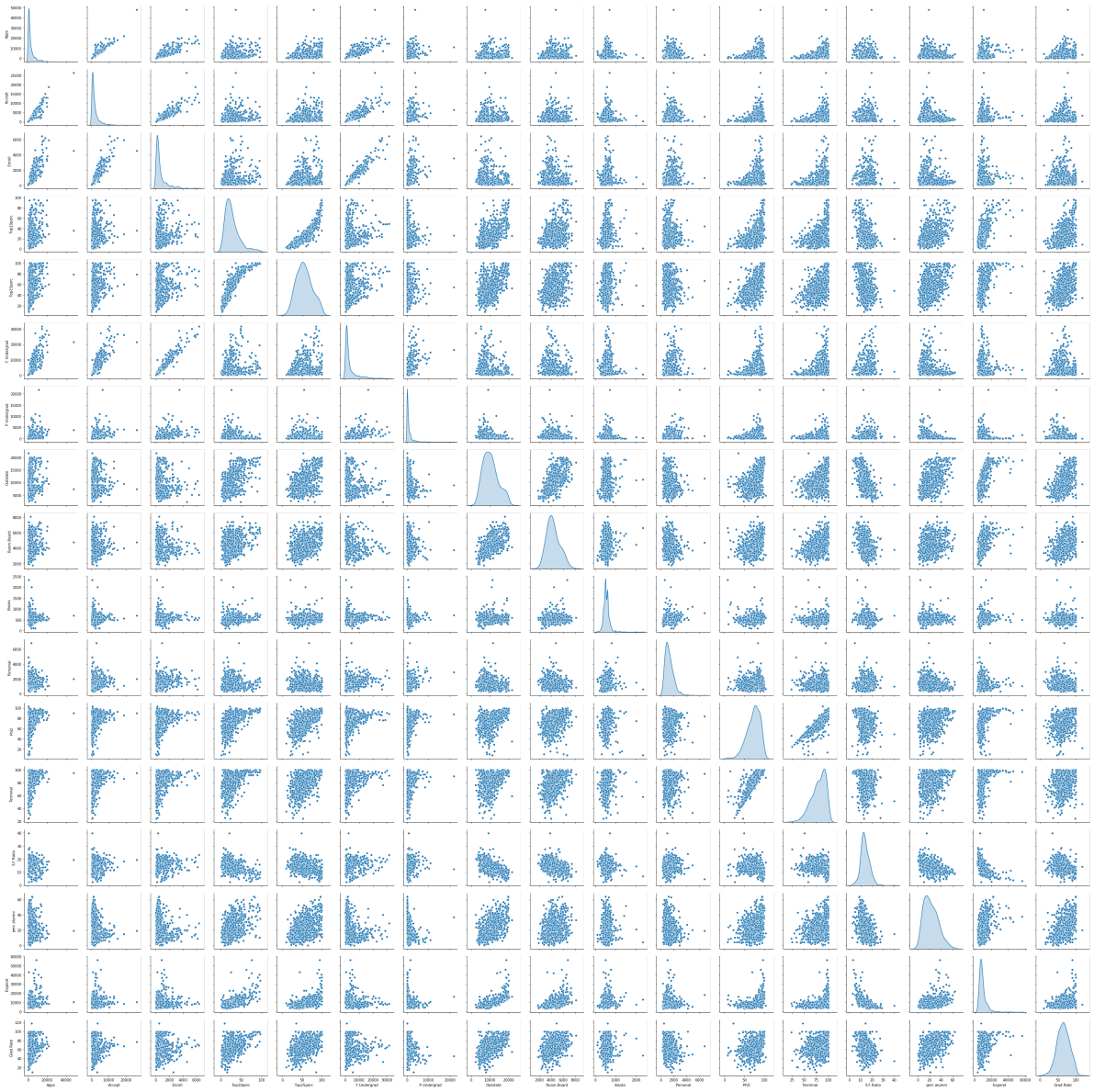
Light colour indicates the Highest Relationship and Dark Colour indicates the Lowest Relationship.

As we can see from the heat-map there are variables which are having the strong relation between the other variables which is the indication of Positive correlation and there are few variable which are Negative correlated.

In [77]:

```
sns.pairplot(edu , diag_kind='kde');
```

executed in 1m 41.1s, finished 18:00:02 2021-01-23



It gives us a clear idea which variable have the strong relationship or we can say Positive relationship with other variables and a Negative relationship with other variables.

In [ ]:

```

```

2.2) Scale the variables and write the inference for using the type of scaling function for this case study.

In [86]:

```
# Lets drop the Categorical Column which is Names and will treate the Outliers before Scali
```

executed in 18ms, finished 19:20:22 2021-01-23

Treating Outliers : Q3 + 1.5 IQR (upper limit) and Q1 - 1.5 IQR (lower limit) anything above the upper limit it will treated as Outliers and it will bring all the values in IQR.

In [87]:

```
# We will define one Function

def remove_outlier (col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1 - (1.5 * IQR)
    upper_range= Q3 + (1.5 * IQR)
    return lower_range,upper_range
```

executed in 23ms, finished 19:29:16 2021-01-23

In [111]:

```
lap,uap = remove_outlier(edu['Apps'])
edu['Apps'] = np.where(edu ['Apps']>uap,uap,edu['Apps'])
edu['Apps'] = np.where(edu ['Apps']<lap,lap,edu['Apps'])

lac,uac = remove_outlier(edu['Accept'])
edu['Accept']=np.where(edu['Accept']>uac,uac,edu['Accept'])
edu['Accept']=np.where(edu['Accept']<lac,lac,edu['Accept'])

len,uen = remove_outlier (edu['Enroll'])
edu['Enroll'] = np.where(edu['Enroll']>uen,uen,edu['Enroll'])
edu['Enroll'] = np.where(edu['Enroll']<len,len,edu['Enroll'])

lt1,ut1 = remove_outlier(edu['Top10perc'])
edu['Top10perc'] = np.where(edu['Top10perc']>ut1,ut1,edu['Top10perc'])
edu['Top10perc'] = np.where(edu['Top10perc']<lt1,lt1,edu['Top10perc'])

lt2,ut2 = remove_outlier (edu['Top25perc'])
edu['Top25perc'] = np.where(edu['Top25perc']>ut2,ut2,edu['Top25perc'])
edu['Top25perc'] = np.where(edu['Top25perc']<lt2,lt2,edu['Top25perc'])

lf,uf = remove_outlier(edu['F.Undergrad'])
edu['F.Undergrad'] = np.where(edu['F.Undergrad']>uf,uf,edu['F.Undergrad'])
edu['F.Undergrad'] = np.where(edu['F.Undergrad']<lf,lf,edu['F.Undergrad'])

lp,up = remove_outlier (edu['P.Undergrad'])
edu['P.Undergrad']=np.where(edu['P.Undergrad']>up,up,edu['P.Undergrad'])
edu['P.Undergrad']=np.where(edu['P.Undergrad']<lp,lp,edu['P.Undergrad'])

lo,uo = remove_outlier(edu['Outstate'])
edu['Outstate']=np.where(edu['Outstate']>uo,uo,edu['Outstate'])
edu['Outstate']=np.where(edu['Outstate']<lo,lo,edu['Outstate'])

lrb,urb = remove_outlier(edu['Room.Board'])
edu['Room.Board']=np.where(edu['Room.Board']>urb,urb,edu['Room.Board'])
edu['Room.Board']=np.where(edu['Room.Board']<lrb,lrb,edu['Room.Board'])

lbo,ubo=remove_outlier(edu['Books'])
edu['Books']=np.where(edu['Books']>ubo,ubo,edu['Books'])
edu['Books']=np.where(edu['Books']<lbo,lbo,edu['Books'])

lpe,upe = remove_outlier(edu['Personal'])
edu['Personal']=np.where(edu['Personal']>upe,upe,edu['Personal'])
edu['Personal']=np.where(edu['Personal']<lpe,lpe,edu['Personal'])

ld,ud = remove_outlier(edu['PhD'])
edu['PhD']=np.where(edu['PhD']>ud,ud,edu['PhD'])
edu['PhD']=np.where(edu['PhD']<ld,ld,edu['PhD'])

lter,uter=remove_outlier(edu['Terminal'])
edu['Terminal']=np.where(edu['Terminal']>uter,uter,edu['Terminal'])
edu['Terminal']=np.where(edu['Terminal']<lter,lter,edu['Terminal'])

lsf,usf=remove_outlier(edu['S.F.Ratio'])
edu['S.F.Ratio']=np.where(edu['S.F.Ratio']>usf,usf,edu['S.F.Ratio'])
edu['S.F.Ratio']=np.where(edu['S.F.Ratio']<lsf,lsf,edu['S.F.Ratio'])

lalu,ualu=remove_outlier(edu['perc.alumni'])
edu['perc.alumni']=np.where(edu['perc.alumni']>ualu,ualu,edu['perc.alumni'])
edu['perc.alumni']=np.where(edu['perc.alumni']<lalu,lalu,edu['perc.alumni'])
```

```
lex,uex = remove_outlier(edu['Expend'])
edu['Expend']=np.where(edu['Expend']>uex,uex,edu['Expend'])
edu['Expend']=np.where(edu['Expend']<lex,lex,edu['Expend'])

lgr,ugr = remove_outlier(edu['Grad.Rate'])
edu['Grad.Rate']=np.where(edu['Grad.Rate']>ugr,ugr,edu['Grad.Rate'])
edu['Grad.Rate']=np.where(edu['Grad.Rate']<lgr,lgr,edu['Grad.Rate'])
```

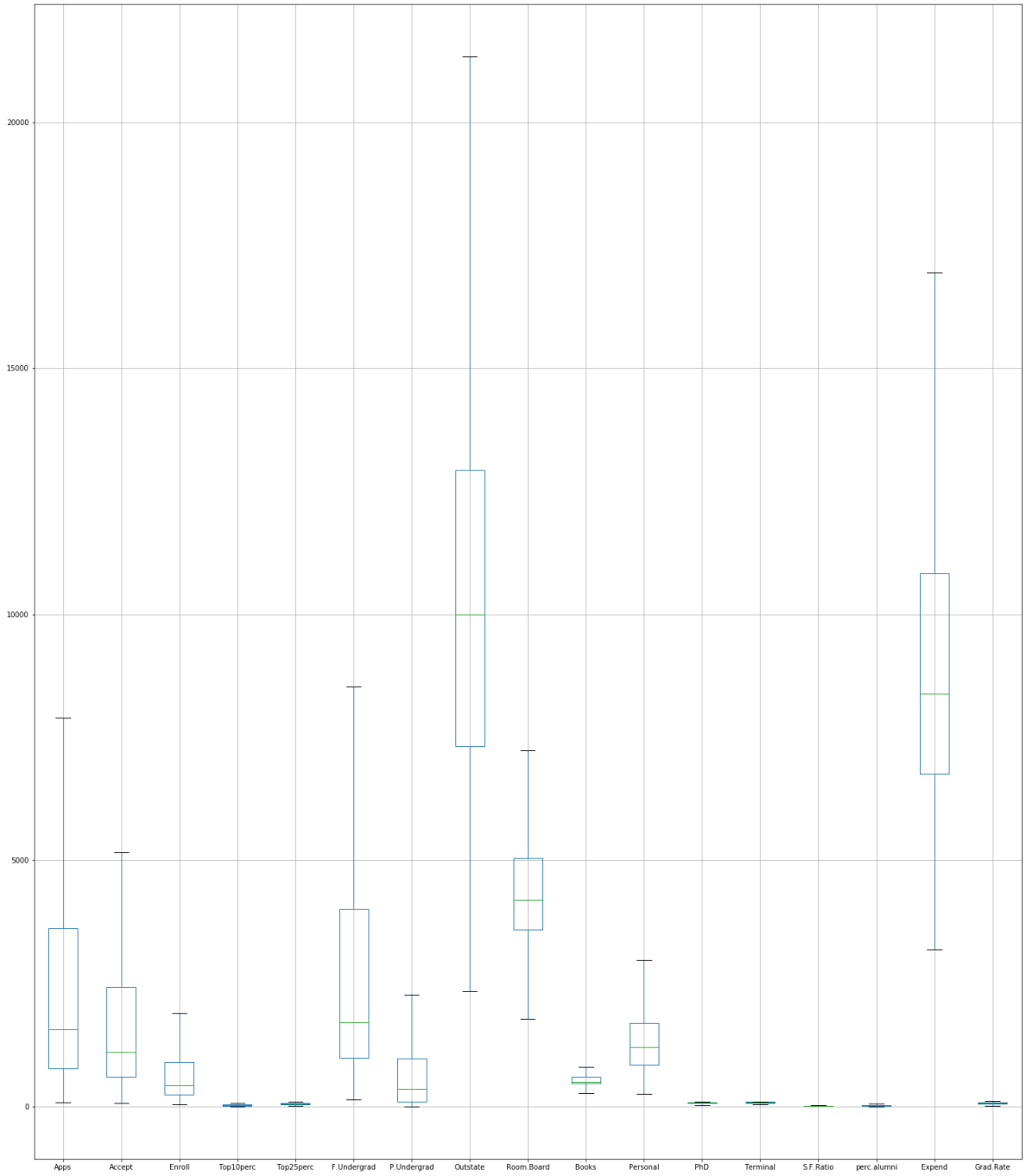
executed in 107ms, finished 20:00:45 2021-01-23

Now we will check the Outliers after handling them.

In [112]:

```
plt.subplots(figsize=(25,30))
edu.boxplot(figsize=(25,20));
```

executed in 1.16s, finished 20:02:57 2021-01-23



So we have handled the Outliers.

In [119]:

```
# We will scaling the variables:

edu.head()
```

executed in 44ms, finished 20:11:08 2021-01-23

Out[119]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Bo
0	1660.0	1232.0	721.0	23.0	52.0	2885.0	537.0	7440.0	330
1	2186.0	1924.0	512.0	16.0	29.0	2683.0	1227.0	12280.0	645
2	1428.0	1097.0	336.0	22.0	50.0	1036.0	99.0	11250.0	375
3	417.0	349.0	137.0	60.0	89.0	510.0	63.0	12960.0	545
4	193.0	146.0	55.0	16.0	44.0	249.0	869.0	7560.0	412

In [130]:

```
from scipy.stats import zscore
edu_scale= edu.apply(zscore) # we have defined one new variable which dont have categorical
```

executed in 25ms, finished 20:17:51 2021-01-23

In [127]:

```
edu_scale.head()
```

executed in 44ms, finished 20:13:45 2021-01-23

Out[127]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
0	-0.376493	-0.337830	0.106380	-0.246780	-0.191827	-0.018769	-0.166083	-0.746480
1	-0.159195	0.116744	-0.260441	-0.696290	-1.353911	-0.093626	0.797856	0.457762
2	-0.472336	-0.426511	-0.569343	-0.310996	-0.292878	-0.703966	-0.777974	0.201488
3	-0.889994	-0.917871	-0.918613	2.129202	1.677612	-0.898889	-0.828267	0.626954
4	-0.982532	-1.051221	-1.062533	-0.696290	-0.596031	-0.995610	0.297726	-0.716623

In [129]:

```
edu_scale.describe()
```

executed in 139ms, finished 20:15:28 2021-01-23

Out[129]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad
count	7.770000e+02	7.770000e+02	7.770000e+02	7.770000e+02	7.770000e+02	7.770000e+02
mean	1.234534e-16	1.340626e-16	1.521645e-16	-2.250452e-18	-1.546739e-16	-1.911679e-16
std	1.000644e+00	1.000644e+00	1.000644e+00	1.000644e+00	1.000644e+00	1.000644e+00
min	-1.028801e+00	-1.099832e+00	-1.097636e+00	-1.659526e+00	-2.364419e+00	-1.036373e+00
25%	-7.416863e-01	-7.503620e-01	-7.343246e-01	-7.605060e-01	-7.476067e-01	-7.202711e-01
50%	-4.186307e-01	-4.179715e-01	-3.973405e-01	-2.467801e-01	-9.077663e-02	-4.553086e-01
75%	4.348639e-01	4.451926e-01	4.240582e-01	5.238086e-01	6.671042e-01	3.962772e-01
max	2.199689e+00	2.238524e+00	2.161632e+00	2.450281e+00	2.233391e+00	2.071100e+00

Inferences:So after scaling the Variables we bring all of them into -3 to +3 Standard Deviation.

In [ ]:

```

```

2.3) Comment on the comparison between covariance and the correlation matrix.

In [131]:

```
# Covariance is the Direction of the Variables and Correlation is how strong is the relatio
```

executed in 16ms, finished 20:20:57 2021-01-23

In [132]:

```
# Lets Create a Covariance matrix:

cov_matrix = np.cov(edu_scale.T)
print('Covariance Matrix %d' , cov_matrix)
```

executed in 493ms, finished 20:26:16 2021-01-23

```
Covariance Matrix %d [[ 1.00128866e+00  9.56537704e-01  8.98039052e-01  3.21
756324e-01
   3.64960691e-01  8.62111140e-01  5.20492952e-01  6.54209711e-02
   1.87717056e-01  2.36441941e-01  2.30243993e-01  4.64521757e-01
   4.35037784e-01  1.26573895e-01 -1.01288006e-01  2.43248206e-01
   1.50997775e-01]
 [ 9.56537704e-01  1.00128866e+00  9.36482483e-01  2.23586208e-01
   2.74033187e-01  8.98189799e-01  5.73428908e-01 -5.00874847e-03
   1.19740419e-01  2.08974091e-01  2.56676290e-01  4.27891234e-01
   4.03929238e-01  1.88748711e-01 -1.65728801e-01  1.62016688e-01
   7.90839722e-02]
 [ 8.98039052e-01  9.36482483e-01  1.00128866e+00  1.71977357e-01
   2.30730728e-01  9.68548601e-01  6.42421828e-01 -1.55856056e-01
  -2.38762560e-02  2.02317274e-01  3.39785395e-01  3.82031198e-01
   3.54835877e-01  2.74622251e-01 -2.23009677e-01  5.42906862e-02
  -2.32810071e-02]
 [ 3.21756324e-01  2.23586208e-01  1.71977357e-01  1.00128866e+00
   9.15052977e-01  1.11358019e-01 -1.80240778e-01  5.62884044e-01
   3.57826139e-01  1.53650150e-01 -1.16880152e-01  5.44748764e-01
   5.07401238e-01 -3.88425719e-01  4.56384036e-01  6.57885921e-01
   4.94306540e-01]
 [ 3.64960691e-01  2.74033187e-01  2.30730728e-01  9.15052977e-01
   1.00128866e+00  1.81429267e-01 -9.94231153e-02  4.90200034e-01
   3.31413314e-01  1.69979808e-01 -8.69219644e-02  5.52172085e-01
   5.28333659e-01 -2.97616423e-01  4.17369123e-01  5.73643193e-01
   4.79601950e-01]
 [ 8.62111140e-01  8.98189799e-01  9.68548601e-01  1.11358019e-01
   1.81429267e-01  1.00128866e+00  6.97027420e-01 -2.26457040e-01
  -5.45459528e-02  2.08147257e-01  3.60246460e-01  3.62030390e-01
   3.35485771e-01  3.24921933e-01 -2.85825062e-01  3.71119607e-04
  -8.23447851e-02]
 [ 5.20492952e-01  5.73428908e-01  6.42421828e-01 -1.80240778e-01
  -9.94231153e-02  6.97027420e-01  1.00128866e+00 -3.54672874e-01
  -6.77252009e-02  1.22686416e-01  3.44495974e-01  1.27827147e-01
   1.22309141e-01  3.71084841e-01 -4.19874031e-01 -2.02189396e-01
  -2.65499420e-01]
 [ 6.54209711e-02 -5.00874847e-03 -1.55856056e-01  5.62884044e-01
   4.90200034e-01 -2.26457040e-01 -3.54672874e-01  1.00128866e+00
   6.56333564e-01  5.11656377e-03 -3.26028927e-01  3.91824814e-01
   4.13110264e-01 -5.74421963e-01  5.66465309e-01  7.76326650e-01
   5.73195743e-01]
 [ 1.87717056e-01  1.19740419e-01 -2.38762560e-02  3.57826139e-01
   3.31413314e-01 -5.45459528e-02 -6.77252009e-02  6.56333564e-01
   1.00128866e+00  1.09064551e-01 -2.19837042e-01  3.41908577e-01
   3.79759015e-01 -3.76915472e-01  2.72743761e-01  5.81370284e-01
   4.26338910e-01]
 [ 2.36441941e-01  2.08974091e-01  2.02317274e-01  1.53650150e-01
   1.69979808e-01  2.08147257e-01  1.22686416e-01  5.11656377e-03
   1.09064551e-01  1.00128866e+00  2.40172145e-01  1.36566243e-01
   1.59523091e-01 -8.54689129e-03 -4.28870629e-02  1.50176551e-01
  -8.06107505e-03]
 [ 2.30243993e-01  2.56676290e-01  3.39785395e-01 -1.16880152e-01
  -8.69219644e-02  3.60246460e-01  3.44495974e-01 -3.26028927e-01
```

```
  -2.19837042e-01  2.40172145e-01  1.00128866e+00 -1.16986124e-02
  -3.20117803e-02  1.74136664e-01 -3.06146886e-01 -1.63481407e-01
  -2.91268705e-01]
 [ 4.64521757e-01  4.27891234e-01  3.82031198e-01  5.44748764e-01
   5.52172085e-01  3.62030390e-01  1.27827147e-01  3.91824814e-01
   3.41908577e-01  1.36566243e-01 -1.16986124e-02  1.00128866e+00
   8.64040263e-01 -1.29556494e-01  2.49197779e-01  5.11186852e-01
   3.10418895e-01]
 [ 4.35037784e-01  4.03929238e-01  3.54835877e-01  5.07401238e-01
   5.28333659e-01  3.35485771e-01  1.22309141e-01  4.13110264e-01
   3.79759015e-01  1.59523091e-01 -3.20117803e-02  8.64040263e-01
   1.00128866e+00 -1.51187934e-01  2.66375402e-01  5.24743500e-01
   2.93180212e-01]
 [ 1.26573895e-01  1.88748711e-01  2.74622251e-01 -3.88425719e-01
  -2.97616423e-01  3.24921933e-01  3.71084841e-01 -5.74421963e-01
  -3.76915472e-01 -8.54689129e-03  1.74136664e-01 -1.29556494e-01
  -1.51187934e-01  1.00128866e+00 -4.12632056e-01 -6.55219504e-01
  -3.08922187e-01]
 [-1.01288006e-01 -1.65728801e-01 -2.23009677e-01  4.56384036e-01
   4.17369123e-01 -2.85825062e-01 -4.19874031e-01  5.66465309e-01
   2.72743761e-01 -4.28870629e-02 -3.06146886e-01  2.49197779e-01
   2.66375402e-01 -4.12632056e-01  1.00128866e+00  4.63518674e-01
   4.92040760e-01]
 [ 2.43248206e-01  1.62016688e-01  5.42906862e-02  6.57885921e-01
   5.73643193e-01  3.71119607e-04 -2.02189396e-01  7.76326650e-01
   5.81370284e-01  1.50176551e-01 -1.63481407e-01  5.11186852e-01
   5.24743500e-01 -6.55219504e-01  4.63518674e-01  1.00128866e+00
   4.15826026e-01]
 [ 1.50997775e-01  7.90839722e-02 -2.32810071e-02  4.94306540e-01
   4.79601950e-01 -8.23447851e-02 -2.65499420e-01  5.73195743e-01
   4.26338910e-01 -8.06107505e-03 -2.91268705e-01  3.10418895e-01
   2.93180212e-01 -3.08922187e-01  4.92040760e-01  4.15826026e-01
   1.00128866e+00]]
```

In [133]:

```
edu_corr = edu.copy()# without scaling
edu_corr.corr()
```

executed in 42ms, finished 20:28:18 2021-01-23

Out[133]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad
Apps	1.000000	0.955307	0.896883	0.321342	0.364491	0.861002	0.519823
Accept	0.955307	1.000000	0.935277	0.223298	0.273681	0.897034	0.572691
Enroll	0.896883	0.935277	1.000000	0.171756	0.230434	0.967302	0.641595
Top10perc	0.321342	0.223298	0.171756	1.000000	0.913875	0.111215	-0.180009
Top25perc	0.364491	0.273681	0.230434	0.913875	1.000000	0.181196	-0.099295
F.Undergrad	0.861002	0.897034	0.967302	0.111215	0.181196	1.000000	0.696130
P.Undergrad	0.519823	0.572691	0.641595	-0.180009	-0.099295	0.696130	1.000000
Outstate	0.065337	-0.005002	-0.155655	0.562160	0.489569	-0.226166	-0.354216
Room.Board	0.187475	0.119586	-0.023846	0.357366	0.330987	-0.054476	-0.067638
Books	0.236138	0.208705	0.202057	0.153452	0.169761	0.207879	0.122529
Personal	0.229948	0.256346	0.339348	-0.116730	-0.086810	0.359783	0.344053
PhD	0.463924	0.427341	0.381540	0.544048	0.551461	0.361564	0.127663
Terminal	0.434478	0.403409	0.354379	0.506748	0.527654	0.335054	0.122152
S.F.Ratio	0.126411	0.188506	0.274269	-0.387926	-0.297233	0.324504	0.370607
perc.alumni	-0.101158	-0.165516	-0.222723	0.455797	0.416832	-0.285457	-0.419334
Expend	0.242935	0.161808	0.054221	0.657039	0.572905	0.000371	-0.201929
Grad.Rate	0.150803	0.078982	-0.023251	0.493670	0.478985	-0.082239	-0.265158

In [134]:

```
edu_scale.corr()# After Scaling.
```

executed in 50ms, finished 20:28:37 2021-01-23

Out[134]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad
Apps	1.000000	0.955307	0.896883	0.321342	0.364491	0.861002	0.519823
Accept	0.955307	1.000000	0.935277	0.223298	0.273681	0.897034	0.572691
Enroll	0.896883	0.935277	1.000000	0.171756	0.230434	0.967302	0.641595
Top10perc	0.321342	0.223298	0.171756	1.000000	0.913875	0.111215	-0.180009
Top25perc	0.364491	0.273681	0.230434	0.913875	1.000000	0.181196	-0.099295
F.Undergrad	0.861002	0.897034	0.967302	0.111215	0.181196	1.000000	0.696130
P.Undergrad	0.519823	0.572691	0.641595	-0.180009	-0.099295	0.696130	1.000000
Outstate	0.065337	-0.005002	-0.155655	0.562160	0.489569	-0.226166	-0.354216
Room.Board	0.187475	0.119586	-0.023846	0.357366	0.330987	-0.054476	-0.067638
Books	0.236138	0.208705	0.202057	0.153452	0.169761	0.207879	0.122529
Personal	0.229948	0.256346	0.339348	-0.116730	-0.086810	0.359783	0.344053
PhD	0.463924	0.427341	0.381540	0.544048	0.551461	0.361564	0.127663
Terminal	0.434478	0.403409	0.354379	0.506748	0.527654	0.335054	0.122152
S.F.Ratio	0.126411	0.188506	0.274269	-0.387926	-0.297233	0.324504	0.370607
perc.alumni	-0.101158	-0.165516	-0.222723	0.455797	0.416832	-0.285457	-0.419334
Expend	0.242935	0.161808	0.054221	0.657039	0.572905	0.000371	-0.201929
Grad.Rate	0.150803	0.078982	-0.023251	0.493670	0.478985	-0.082239	-0.265158

So After Scaling Data Covariance and Correlation has the SAME VALUE.

In [ ]:

```
```

In [ ]:

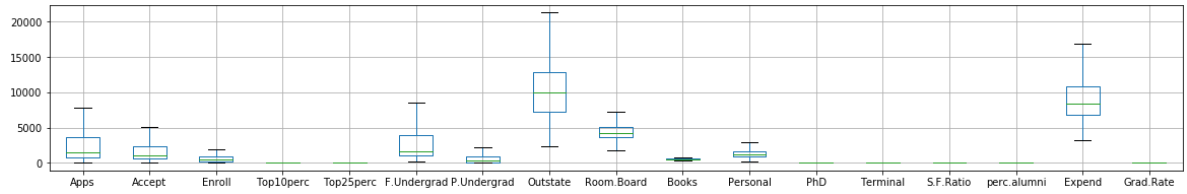
```
```

2.4) Check the dataset for outliers before and after scaling. Draw your inferences from this exercise.

In [135]:

```
# Before scaling :
edu.boxplot (figsize=(20,3));
```

executed in 636ms, finished 20:37:26 2021-01-23



There are Multple Outliers with skewness in the Data.

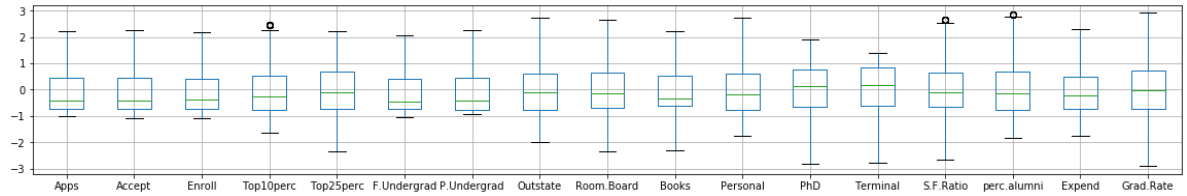
In [ ]:

```
```

In [136]:

```
# After Scaling :
edu_scale.boxplot (figsize=(20,3));
```

executed in 653ms, finished 20:37:54 2021-01-23



Outliers have been removed and all the variables have near by equal median's and all are in the Same Scale.

In [ ]:

```
```

In [ ]:

```
```

2.5) Build the covariance matrix, eigenvalues and eigenvector.

In [139]:

```
#Cavirance Matrix:

cov_matrix = np.cov(edu_scale.T)
print('Covariance Matrix %d' , cov_matrix)
```

executed in 24ms, finished 01:05:34 2021-01-24

```
Covariance Matrix %d [[ 1.00128866e+00  9.56537704e-01  8.98039052e-01  3.21
756324e-01
   3.64960691e-01  8.62111140e-01  5.20492952e-01  6.54209711e-02
   1.87717056e-01  2.36441941e-01  2.30243993e-01  4.64521757e-01
   4.35037784e-01  1.26573895e-01 -1.01288006e-01  2.43248206e-01
   1.50997775e-01]
 [ 9.56537704e-01  1.00128866e+00  9.36482483e-01  2.23586208e-01
   2.74033187e-01  8.98189799e-01  5.73428908e-01 -5.00874847e-03
   1.19740419e-01  2.08974091e-01  2.56676290e-01  4.27891234e-01
   4.03929238e-01  1.88748711e-01 -1.65728801e-01  1.62016688e-01
   7.90839722e-02]
 [ 8.98039052e-01  9.36482483e-01  1.00128866e+00  1.71977357e-01
   2.30730728e-01  9.68548601e-01  6.42421828e-01 -1.55856056e-01
  -2.38762560e-02  2.02317274e-01  3.39785395e-01  3.82031198e-01
   3.54835877e-01  2.74622251e-01 -2.23009677e-01  5.42906862e-02
  -2.32810071e-02]
 [ 3.21756324e-01  2.23586208e-01  1.71977357e-01  1.00128866e+00
   9.15052977e-01  1.11358019e-01 -1.80240778e-01  5.62884044e-01
   3.57826139e-01  1.53650150e-01 -1.16880152e-01  5.44748764e-01
   5.07401238e-01 -3.88425719e-01  4.56384036e-01  6.57885921e-01
   4.94306540e-01]
 [ 3.64960691e-01  2.74033187e-01  2.30730728e-01  9.15052977e-01
   1.00128866e+00  1.81429267e-01 -9.94231153e-02  4.90200034e-01
   3.31413314e-01  1.69979808e-01 -8.69219644e-02  5.52172085e-01
   5.28333659e-01 -2.97616423e-01  4.17369123e-01  5.73643193e-01
   4.79601950e-01]
 [ 8.62111140e-01  8.98189799e-01  9.68548601e-01  1.11358019e-01
   1.81429267e-01  1.00128866e+00  6.97027420e-01 -2.26457040e-01
  -5.45459528e-02  2.08147257e-01  3.60246460e-01  3.62030390e-01
   3.35485771e-01  3.24921933e-01 -2.85825062e-01  3.71119607e-04
  -8.23447851e-02]
 [ 5.20492952e-01  5.73428908e-01  6.42421828e-01 -1.80240778e-01
  -9.94231153e-02  6.97027420e-01  1.00128866e+00 -3.54672874e-01
  -6.77252009e-02  1.22686416e-01  3.44495974e-01  1.27827147e-01
   1.22309141e-01  3.71084841e-01 -4.19874031e-01 -2.02189396e-01
  -2.65499420e-01]
 [ 6.54209711e-02 -5.00874847e-03 -1.55856056e-01  5.62884044e-01
   4.90200034e-01 -2.26457040e-01 -3.54672874e-01  1.00128866e+00
   6.56333564e-01  5.11656377e-03 -3.26028927e-01  3.91824814e-01
   4.13110264e-01 -5.74421963e-01  5.66465309e-01  7.76326650e-01
   5.73195743e-01]
 [ 1.87717056e-01  1.19740419e-01 -2.38762560e-02  3.57826139e-01
   3.31413314e-01 -5.45459528e-02 -6.77252009e-02  6.56333564e-01
   1.00128866e+00  1.09064551e-01 -2.19837042e-01  3.41908577e-01
   3.79759015e-01 -3.76915472e-01  2.72743761e-01  5.81370284e-01
   4.26338910e-01]
 [ 2.36441941e-01  2.08974091e-01  2.02317274e-01  1.53650150e-01
   1.69979808e-01  2.08147257e-01  1.22686416e-01  5.11656377e-03
   1.09064551e-01  1.00128866e+00  2.40172145e-01  1.36566243e-01
   1.59523091e-01 -8.54689129e-03 -4.28870629e-02  1.50176551e-01
  -8.06107505e-03]
 [ 2.30243993e-01  2.56676290e-01  3.39785395e-01 -1.16880152e-01
  -8.69219644e-02  3.60246460e-01  3.44495974e-01 -3.26028927e-01
```

```
  -2.19837042e-01  2.40172145e-01  1.00128866e+00 -1.16986124e-02
  -3.20117803e-02  1.74136664e-01 -3.06146886e-01 -1.63481407e-01
  -2.91268705e-01]
 [ 4.64521757e-01  4.27891234e-01  3.82031198e-01  5.44748764e-01
   5.52172085e-01  3.62030390e-01  1.27827147e-01  3.91824814e-01
   3.41908577e-01  1.36566243e-01 -1.16986124e-02  1.00128866e+00
   8.64040263e-01 -1.29556494e-01  2.49197779e-01  5.11186852e-01
   3.10418895e-01]
 [ 4.35037784e-01  4.03929238e-01  3.54835877e-01  5.07401238e-01
   5.28333659e-01  3.35485771e-01  1.22309141e-01  4.13110264e-01
   3.79759015e-01  1.59523091e-01 -3.20117803e-02  8.64040263e-01
   1.00128866e+00 -1.51187934e-01  2.66375402e-01  5.24743500e-01
   2.93180212e-01]
 [ 1.26573895e-01  1.88748711e-01  2.74622251e-01 -3.88425719e-01
  -2.97616423e-01  3.24921933e-01  3.71084841e-01 -5.74421963e-01
  -3.76915472e-01 -8.54689129e-03  1.74136664e-01 -1.29556494e-01
  -1.51187934e-01  1.00128866e+00 -4.12632056e-01 -6.55219504e-01
  -3.08922187e-01]
 [-1.01288006e-01 -1.65728801e-01 -2.23009677e-01  4.56384036e-01
   4.17369123e-01 -2.85825062e-01 -4.19874031e-01  5.66465309e-01
   2.72743761e-01 -4.28870629e-02 -3.06146886e-01  2.49197779e-01
   2.66375402e-01 -4.12632056e-01  1.00128866e+00  4.63518674e-01
   4.92040760e-01]
 [ 2.43248206e-01  1.62016688e-01  5.42906862e-02  6.57885921e-01
   5.73643193e-01  3.71119607e-04 -2.02189396e-01  7.76326650e-01
   5.81370284e-01  1.50176551e-01 -1.63481407e-01  5.11186852e-01
   5.24743500e-01 -6.55219504e-01  4.63518674e-01  1.00128866e+00
   4.15826026e-01]
 [ 1.50997775e-01  7.90839722e-02 -2.32810071e-02  4.94306540e-01
   4.79601950e-01 -8.23447851e-02 -2.65499420e-01  5.73195743e-01
   4.26338910e-01 -8.06107505e-03 -2.91268705e-01  3.10418895e-01
   2.93180212e-01 -3.08922187e-01  4.92040760e-01  4.15826026e-01
   1.00128866e+00]]
```

In [140]:

```
# Eigen Values and Eigen vectors.

eig_vals , eig_vecs = np.linalg.eig(cov_matrix)
print('\n Eigen Values for the Data set are \n %' , eig_vals)
print('\n')
print('\n Eigen Vectors for the Data set are \n % ', eig_vecs)
```

executed in 960ms, finished 01:11:36 2021-01-24

```
 Eigen Values for the Data set are
 % [5.6625219  4.89470815 1.12636744 1.00397659 0.87218426 0.7657541
 0.58491404 0.5445048  0.42352336 0.38101777 0.24701456 0.02239369
 0.03789395 0.14726392 0.13434483 0.09883384 0.07469003]



 Eigen Vectors for the Data set are
 %  [[-2.62171542e-01  3.14136258e-01  8.10177245e-02 -9.87761685e-02
  -2.19898081e-01  2.18800617e-03 -2.83715076e-02 -8.99498102e-02
   1.30566998e-01 -1.56464458e-01 -8.62132843e-02  1.82169814e-01
  -5.99137640e-01  8.99775288e-02  8.88697944e-02  5.49428396e-01
   5.41453698e-03]
 [-2.30562461e-01  3.44623583e-01  1.07658626e-01 -1.18140437e-01
  -1.89634940e-01 -1.65212882e-02 -1.29584896e-02 -1.37606312e-01
   1.42275847e-01 -1.49209799e-01 -4.25899061e-02 -3.91041719e-01
   6.61496927e-01  1.58861886e-01  4.37945938e-02  2.91572312e-01
   1.44582845e-02]
 [-1.89276397e-01  3.82813322e-01  8.55296892e-02 -9.30717094e-03
  -1.62314818e-01 -6.80794143e-02 -1.52403625e-02 -1.44216938e-01
   5.08712481e-02 -6.48997860e-02 -4.38408622e-02  7.16684935e-01
   2.33235272e-01 -3.53988202e-02 -6.19241658e-02 -4.17001280e-01
  -4.97908902e-02]
 [-3.38874521e-01 -9.93191661e-02 -7.88293849e-02  3.69115031e-01
  -1.57211016e-01 -8.88656824e-02 -2.57455284e-01  2.89538833e-01
  -1.22467790e-01 -3.58776186e-02  1.77837341e-03 -5.62053913e-02
   2.21448729e-02 -3.92277722e-02  6.99599977e-02  8.79767299e-03
  -7.23645373e-01]
 [-3.34690532e-01 -5.95055011e-02 -5.07938247e-02  4.16824361e-01
  -1.44449474e-01 -2.76268979e-02 -2.39038849e-01  3.45643551e-01
  -1.93936316e-01  6.41786425e-03 -1.02127328e-01  1.96735274e-02
   3.22646978e-02  1.45621999e-01 -9.70282598e-02 -1.07779150e-02
   6.55464648e-01]
 [-1.63293010e-01  3.98636372e-01  7.37077827e-02 -1.39504424e-02
  -1.02728468e-01 -5.16468727e-02 -3.11751439e-02 -1.08748900e-01
   1.45452749e-03 -1.63981359e-04 -3.49993487e-02 -5.42774834e-01
  -3.67681187e-01 -1.33555923e-01 -8.71753137e-02 -5.70683843e-01
   2.53059904e-02]
 [-2.24797091e-02  3.57550046e-01  4.03568700e-02 -2.25351078e-01
   9.56790178e-02 -2.45375721e-02 -1.00138971e-02  1.23841696e-01
  -6.34774326e-01  5.46346279e-01  2.52107094e-01  2.95029745e-02
   2.62494456e-02  5.02487566e-02  4.45537493e-02  1.46321060e-01
  -3.97146972e-02]
 [-2.83547285e-01 -2.51863617e-01  1.49394795e-02 -2.62975384e-01
  -3.72750885e-02 -2.03860462e-02  9.45370782e-02  1.12721477e-02
  -8.36648339e-03 -2.31799759e-01  5.93433149e-01  1.03393587e-03
  -8.14247697e-02  5.60392799e-01  6.72405494e-02 -2.11561014e-01
  -1.59275617e-03]
 [-2.44186588e-01 -1.31909124e-01 -2.11379165e-02 -5.80894132e-01
   6.91080879e-02  2.37267409e-01  9.45210745e-02  3.89639465e-01
```

```
  -2.20526518e-01 -2.55107620e-01 -4.75297296e-01  9.85725168e-03
   2.67779296e-02 -1.07365653e-01  1.77715010e-02 -1.00935084e-01
  -2.82578388e-02]
 [-9.67082754e-02  9.39739472e-02 -6.97121128e-01  3.61562884e-02
  -3.54056654e-02  6.38604997e-01 -1.11193334e-01 -2.39817267e-01
   2.10246624e-02  9.11624912e-02  4.35697999e-02  4.36086500e-03
   1.04624246e-02  5.16224550e-02  3.54343707e-02 -2.86384228e-02
  -8.06259380e-03]
 [ 3.52299594e-02  2.32439594e-01 -5.30972806e-01  1.14982973e-01
   4.75358244e-04 -3.81495854e-01  6.39418106e-01  2.77206569e-01
   1.73715184e-02 -1.27647512e-01  1.51627393e-02 -1.08725257e-02
   4.54572099e-03  9.39409228e-03 -1.18604404e-02  3.38197909e-02
   1.42590097e-03]
 [-3.26410696e-01  5.51390195e-02  8.11134044e-02  1.47260891e-01
   5.50786546e-01  3.34444832e-03  8.92320786e-02 -3.42628480e-02
   1.66510079e-01  1.00975002e-01 -3.91865961e-02  1.33146759e-02
   1.25137966e-02 -7.16590441e-02  7.02656469e-01 -6.38096394e-02
   8.31471932e-02]
 [-3.23115980e-01  4.30332048e-02  5.89785929e-02  8.90079921e-02
   5.90407136e-01  3.54121294e-02  9.16985445e-02 -9.03076644e-02
   1.12609034e-01  8.60363025e-02 -8.48575651e-02  7.38135022e-03
  -1.79275275e-02  1.63820871e-01 -6.62488717e-01  9.85019644e-02
  -1.13374007e-01]
 [ 1.63151642e-01  2.59804556e-01  2.74150657e-01  2.59486122e-01
   1.42842546e-01  4.68752604e-01  1.52864837e-01  2.42807562e-01
  -1.53685343e-01 -4.70527925e-01  3.63042716e-01  8.85797314e-03
   1.83059753e-02 -2.39902591e-01 -4.79006197e-02  6.19970446e-02
   3.83160891e-03]
 [-1.86610828e-01 -2.57092552e-01  1.03715887e-01  2.23982467e-01
  -1.28215768e-01  1.25669415e-02  3.91400512e-01 -5.66073056e-01
  -5.39235753e-01 -1.47628917e-01 -1.73918533e-01 -2.40534190e-02
  -8.03169296e-05 -4.89753356e-02  3.58875507e-02  2.80805469e-02
  -7.32598621e-03]
 [-3.28955847e-01 -1.60008951e-01 -1.84205687e-01 -2.13756140e-01
   2.24240837e-02 -2.31562325e-01 -1.50501305e-01 -1.18823549e-01
   2.42371616e-02 -8.04154875e-02  3.93722676e-01  1.05658769e-02
   5.60069250e-02 -6.90417042e-01 -1.26667522e-01  1.28739213e-01
   1.45099786e-01]
 [-2.38822447e-01 -1.67523664e-01  2.45335837e-01  3.61915064e-02
  -3.56843227e-01  3.13556243e-01  4.68641965e-01  1.80458508e-01
   3.15812873e-01  4.88415259e-01  8.72638706e-02 -2.51028410e-03
   1.48410810e-02 -1.59332164e-01 -6.30737002e-02 -7.09643331e-03
  -3.29024228e-03]]
```

In [ ]:

```
```

In [ ]:

```
```

2.6) Write the explicit form of the first PC (in terms of Eigen Vectors).

In [148]:

```
eig_vecs[0]
```

executed in 14ms, finished 01:27:43 2021-01-24

Out[148]:

```
array([-0.26217154,  0.31413626,  0.08101772, -0.09877617, -0.21989808,
        0.00218801, -0.02837151, -0.08994981,  0.130567  , -0.15646446,
       -0.08621328,  0.18216981, -0.59913764,  0.08997753,  0.08886979,
        0.5494284 ,  0.00541454])
```

In [ ]:

```

```

In [ ]:

```

```

2.7) Discuss the cumulative values of the eigenvalues. How does it help you to decide on the optimum number of principal components? What do the eigenvectors indicate?

Perform PCA and export the data of the Principal Component scores into a data frame.

In [149]:

```
# Cumulative Values : This are the sum of all the values in one array which decides the per
# We will be deciding the number of PC based on the Cumulative values which will give above
```

executed in 6ms, finished 01:30:55 2021-01-24

Cumulative Distribution for Eigen Values

Lets check the cumulative values for the Given Data Set

In [151]:

```
total = sum (eig_vals)
var_exp = [(i / total) * 100 for i in sorted(eig_vals,reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print('Cumulative Distribution for Eigen Values', cum_var_exp)
```

executed in 14ms, finished 01:35:47 2021-01-24

```
Cumulative Distribution for Eigen Values [ 33.26608367  62.02142867  68.6385
9223  74.53673619  79.66062886
  84.15926753  87.59551019  90.79435736  93.28246491  95.52086136
  96.97201814  97.83716159  98.62640821  99.20703552  99.64582321
  99.86844192 100.        ]
```

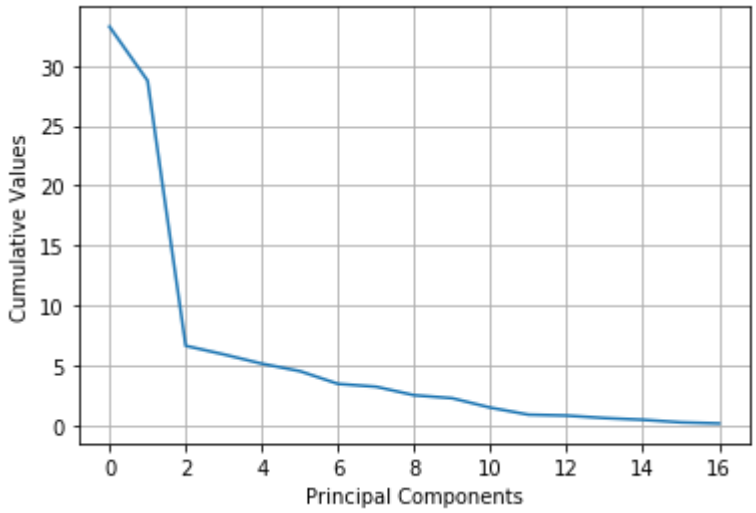
So here we can are getting 80 % in the PC5 but 84 % in PC6 so we will be selecting the 6 principal components.

Lets check visually.

In [156]:

```
plt.plot (var_exp)
plt.xlabel('Principal Components')
plt.ylabel('Cumulative Values')
plt.grid()
plt.show()
```

executed in 323ms, finished 01:48:43 2021-01-24



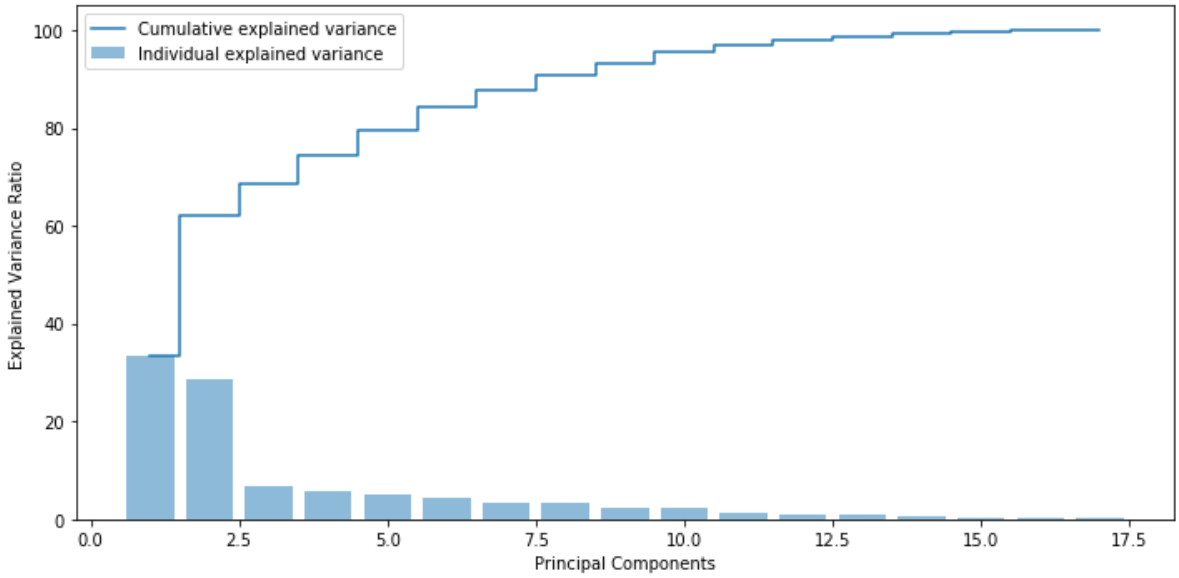
As we can see that Variance is dropping as the PC increasing.

We will be selecting the 6 Principal Components which gives us the 84 % Data which is above the benchmark of 80 %.

In [162]:

```
plt.figure(figsize=(10 , 5))
plt.bar(range(1, eig_vals.size + 1), var_exp, alpha = 0.5, align = 'center', label = 'Indiv
plt.step(range(1, eig_vals.size + 1), cum_var_exp, where='mid', label = 'Cumulative explair
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc = 'best')
plt.tight_layout()
plt.show()
```

executed in 387ms, finished 01:56:42 2021-01-24



Above plot shows us that the first Principal component is giving us the 33.26 % and so on and the Cumulative variance is reaching 84 % at Principal Component 6.

Hence we will be selecting the 6 PC's for further study.

In [165]:

```
# we will be performing PCA on 6 components.

from sklearn.decomposition import PCA

pca = PCA(n_components=6)
data_reduced = pca.fit_transform(edu_scale)
data_reduced.transpose()
```

executed in 25ms, finished 02:03:54 2021-01-24

Out[165]:

```
array([[-1.60249937, -1.80467545, -1.60828257, ..., -0.57688267,
         6.570952  , -0.47739307],
       [ 0.99368301, -0.07041499, -1.38279212, ...,  0.01779846,
        -1.18493014,  1.04394672],
       [ 0.03004479,  2.12212749, -0.50151255, ...,  0.32216033,
         1.32596569, -1.42543834],
       [-1.00842234,  3.13894106, -0.03637346, ..., -0.58725875,
         0.07770725, -1.30027367],
       [-0.3668862 ,  2.45321185,  0.76599685, ...,  0.17522458,
         1.36851671,  0.72091762],
       [-0.6974759 ,  0.99485861, -1.02623665, ...,  0.50404281,
        -0.82274585,  1.05180967]])
```

In [166]:

```
pca.components_
```

executed in 19ms, finished 02:04:16 2021-01-24

Out[166]:

```
array([[ 2.62171542e-01,  2.30562461e-01,  1.89276397e-01,
         3.38874521e-01,  3.34690532e-01,  1.63293010e-01,
         2.24797091e-02,  2.83547285e-01,  2.44186588e-01,
         9.67082754e-02, -3.52299594e-02,  3.26410696e-01,
         3.23115980e-01, -1.63151642e-01,  1.86610828e-01,
         3.28955847e-01,  2.38822447e-01],
       [ 3.14136258e-01,  3.44623583e-01,  3.82813322e-01,
        -9.93191661e-02, -5.95055011e-02,  3.98636372e-01,
         3.57550046e-01, -2.51863617e-01, -1.31909124e-01,
         9.39739472e-02,  2.32439594e-01,  5.51390195e-02,
         4.30332048e-02,  2.59804556e-01, -2.57092552e-01,
        -1.60008951e-01, -1.67523664e-01],
       [-8.10177238e-02, -1.07658627e-01, -8.55296867e-02,
         7.88293847e-02,  5.07938248e-02, -7.37077846e-02,
        -4.03568699e-02, -1.49394795e-02,  2.11379165e-02,
         6.97121128e-01,  5.30972806e-01, -8.11134044e-02,
        -5.89785929e-02, -2.74150657e-01, -1.03715887e-01,
         1.84205687e-01, -2.45335837e-01],
       [ 9.87761696e-02,  1.18140435e-01,  9.30717485e-03,
        -3.69115031e-01, -4.16824361e-01,  1.39504394e-02,
         2.25351078e-01,  2.62975384e-01,  5.80894132e-01,
        -3.61562884e-02, -1.14982973e-01, -1.47260891e-01,
        -8.90079921e-02, -2.59486122e-01, -2.23982467e-01,
         2.13756140e-01, -3.61915064e-02],
       [ 2.19898082e-01,  1.89634937e-01,  1.62314823e-01,
         1.57211016e-01,  1.44449474e-01,  1.02728465e-01,
        -9.56790176e-02,  3.72750886e-02, -6.91080879e-02,
         3.54056654e-02, -4.75358319e-04, -5.50786546e-01,
        -5.90407136e-01, -1.42842546e-01,  1.28215768e-01,
        -2.24240836e-02,  3.56843227e-01],
       [ 2.18800314e-03, -1.65212819e-02, -6.80794254e-02,
        -8.88656816e-02, -2.76268982e-02, -5.16468643e-02,
        -2.45375726e-02, -2.03860463e-02,  2.37267409e-01,
         6.38604997e-01, -3.81495854e-01,  3.34444811e-03,
         3.54121293e-02,  4.68752603e-01,  1.25669419e-02,
        -2.31562325e-01,  3.13556243e-01]])
```

In [167]:

```
pca.explained_variance_ratio_
```

executed in 12ms, finished 02:04:29 2021-01-24

Out[167]:

```
array([0.33266084, 0.28755345, 0.06617164, 0.05898144, 0.05123893,
       0.04498639])
```

In [168]:

```
var=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=3)*100)
var
```

executed in 8ms, finished 02:04:51 2021-01-24

Out[168]:

```
array([33.3, 62.1, 68.7, 74.6, 79.7, 84.2])
```

In the above array , we can see that the First PCA explained the 33.3 % of the Variance within our Data and the first two explained 62.1, first three explains 68.7 and so on so after combining 6 PCA's Variance is explained 84.2 % of Data.

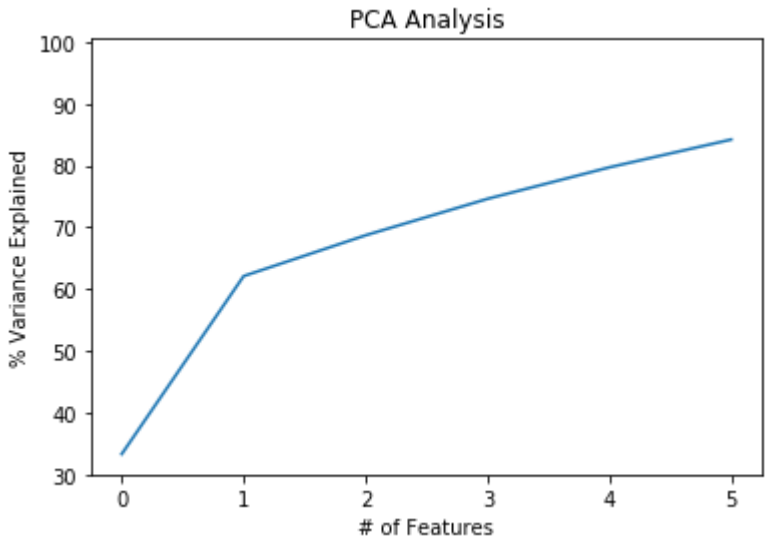
In [169]:

```
plt.ylabel('% Variance Explained')
plt.xlabel('# of Features')
plt.title('PCA Analysis')
plt.ylim(30,100.5)
plt.style.context('seaborn-whitegrid')
plt.plot(var)
```

executed in 300ms, finished 02:10:34 2021-01-24

Out[169]:

```
[<matplotlib.lines.Line2D at 0x239666ecf88>]
```



Number of Principal Components are 6.

In [170]:

```
# Exporting Data into a new Data Frame.

edu_pca = pd.DataFrame(pca.components_,columns=list(edu_scale))
edu_pca.head()
```

executed in 33ms, finished 02:14:16 2021-01-24

Out[170]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
0	0.262172	0.230562	0.189276	0.338875	0.334691	0.163293	0.022480	0.283547
1	0.314136	0.344624	0.382813	-0.099319	-0.059506	0.398636	0.357550	-0.251864
2	-0.081018	-0.107659	-0.085530	0.078829	0.050794	-0.073708	-0.040357	-0.014939
3	0.098776	0.118140	0.009307	-0.369115	-0.416824	0.013950	0.225351	0.262975
4	0.219898	0.189635	0.162315	0.157211	0.144449	0.102728	-0.095679	0.037275

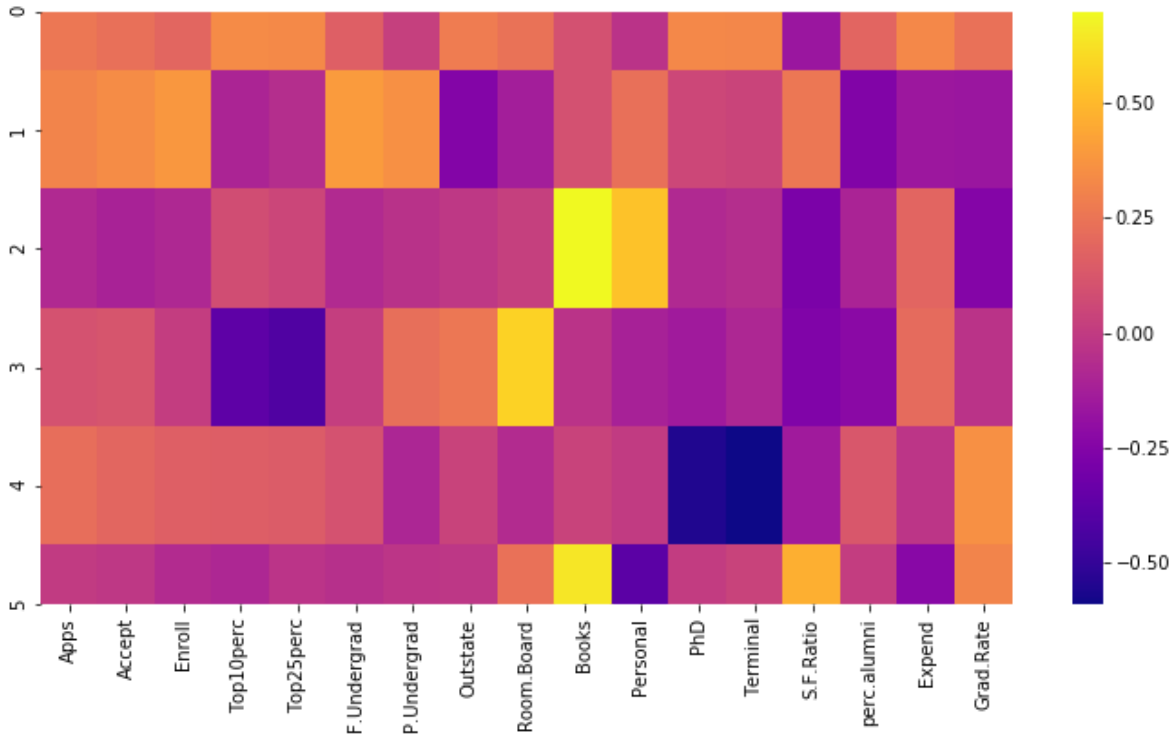
In [ ]:

```

```

In [172]:

```
# Heatmap for the new data frame to get the insights of PCA.

plt.figure(figsize=(12,6))
sns.heatmap(edu_pca,cmap='plasma');
```

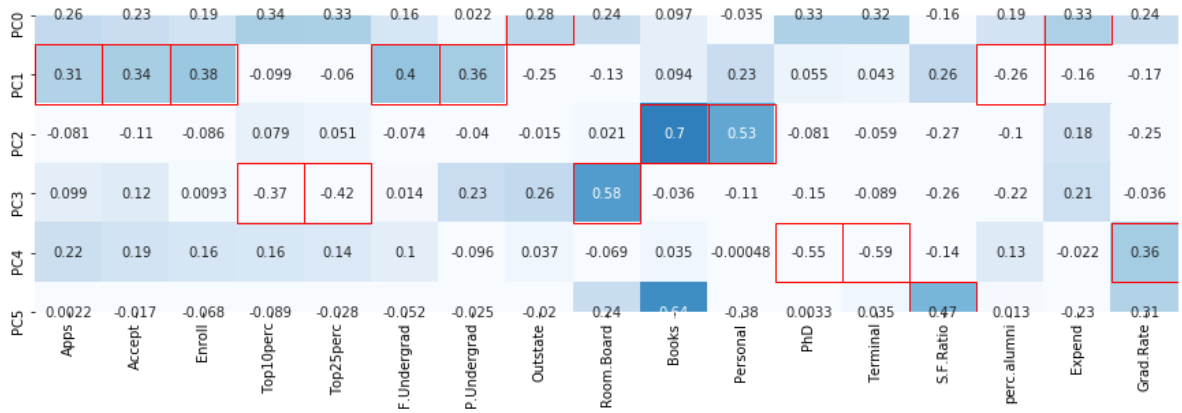
executed in 400ms, finished 02:16:30 2021-01-24



In [178]:

```
from matplotlib.patches import Rectangle

fig, ax = plt.subplots(figsize=(15,4), facecolor='w', edgecolor='k')
ax = sns.heatmap(edu_pca, annot=True, vmax=1, vmin=0, cmap='Blues', cbar=False, fmt='.2g',
                 yticklabels=['PC0','PC1','PC2','PC3','PC4','PC5'])

column_max = edu_pca.abs().idxmax(axis=0)

for col, variable in enumerate(edu_pca.columns):
    position = edu_pca.index.get_loc(column_max[variable])
    ax.add_patch(Rectangle((col, position), 1, 1, fill=False, edgecolor='red', lw=1))
```

executed in 1.08s, finished 02:23:50 2021-01-24



2.8) Mention the business implication of using the Principal Component Analysis for this case study.

Principal Component Analysis (PCA) is a method to reduce the multidimensional data into a lower dimension by keeping the highest Variance and not losing the Information from the Data set.

The given Dataset is a Post 12th Standard in which we had several columns and now after doing the PCA we have reduced the dimensions by keeping the highest Variance.

We Merge the entire Data into 6 principal Components which are PC0, PC1, PC2, PC3, PC4 and PC5.

FOR PC0:

In PC0, total variance is 33.3 which consists of students who are studying out of their home state and expenditure for those students for travelling so we club those into one single variable which is Transport

FOR PC1:

As per above Variance with Components we can create one new Variable which will name as a Admission Process in which we can get all the details of application received, applications accepted and applications processed and other variable can create is Graduates under which we can add the Full time Graduates and Part Time graduates.

Per alumni can merge into the Donation (we can keep or not).

FOR PC2:

In PC2, we can see that all the components with maximum variance are belongs to the Commercial side so we can create a New variable which is income slab which will help applicants to put their income and then Faculty will be able to decide the Room. Board, Books and Personal.

FOR PC3:

In PC3, we can merge both the components into a single variable which can name Percentage, in which Top 10% and Top 20% will differentiate the percentage of higher secondary class from the applicants.

FOR PC4:

In PC4, all the components have the large variance and all those can merge into a single variable which can name as Highly Qualified or Top qualification by doing we will be able to understand the Top qualifications of the faculties and also defined the Requirements for the NEW hiring for faculties.

FOR PC5:

In PC5, we can either keep the Student/Faculty ratio or we can use the new term Capacity which will add all the students from different colleges and compare the Faculty per student which will help to decide when to increase the Faculty based on the applications proceed and which faculty needs to increase for specific subjects and when to degrade or move some faculty into different courses or batches.

In [ ]:

By looking all the PC's we can give our inputs to the decision makers of the Business that mostly we need to increase the number of SEATS in every college so that maximum number of students can apply as we know that max applications which are applied is almost equal to the admission enrollment.

Also income variable is important to understand whether Colleges needs to spend on the Personal stuffs of students like upgraded hostels, books and other amenities.

By looking at the percentage scored by the students colleges can also slab their seats and can Ranked for top colleges because of the Highly qualified students.

Faculty is the most important in any institution as it decides quality of Education.

In [ ]: