In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from statsmodels.formula.api import ols
from statsmodels.stats.anova import _get_covariance,anova_lm
import os
```

Problem 1

Salaries of 40 individual, ANOVA test for Salary is hypothesized to depend on educational qualification and occupation as given data.

In [2]:

```
empdata=pd.read_csv("SalaryData.csv")
```

In [3]:

```
empdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Education   40 non-null     object
 1   Occupation  40 non-null     object
 2   Salary      40 non-null     int64 
dtypes: int64(1), object(2)
memory usage: 1.1+ KB
```

In [4]:

```
empdata.isnull().sum()
```

Out[4]:

```
Education     0
Occupation    0
Salary        0
dtype: int64
```

In [5]:

```
empdata.head(8)
```

Out[5]:

	Education	Occupation	Salary
0	Doctorate	Adm-clerical	153197
1	Doctorate	Adm-clerical	115945
2	Doctorate	Adm-clerical	175935
3	Doctorate	Adm-clerical	220754
4	Doctorate	Sales	170769
5	Doctorate	Sales	219420
6	Doctorate	Sales	237920
7	Doctorate	Sales	160540

In [6]:

```
pd.DataFrame(empdata.groupby('Education').Occupation.value_counts())
```

Out[6]:

		Occupation
Education	**Occupation**	
Bachelors	**Exec-managerial**	4
	Prof-specialty	4
	Sales	4
	Adm-clerical	3
Doctorate	**Prof-specialty**	6
	Sales	5
	Adm-clerical	4
	Exec-managerial	1
HS-grad	**Adm-clerical**	3
	Prof-specialty	3
	Sales	3

In [7]:

```
empdata.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Salary	40.0	162186.875	64860.407506	50103.0	99897.5	169100.0	214440.75	260151.0

Q1.1 State the null and the alternate hypothesis for conducting one-way ANOVA for both Education and Occupation individually.

a.*Education*

Null and alternate hypothesis when salary dependent on Education .

Ho:(Null Hypothesis)- The mean salary is same for all the Three levels of education ..

Ha:(Alternate Hypothesis)-The mean salary is different for atleast one level of education ..

B.*Occupation*

Null and alternate hypothesis when salary dependent of occupation .

Ho:(Null Hypothesis)- The mean salary is same for all the four level of occupation .

Ha:(Alternate Hypothesis)- The mean salary is different for at least one level occupation .

Q1.2 Perform one-way ANOVA for Education with respect to the variable ‘Salary’. State whether the null hypothesis is accepted or rejected based on the ANOVA results.

In [8]:

```
formula = 'Salary ~ C(Education)'
model = ols(formula, empdata).fit()
aov_table = anova_lm(model)
print(aov_table)
```

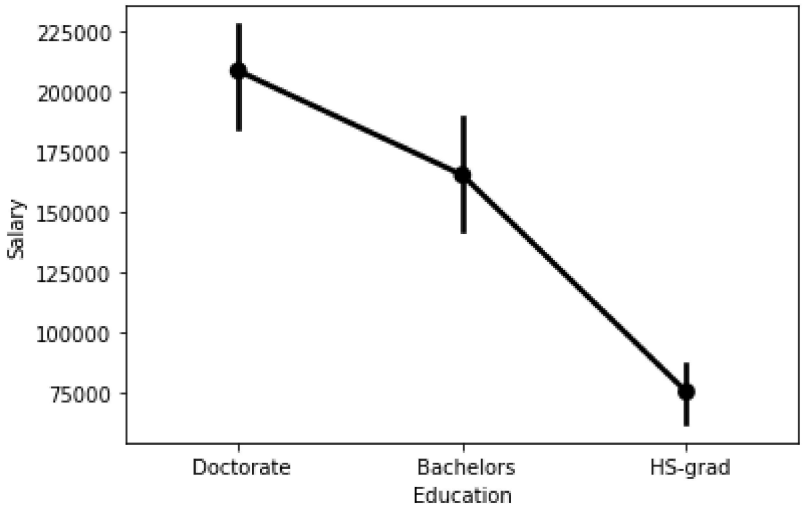
```
                df        sum_sq       mean_sq          F        PR(>F)
C(Education)   2.0  1.026955e+11  5.134773e+10  30.95628  1.257709e-08
Residual      37.0  6.137256e+10  1.658718e+09       NaN           NaN
```

In [9]:

```
sns.pointplot(x='Education', y='Salary', data=empdata,color="black")
```

Out[9]:

```
<AxesSubplot:xlabel='Education', ylabel='Salary'>
```

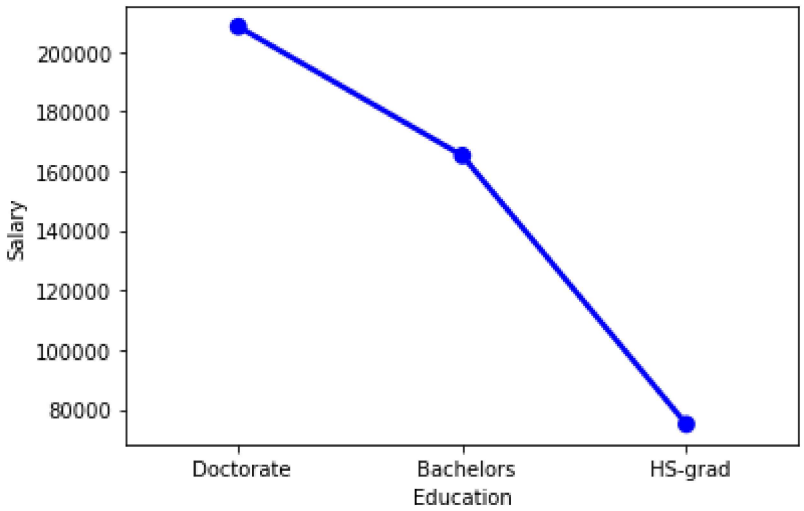


In [10]:

```
sns.pointplot(x='Education', y='Salary', data=empdata, ci=None,color="blue")
```

Out[10]:

```
<AxesSubplot:xlabel='Education', ylabel='Salary'>
```



Q1.3 Perform one-way ANOVA for variable Occupation with respect to the variable 'Salary'. State whether the

null hypothesis is accepted or rejected based on the ANOVA results.

In [11]:

```
formula = 'Salary ~ C(Occupation)'
model = ols(formula, empdata).fit()
aov_table = anova_lm(model)
print(aov_table)
```

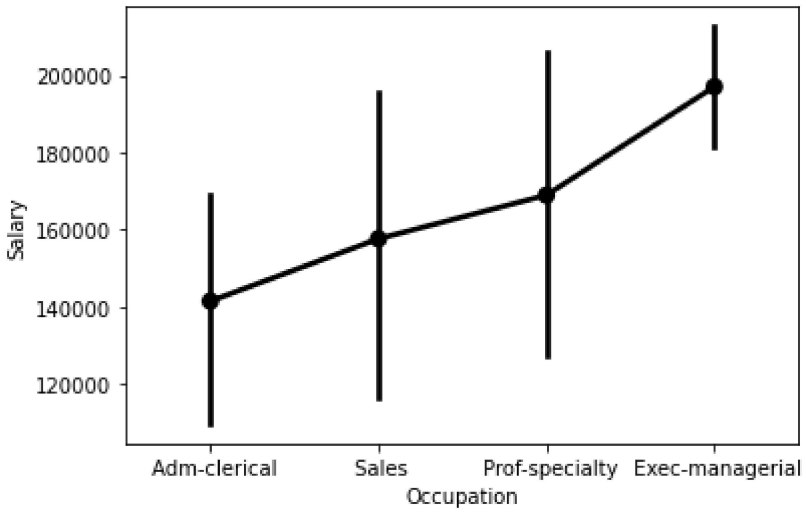
```
                 df        sum_sq       mean_sq          F    PR(>F)
C(Occupation)   3.0  1.125878e+10  3.752928e+09   0.884144  0.458508
Residual       36.0  1.528092e+11  4.244701e+09        NaN       NaN
```

In [12]:

```
sns.pointplot(x='Occupation', y='Salary', data=empdata,color="black")
```

Out[12]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Salary'>
```

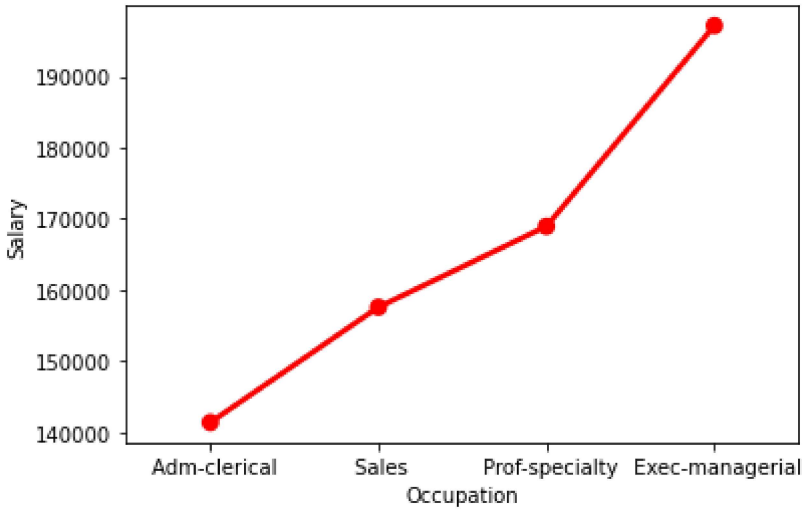


In [13]:

```
sns.pointplot(x='Occupation', y='Salary', data=empdata, ci=None,color="red")
```

Out[13]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Salary'>
```



In [ ]:

```
```

Q 1.4 If the null hypothesis is rejected in either (1.2) or in (1.3), find out which class means are significantly different. Interpret the result.

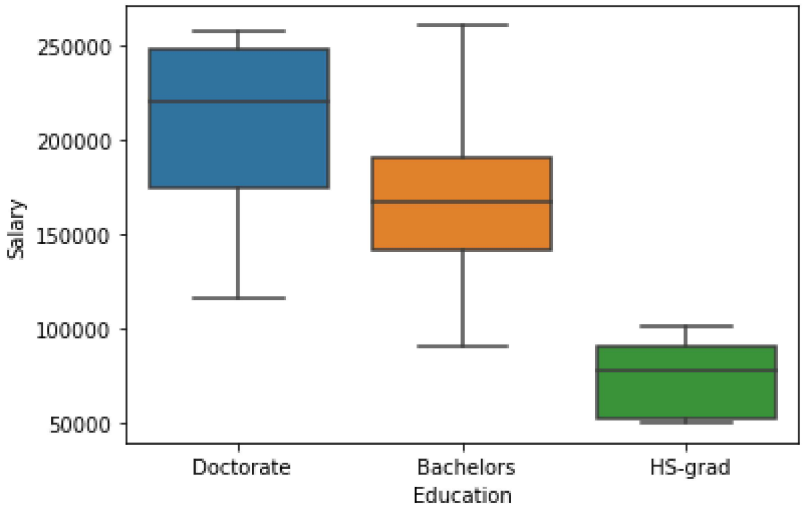
Ans:- 3 mean are differnet below mentioned boxplot

In [14]:

```
sns.boxplot(x='Education',y='Salary',data=empdata)
```

Out[14]:

```
<AxesSubplot:xlabel='Education', ylabel='Salary'>
```



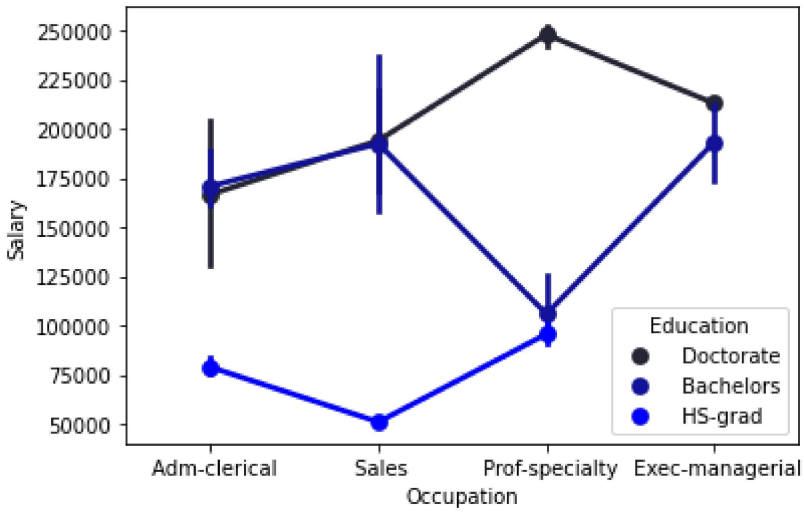
Q 1.5 What is the interaction between the two treatments? Analyze the effects of one variable on the other (Education and Occupation) with the help of an interaction plot.

In [15]:

```
sns.pointplot(x='Occupation', y='Salary',hue='Education',data=empdata,color="blue")
```

Out[15]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Salary'>
```

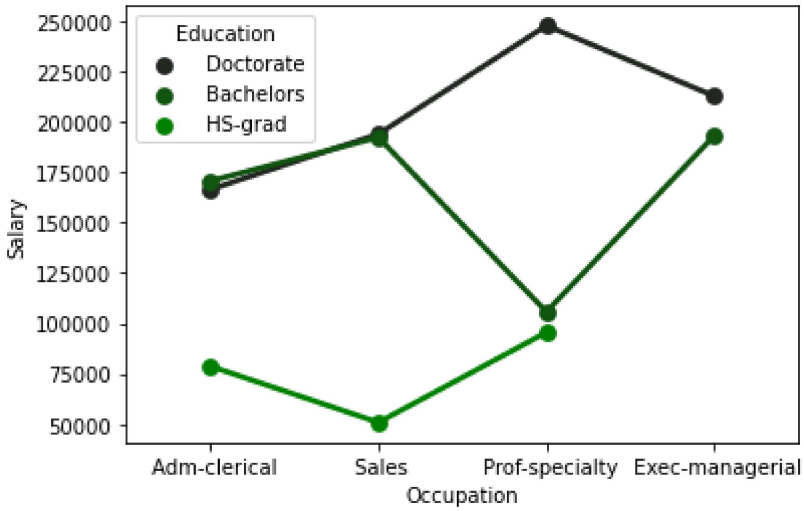


In [16]:

```
sns.pointplot(x='Occupation', y='Salary',hue='Education',data=empdata,ci=None,color="green"
```

Out[16]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Salary'>
```



In [ ]:

Q 1.6 Perform a two-way ANOVA based on the Education and Occupation (along with their interaction Education*Occupation) with the variable 'Salary'. State the null and alternative hypotheses and state your results. How will you interpret this result?

In [17]:

```
formula = 'Salary ~ C(Education) + C(Occupation) + C(Education):C(Occupation)'
model = ols(formula, empdata).fit()
aov_table = anova_lm(model)
(aov_table)
```

Out[17]:

	df	sum_sq	mean_sq	F	PR(>F)
C(Education)	2.0	1.026955e+11	5.134773e+10	72.211958	5.466264e-12
C(Occupation)	3.0	5.519946e+09	1.839982e+09	2.587626	7.211580e-02
C(Education):C(Occupation)	6.0	3.634909e+10	6.058182e+09	8.519815	2.232500e-05
Residual	29.0	2.062102e+10	7.110697e+08	NaN	NaN

Q 1.7 Explain the business implications of performing ANOVA for this particular case study.

Ans:- mention in report

..... Problem 2.....

Education - Post 12th Standard, expected to do a Principal Component Analysis for this case study according to the instructions

In [18]:

```
edu_data=pd.read_csv("Education+-+Post+12th+Standard.csv")
```

Q2.1 Perform Exploratory Data Analysis [both univariate and multivariate analysis to be performed]. What insight do you draw from the EDA?

In [19]:

```
edu_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Names        777 non-null    object 
 1   Apps         777 non-null    int64  
 2   Accept       777 non-null    int64  
 3   Enroll       777 non-null    int64  
 4   Top10perc    777 non-null    int64  
 5   Top25perc    777 non-null    int64  
 6   F.Undergrad  777 non-null    int64  
 7   P.Undergrad  777 non-null    int64  
 8   Outstate     777 non-null    int64  
 9   Room.Board   777 non-null    int64  
 10  Books        777 non-null    int64  
 11  Personal     777 non-null    int64  
 12  PhD          777 non-null    int64  
 13  Terminal     777 non-null    int64  
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64  
 16  Expend       777 non-null    int64  
 17  Grad.Rate    777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 109.4+ KB
```

In [20]:

```
edu_data.shape
```

Out[20]:

```
(777, 18)
```

Check for missing value

In [21]:

```
edu_data.isnull().sum()
```

Out[21]:

```
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 [22]:

```
edu_data.head()
```

Out[22]:

	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

In [23]:

```
edu_data.Names.value_counts()
```

Out[23]:

```
Northeast Missouri State University    1
Saint Joseph's College IN              1
Benedictine College                    1
Louisiana College                      1
Mankato State University               1
                                      ..
Knox College                           1
Willamette University                  1
Florida International University       1
Southwest State University             1
College of Mount St. Vincent           1
Name: Names, Length: 777, dtype: int64
```

In [24]:

```
edu_data.describe().T
```

Out[24]:

	count	mean	std	min	25%	50%	75%	max
Apps	777.0	3001.638353	3870.201484	81.0	776.0	1558.0	3624.0	48094.0
Accept	777.0	2018.804376	2451.113971	72.0	604.0	1110.0	2424.0	26330.0
Enroll	777.0	779.972973	929.176190	35.0	242.0	434.0	902.0	6392.0
Top10perc	777.0	27.558559	17.640364	1.0	15.0	23.0	35.0	96.0
Top25perc	777.0	55.796654	19.804778	9.0	41.0	54.0	69.0	100.0
F.Undergrad	777.0	3699.907336	4850.420531	139.0	992.0	1707.0	4005.0	31643.0
P.Undergrad	777.0	855.298584	1522.431887	1.0	95.0	353.0	967.0	21836.0
Outstate	777.0	10440.669241	4023.016484	2340.0	7320.0	9990.0	12925.0	21700.0
Room.Board	777.0	4357.526384	1096.696416	1780.0	3597.0	4200.0	5050.0	8124.0
Books	777.0	549.380952	165.105360	96.0	470.0	500.0	600.0	2340.0
Personal	777.0	1340.642214	677.071454	250.0	850.0	1200.0	1700.0	6800.0
PhD	777.0	72.660232	16.328155	8.0	62.0	75.0	85.0	103.0
Terminal	777.0	79.702703	14.722359	24.0	71.0	82.0	92.0	100.0
S.F.Ratio	777.0	14.089704	3.958349	2.5	11.5	13.6	16.5	39.8
perc.alumni	777.0	22.743887	12.391801	0.0	13.0	21.0	31.0	64.0
Expend	777.0	9660.171171	5221.768440	3186.0	6751.0	8377.0	10830.0	56233.0
Grad.Rate	777.0	65.463320	17.177710	10.0	53.0	65.0	78.0	118.0

Check for Duplicate records

In [25]:

```
dups = edu_data.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 0
```

Data visulitiation

Univariate – Analysis

In [26]:

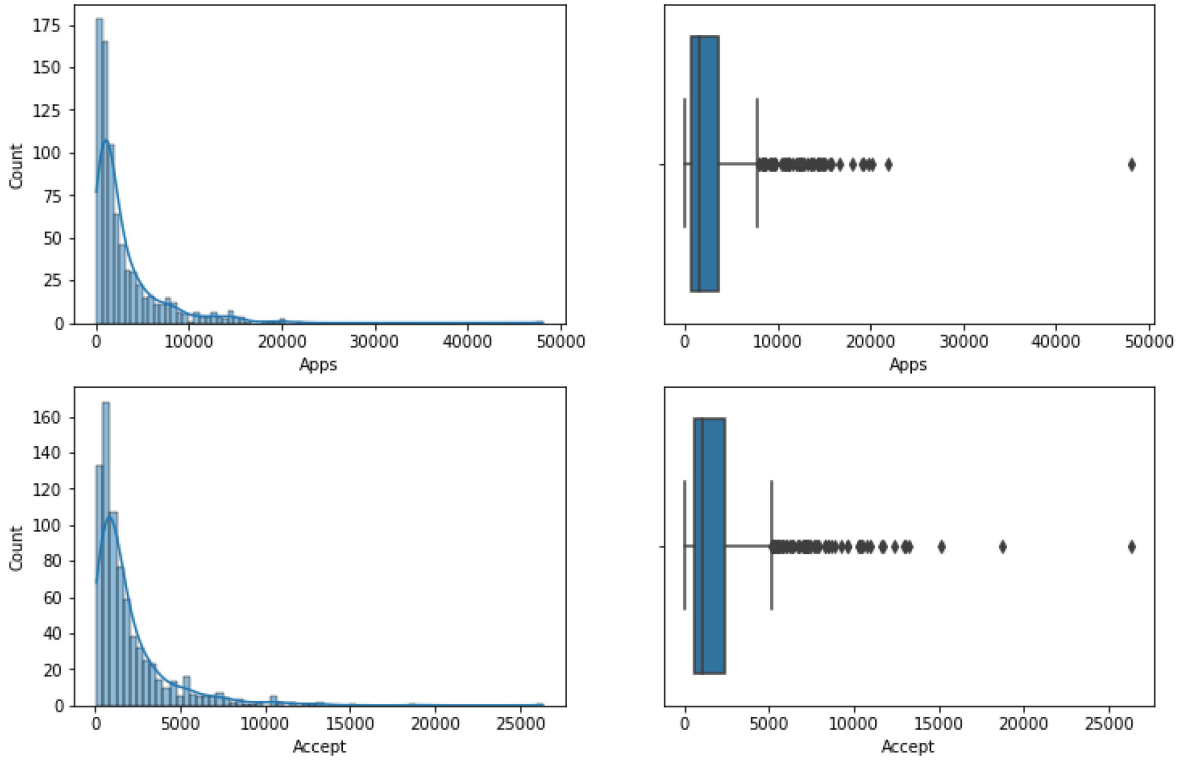
```
edu_data[['Apps','Accept']].describe()
```

Out[26]:

	Apps	Accept
count	777.000000	777.000000
mean	3001.638353	2018.804376
std	3870.201484	2451.113971
min	81.000000	72.000000
25%	776.000000	604.000000
50%	1558.000000	1110.000000
75%	3624.000000	2424.000000
max	48094.000000	26330.000000

In [27]:

```
fig,axes=plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(12,8)
sns.histplot(edu_data['Apps'],kde=True,ax=axes[0][0])
sns.boxplot(x='Apps',data=edu_data,ax=axes[0][1])
sns.histplot(edu_data['Accept'],kde=True,ax=axes[1][0])
sns.boxplot(x='Accept',data=edu_data,ax=axes[1][1])
plt.show()
```



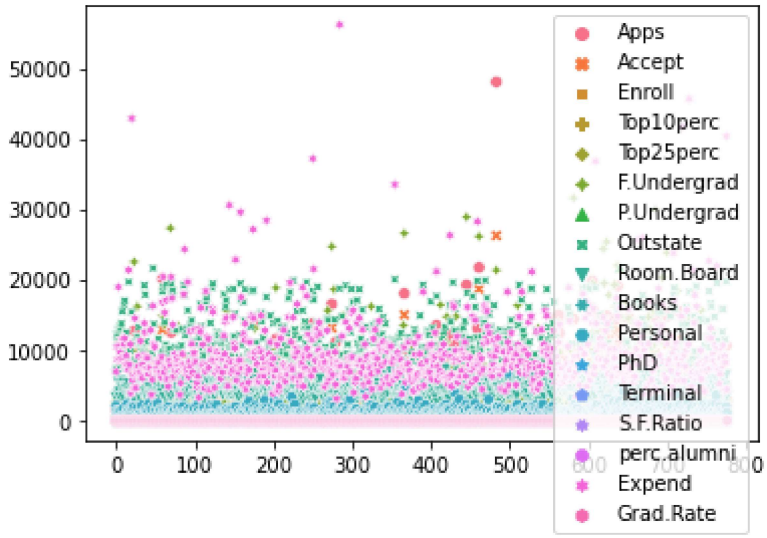
multivariate − analysis

In [28]:

```
sns.scatterplot(data=edu_data)
```

Out[28]:

<AxesSubplot:>

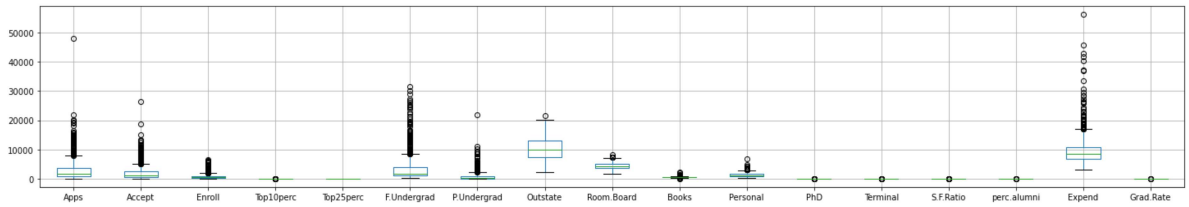


In [29]:

```
edu_data.boxplot(figsize=(25,4))
```

Out[29]:

<AxesSubplot:>

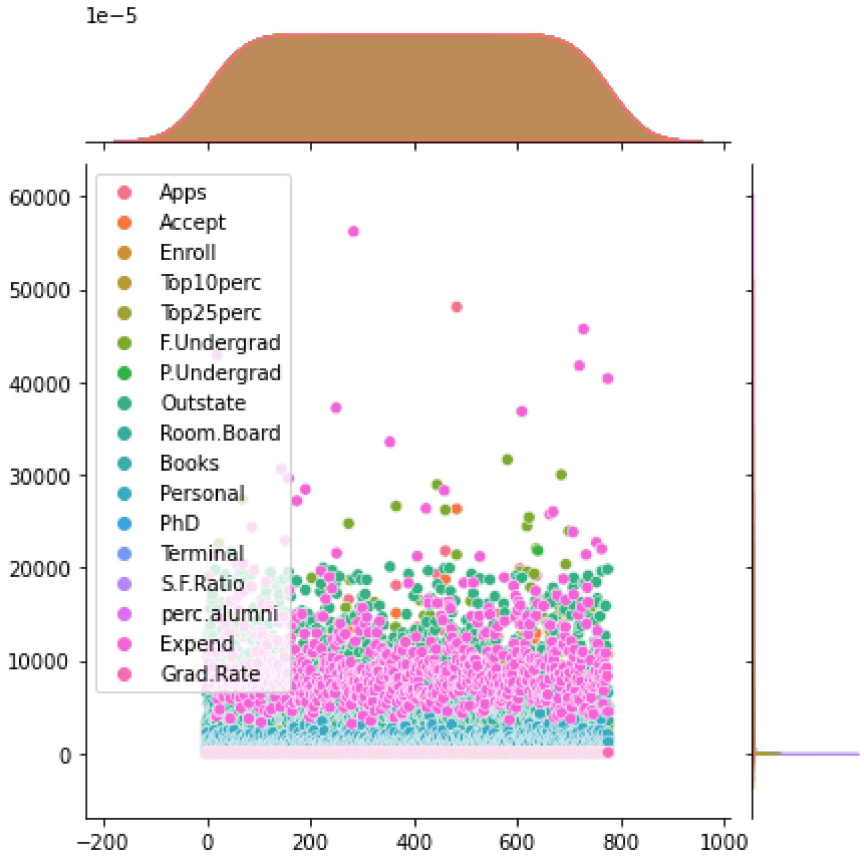


In [30]:

```
sns.jointplot(data=edu_data)
```

Out[30]:

```
<seaborn.axisgrid.JointGrid at 0x12fcbc94580>
```

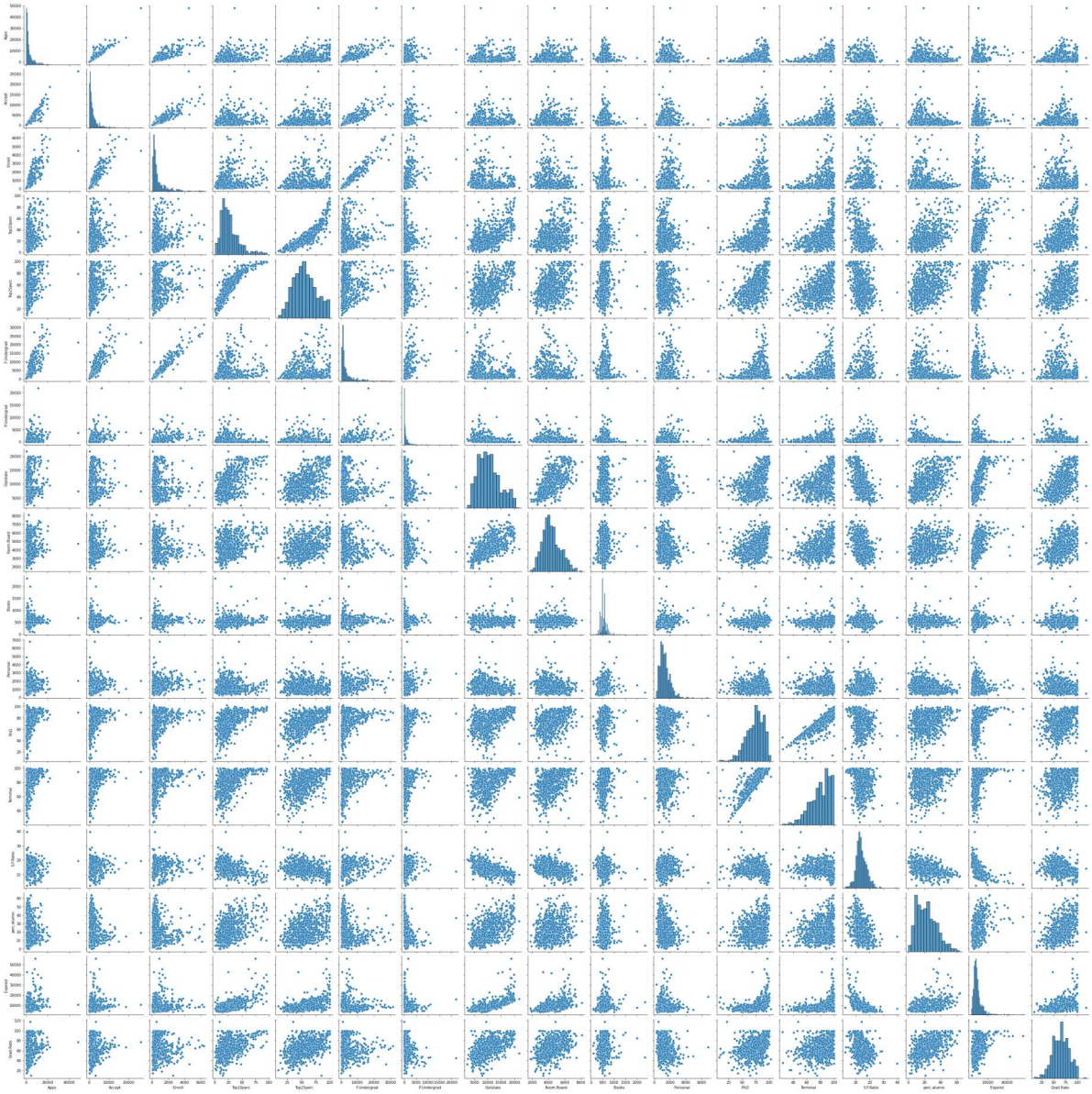


In [31]:

```
sns.pairplot(edu_data)
```

Out[31]:

```
<seaborn.axisgrid.PairGrid at 0x12fcc073b50>
```



In [32]:

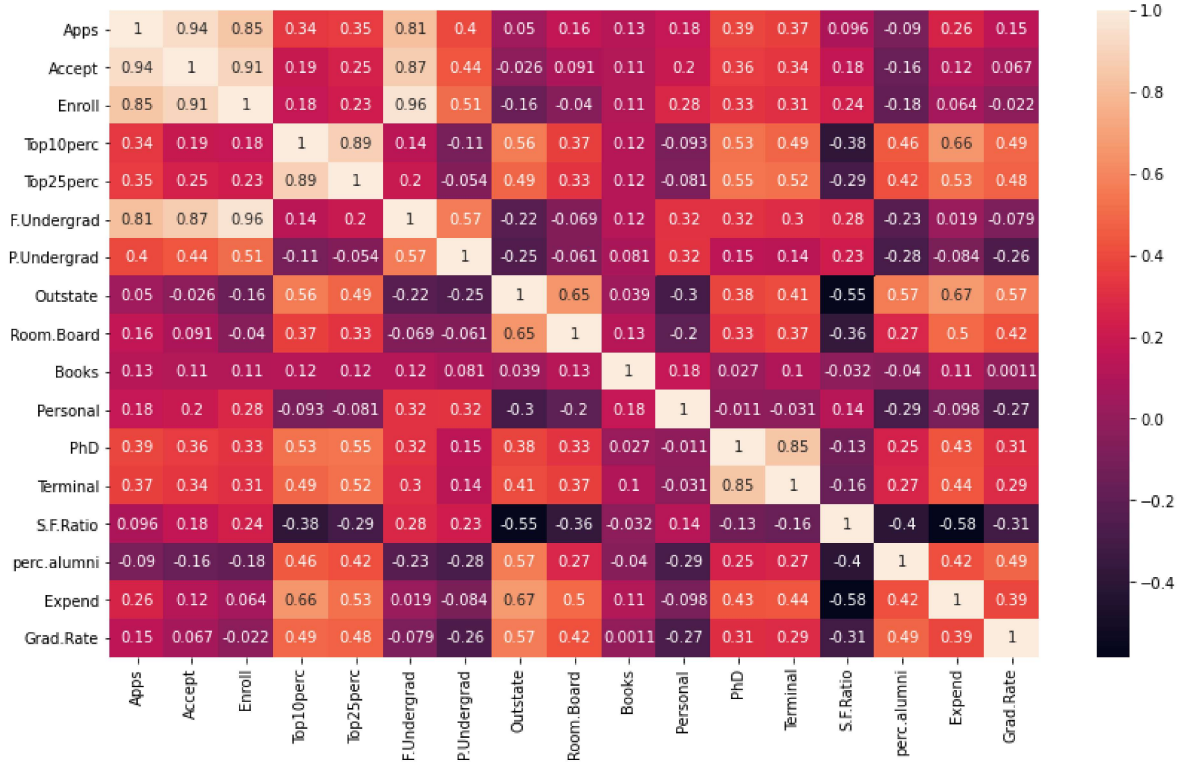
```
corr=edu_data.corr()
```

In [33]:

```
plt.figure(figsize=(14,8))
sns.heatmap(corr,annot=True)
```

Out[33]:

```
<AxesSubplot:>
```



Q 2.2 Is scaling necessary for PCA in this case? Give justification and perform scaling.

In [34]:

```
edu_data_num=edu_data.select_dtypes(['int64','float64'])
edu_data_num.columns
```

Out[34]:

```
Index(['Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F.Undergrad',
       'P.Undergrad', 'Outstate', 'Room.Board', 'Books', 'Personal', 'PhD',
       'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expend', 'Grad.Rate'],
      dtype='object')
```

In [35]:

```
from scipy.stats import zscore
data_scaled=edu_data_num.apply(zscore)
```

In [36]:

```
data_scaled.head(12)
```

Out[36]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
0	-0.346882	-0.321205	-0.063509	-0.258583	-0.191827	-0.168116	-0.209207	-0.746356
1	-0.210884	-0.038703	-0.288584	-0.655656	-1.353911	-0.209788	0.244307	0.457496
2	-0.406866	-0.376318	-0.478121	-0.315307	-0.292878	-0.549565	-0.497090	0.201305
3	-0.668261	-0.681682	-0.692427	1.840231	1.677612	-0.658079	-0.520752	0.626633
4	-0.726176	-0.764555	-0.780735	-0.655656	-0.596031	-0.711924	0.009005	-0.716508
5	-0.624307	-0.628611	-0.669812	0.592287	0.313426	-0.623421	-0.535212	0.760947
6	-0.684808	-0.685356	-0.729043	-0.598931	-0.545505	-0.677472	-0.410988	0.708713
7	-0.285088	-0.121984	-0.313353	0.535563	0.616579	-0.434450	-0.541127	0.852479
8	-0.507700	-0.481644	-0.595505	0.138490	0.363952	-0.562562	-0.361036	1.282036
9	-0.625600	-0.620854	-0.654735	-0.372032	-0.596031	-0.598459	-0.510893	0.006798
10	-0.328266	-0.242415	-0.331661	0.535563	0.970257	-0.385763	-0.489860	1.519075
11	-0.090399	-0.048501	-0.318738	0.932636	1.071307	-0.411138	-0.533240	1.651399

concususing :-scaling is necesri becouse data prameter is hay after scaling data size and parameter is also redused .

Q 2.3 Comment on the comparison between the covariance and the correlation matrices from this data. [on scaled data]

In [37]:

```
cov_matrix = pd.DataFrame(np.cov(data_scaled))
cov_matrix
```

Out[37]:

	0	1	2	3	4	5	6	7	
0	0.331066	-0.154986	-0.029506	-0.309064	0.062573	-0.115809	-0.098542	-0.156053	-0.1
1	-0.154986	1.567628	0.166152	-0.429341	0.242678	-0.074895	-0.134492	-0.472318	-0.2
2	-0.029506	0.166152	0.171533	0.058161	-0.126113	0.023802	-0.041464	0.032429	0.0
3	-0.309064	-0.429341	0.058161	1.230082	0.012613	0.363048	0.344101	0.550846	0.4
4	0.062573	0.242678	-0.126113	0.012613	0.853935	0.053340	0.066773	-0.212979	-0.3
...	...	...	...	...	...	...	...	...	
772	0.222201	0.340347	0.009762	-0.684302	0.205923	-0.267525	-0.176610	-0.331819	-0.3
773	-0.102925	0.184185	0.023202	-0.012337	-0.148581	-0.042983	0.090610	0.041896	0.0
774	-0.026547	0.013808	-0.020525	0.060832	0.145036	0.024400	-0.110479	-0.025924	-0.1
775	-0.457125	0.039471	0.183534	1.507365	-0.092929	0.550588	0.237833	0.479924	0.6
776	0.137730	-0.240147	-0.044969	-0.292290	-0.389124	-0.166815	-0.176986	-0.039487	-0.0

777 rows × 777 columns

In [38]:

```
corr_matrix=data_scaled.corr()
corr_matrix
```

Out[38]:

	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 [ ]:

```

```

Q 2.4 Check the dataset for outliers before and after scaling. What insight do you derive here?

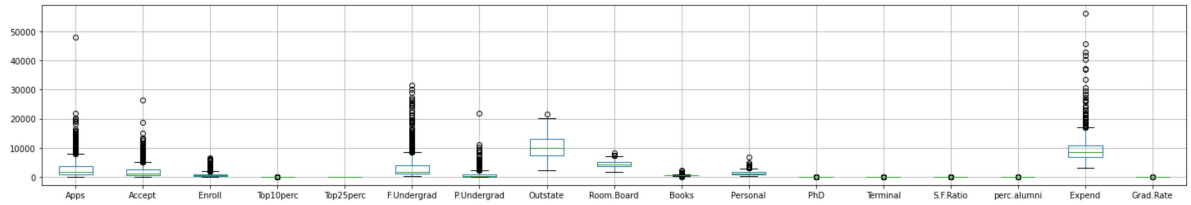
Before Scaling

In [39]:

```
edu_data.boxplot(figsize=(25,4))
```

Out[39]:

<AxesSubplot:>



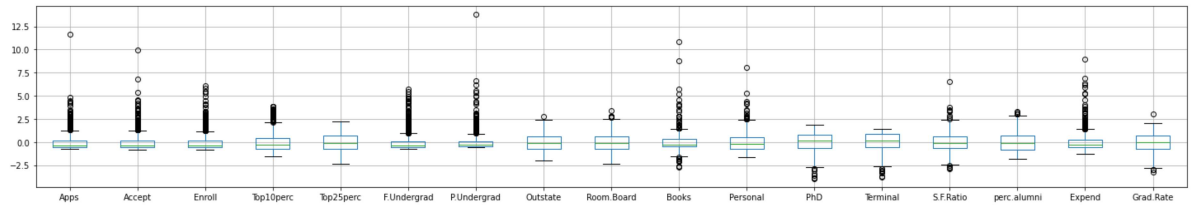
After Scaling

In [40]:

```
data_scaled.boxplot(figsize=(25,4))
```

Out[40]:

<AxesSubplot:>



pramer is reduest and all mean value is near by zero ...

Q 2.5 Extract the eigenvalues and eigenvectors.[print both]

In [41]:

```
eig_vals, eig_vecs = np.linalg.eig(cov_matrix )
print('Eigen Vectors \n%s', pd.DataFrame(eig_vecs))
```

```
Eigen Vectors
%s                     0                     1                     2    \
0   -0.022919+0.000000j -0.018999+0.000000j -0.015433+0.000000j
1   -0.007299+0.000000j -0.056117+0.000000j  0.081335+0.000000j
2    0.007012+0.000000j -0.008281+0.000000j -0.005125+0.000000j
3    0.060368+0.000000j  0.033981+0.000000j  0.019605+0.000000j
4   -0.015208+0.000000j -0.025249+0.000000j  0.079143+0.000000j
..                  ...                 ...                 ...
772 -0.041957+0.000000j -0.036483+0.000000j -0.025293+0.000000j
773  0.011221+0.000000j -0.016373+0.000000j -0.006555+0.000000j
774 -0.003378+0.000000j  0.003735+0.000000j  0.004142+0.000000j
775  0.085569+0.000000j  0.054337+0.000000j  0.096314+0.000000j
776 -0.009372+0.000000j -0.013009+0.000000j -0.063788+0.000000j

                    3                     4                     5    \
0    0.029136+0.000000j -0.028335+0.000000j  0.016837+0.000000j
1   -0.117802+0.000000j  0.042004+0.000000j  0.026261+0.000000j
2   -0.022541+0.000000j -0.015303+0.000000j  0.039053+0.000000j
3    0.057081+0.000000j -0.014407+0.000000j  0.007391+0.000000j
4    0.057174+0.000000j  0.018533+0.000000j -0.051403+0.000000j
..                  ...                 ...                 ...
772 -0.010533+0.000000j  0.032081+0.000000j  0.006716+0.000000j
773 -0.024601+0.000000j  0.017332+0.000000j  0.007922+0.000000j
774 -0.001260+0.000000j -0.019143+0.000000j -0.039773+0.000000j
775 -0.007773+0.000000j -0.044637+0.000000j  0.058639+0.000000j
776 -0.014326+0.000000j -0.035182+0.000000j  0.001677+0.000000j

                    6                     7                     8    \
0    0.048905+0.000000j  0.004521+0.000000j  0.004458+0.000000j
1   -0.000689+0.000000j  0.052287+0.000000j  0.027513+0.000000j
2   -0.019308+0.000000j  0.001863+0.000000j  0.039684+0.000000j
3   -0.018605+0.000000j  0.038008+0.000000j  0.036335+0.000000j
4   -0.014092+0.000000j  0.028938+0.000000j  0.059706+0.000000j
..                  ...                 ...                 ...
772 -0.020332+0.000000j  0.030473+0.000000j  0.043146+0.000000j
773 -0.010011+0.000000j -0.022322+0.000000j -0.035166+0.000000j
774 -0.019735+0.000000j  0.029140+0.000000j  0.029352+0.000000j
775  0.023124+0.000000j  0.014897+0.000000j  0.065719+0.000000j
776 -0.001818+0.000000j  0.003581+0.000000j -0.094044+0.000000j

                    9    ...                   767                   768  \
0   -0.021455+0.000000j  ...  0.008068-0.002792j  0.008068+0.002792j
1    0.036213+0.000000j  ... -0.001143-0.000264j -0.001143+0.000264j
2    0.014708+0.000000j  ... -0.000596-0.001459j -0.000596+0.001459j
3    0.024260+0.000000j  ...  0.000781+0.001304j  0.000781-0.001304j
4   -0.004120+0.000000j  ...  0.001391+0.000826j  0.001391-0.000826j
..                  ...  ...                 ...                 ...
772 -0.008428+0.000000j  ...  0.023082+0.002294j  0.023082-0.002294j
773  0.002770+0.000000j  ...  0.014471+0.036856j  0.014471-0.036856j
774  0.023380+0.000000j  ... -0.005254+0.028178j -0.005254-0.028178j
775 -0.104637+0.000000j  ... -0.007371+0.007649j -0.007371-0.007649j
776 -0.018337+0.000000j  ... -0.020459+0.011519j -0.020459-0.011519j

                    769                   770                   771  \
```

```
0   -0.007951-0.002831j -0.007951+0.002831j  0.007352+0.001670j
1   -0.000423-0.000458j -0.000423+0.000458j -0.001891+0.001969j
2   -0.000964-0.002609j -0.000964+0.002609j  0.002726+0.001076j
3   -0.000863-0.001082j -0.000863+0.001082j -0.003566+0.000763j
4    0.000774+0.000855j  0.000774-0.000855j  0.000555+0.001506j
..                  ...                 ...                 ...
772  0.001858+0.020462j  0.001858-0.020462j  0.000577-0.012412j
773 -0.103666-0.002374j -0.103666+0.002374j -0.002946+0.073773j
774  0.080498+0.019424j  0.080498-0.019424j  0.052564+0.015294j
775  0.011129-0.003143j  0.011129+0.003143j  0.016798-0.002564j
776 -0.006097+0.020388j -0.006097-0.020388j  0.013968-0.013061j

                   772                 773                 774  \
0    0.007352-0.001670j  0.008170+0.000000j  0.000195-0.004662j
1   -0.001891-0.001969j  0.002371+0.000000j  0.000551-0.000394j
2    0.002726-0.001076j -0.001065+0.000000j -0.001532-0.001815j
3   -0.003566-0.000763j  0.005522+0.000000j  0.002407+0.001686j
4    0.000555-0.001506j -0.001961+0.000000j  0.003296+0.001043j
..                  ...                 ...                 ...
772  0.000577+0.012412j -0.016485+0.000000j  0.008445+0.019849j
773 -0.002946-0.073773j  0.016255+0.000000j  0.036383+0.002086j
774  0.052564-0.015294j -0.027824+0.000000j  0.008322+0.000188j
775  0.016798+0.002564j -0.025413+0.000000j -0.016892-0.006234j
776  0.013968+0.013061j  0.034376+0.000000j -0.012886+0.012447j

                   775                 776
0    0.000195+0.004662j  0.005225+0.000000j
1    0.000551+0.000394j  0.000967+0.000000j
2   -0.001532+0.001815j -0.001197+0.000000j
3    0.002407-0.001686j  0.002737+0.000000j
4    0.003296-0.001043j -0.000668+0.000000j
..                  ...                 ...
772  0.008445-0.019849j -0.006459+0.000000j
773  0.036383-0.002086j  0.112103+0.000000j
774  0.008322-0.000188j -0.114487+0.000000j
775 -0.016892+0.006234j -0.017905+0.000000j
776 -0.012886-0.012447j  0.003802+0.000000j

[777 rows x 777 columns]
```

In [42]:

```
print('\n Eigen Values \n%s', pd.DataFrame(eig_vals))
```

```
 Eigen Values
%s                                0
0    2.250938e+02+0.000000e+00j
1    8.626531e+01+0.000000e+00j
2    5.341962e+01+0.000000e+00j
3    4.858358e+01+0.000000e+00j
4    4.525109e+01+0.000000e+00j
..                          ...
772  8.482920e-18-5.859576e-17j
773  3.836886e-17+0.000000e+00j
774 -2.116932e-17+2.654029e-17j
775 -2.116932e-17-2.654029e-17j
776 -2.688490e-17+0.000000e+00j

[777 rows x 1 columns]
```

Q 2.6 Perform PCA and export the data of the Principal Component (eigenvectors) into a data frame with the original features

In [43]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=17)
data_reduced = pd.DataFrame(pca.fit_transform(data_scaled))
data_reduced.transpose()
```

Out[43]:

	0	1	2	3	4	5	6	7	
0	-1.592855	-2.192402	-1.430964	2.855557	-2.212008	-0.571665	0.241952	1.750474	0.76
1	0.767334	-0.578830	-1.092819	-2.630612	0.021631	-1.496325	-1.506368	-1.461412	-1.98
2	-0.101074	2.278798	-0.438093	0.141722	2.387030	0.024354	0.234194	-1.026589	-1.42
3	-0.921749	3.588918	0.677241	-1.295486	-1.114538	0.066944	-1.142024	-0.981184	-0.07
4	-0.743975	1.059997	-0.369613	-0.183837	0.684451	-0.376261	1.546983	0.217044	0.58
5	-0.298306	-0.177137	-0.960592	-1.059508	0.004918	-0.668344	-0.009995	0.222924	-0.65
6	0.638443	0.236753	-0.248276	-1.249356	-2.159220	-1.609835	0.590933	0.038169	-0.21
7	-0.879386	0.046925	0.308740	-0.147694	-0.624413	-0.529391	-0.329858	0.173929	-0.27
8	0.093084	1.113780	-0.105452	0.378997	-0.160383	0.263527	-0.653409	-0.352525	0.22
9	0.048593	0.965154	0.640660	0.461244	0.363428	-0.694509	0.157126	0.343354	-0.62
10	0.399747	-0.212509	-0.154993	-0.420651	-0.153339	-0.100298	-0.604160	-0.109102	0.23
11	-0.089690	0.097239	-0.344731	0.687143	-0.050552	-1.093768	-0.027715	-0.097020	-0.68
12	-0.052098	-0.243518	0.097551	-0.075461	0.267207	-0.131218	-0.029158	0.026668	-0.21
13	0.180140	-0.744204	0.227527	-0.003380	-0.614409	-0.094452	-0.252486	0.255895	0.03
14	0.001752	0.103710	-0.022558	-0.073181	-0.273993	0.221006	-0.004814	0.008120	-0.16
15	-0.093140	-0.050256	-0.004053	-0.191546	0.010653	0.057024	0.103877	-0.139536	0.09
16	0.093552	-0.174057	0.003759	-0.175254	0.048344	-0.038585	0.063393	-0.053251	0.00

17 rows × 777 columns

In [44]:

```
pd.DataFrame(pca.components_)
```

Out[44]:

	0	1	2	3	4	5	6	7	
0	0.248766	0.207602	0.176304	0.354274	0.344001	0.154641	0.026443	0.294736	0.24
1	0.331598	0.372117	0.403724	-0.082412	-0.044779	0.417674	0.315088	-0.249644	-0.13
2	-0.063092	-0.101249	-0.082986	0.035056	-0.024148	-0.061393	0.139682	0.046599	0.14
3	0.281311	0.267817	0.161827	-0.051547	-0.109767	0.100412	-0.158558	0.131291	0.18
4	0.005741	0.055786	-0.055694	-0.395434	-0.426534	-0.043454	0.302385	0.222532	0.56
5	-0.016237	0.007535	-0.042558	-0.052693	0.033092	-0.043454	-0.191199	-0.030000	0.16
6	-0.042486	-0.012950	-0.027693	-0.161332	-0.118486	-0.025076	0.061042	0.108529	0.20
7	-0.103090	-0.056271	0.058662	-0.122678	-0.102492	0.078890	0.570784	0.009846	-0.22
8	-0.090227	-0.177865	-0.128561	0.341100	0.403712	-0.059442	0.560673	-0.004573	0.27
9	0.052510	0.041140	0.034488	0.064026	0.014549	0.020847	-0.223106	0.186675	0.29
10	0.043046	-0.058406	-0.069399	-0.008105	-0.273128	-0.081158	0.100693	0.143221	-0.35
11	0.024071	-0.145102	0.011143	0.038554	-0.089352	0.056177	-0.063536	-0.823444	0.35
12	0.595831	0.292642	-0.444638	0.001023	0.021884	-0.523622	0.125998	-0.141856	-0.06
13	0.080633	0.033467	-0.085697	-0.107828	0.151742	-0.056373	0.019286	-0.034012	-0.05
14	0.133406	-0.145498	0.029590	0.697723	-0.617275	0.009916	0.020952	0.038354	0.00
15	0.459139	-0.518569	-0.404318	-0.148739	0.051868	0.560363	-0.052731	0.101595	-0.02
16	0.358970	-0.543427	0.609651	-0.144986	0.080348	-0.414705	0.009018	0.050900	0.00

Q 2.7 Write down the explicit form of the first PC (in terms of the eigenvectors. Use values with two places of decimals only).

In [45]:

```
pd.DataFrame(pca.components_.round(2))
```

Out[45]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0.25	0.21	0.18	0.35	0.34	0.15	0.03	0.29	0.25	0.06	-0.04	0.32	0.32	-0.18	(
1	0.33	0.37	0.40	-0.08	-0.04	0.42	0.32	-0.25	-0.14	0.06	0.22	0.06	0.05	0.25	-
2	-0.06	-0.10	-0.08	0.04	-0.02	-0.06	0.14	0.05	0.15	0.68	0.50	-0.13	-0.07	-0.29	-
3	0.28	0.27	0.16	-0.05	-0.11	0.10	-0.16	0.13	0.18	0.09	-0.23	-0.53	-0.52	-0.16	(
4	0.01	0.06	-0.06	-0.40	-0.43	-0.04	0.30	0.22	0.56	-0.13	-0.22	0.14	0.20	-0.08	-
5	-0.02	0.01	-0.04	-0.05	0.03	-0.04	-0.19	-0.03	0.16	0.64	-0.33	0.09	0.15	0.49	-
6	-0.04	-0.01	-0.03	-0.16	-0.12	-0.03	0.06	0.11	0.21	-0.15	0.63	-0.00	-0.03	0.22	(
7	-0.10	-0.06	0.06	-0.12	-0.10	0.08	0.57	0.01	-0.22	0.21	-0.23	-0.08	-0.01	-0.08	(
8	-0.09	-0.18	-0.13	0.34	0.40	-0.06	0.56	-0.00	0.28	-0.13	-0.09	-0.19	-0.25	0.27	-
9	0.05	0.04	0.03	0.06	0.01	0.02	-0.22	0.19	0.30	-0.08	0.14	-0.12	-0.09	0.47	(
10	0.04	-0.06	-0.07	-0.01	-0.27	-0.08	0.10	0.14	-0.36	0.03	-0.02	0.04	-0.06	0.45	-
11	0.02	-0.15	0.01	0.04	-0.09	0.06	-0.06	-0.82	0.35	-0.03	-0.04	0.02	0.02	-0.01	(
12	0.60	0.29	-0.44	0.00	0.02	-0.52	0.13	-0.14	-0.07	0.01	0.04	0.13	-0.06	-0.02	(
13	0.08	0.03	-0.09	-0.11	0.15	-0.06	0.02	-0.03	-0.06	-0.07	0.03	-0.69	0.67	0.04	-
14	0.13	-0.15	0.03	0.70	-0.62	0.01	0.02	0.04	0.00	-0.01	-0.00	-0.11	0.16	-0.02	-
15	0.46	-0.52	-0.40	-0.15	0.05	0.56	-0.05	0.10	-0.03	0.00	-0.01	0.03	-0.03	-0.02	(
16	0.36	-0.54	0.61	-0.14	0.08	-0.41	0.01	0.05	0.00	0.00	-0.00	0.01	0.01	-0.00	-

In [ ]:

```

```

Q 2.8 Consider 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?

In [46]:

```
tot = sum(eig_vals)
var_exp = [( i /tot ) * 100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained",pd.DataFrame(cum_var_exp))
```

```
Cumulative Variance Explained                          0
0      36.165187+0.000000j
1      50.025194+0.000000j
2      58.607976+0.000000j
3      66.413765+0.000000j
4      73.684131+0.000000j
..                     ...
772  100.000000+0.000000j
773  100.000000+0.000000j
774  100.000000+0.000000j
775  100.000000+0.000000j
776  100.000000+0.000000j

[777 rows x 1 columns]
```

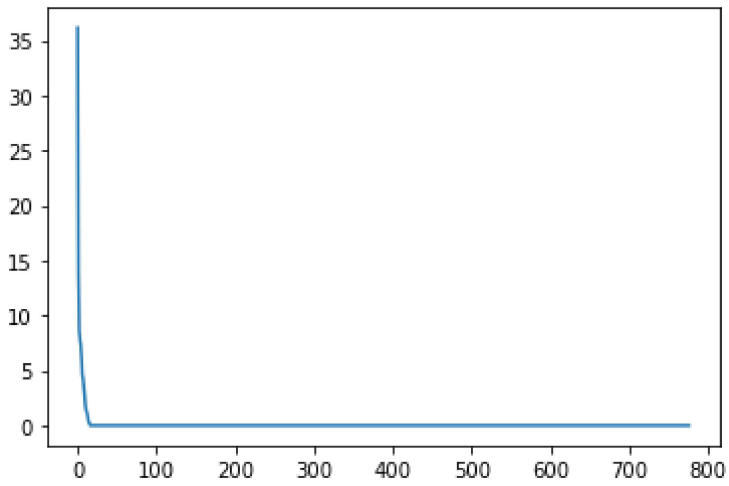
In [47]:

```
plt.plot(var_exp)
```

```
C:\Users\rahul\anaconda3\lib\site-packages\numpy\core\_asarray.py:83: Comple
xWarning: Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

Out[47]:

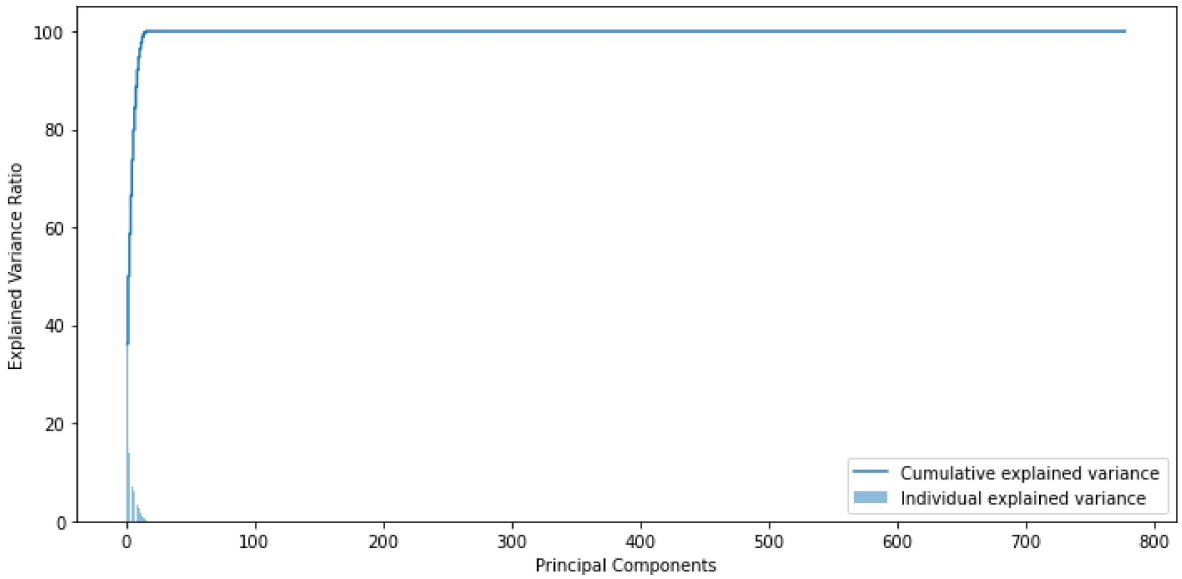
```
[<matplotlib.lines.Line2D at 0x12fdea57040>]
```



In [48]:

```
# Ploting
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 explain
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal Components')
plt.legend(loc = 'best')
plt.tight_layout()
plt.show()
```

```
C:\Users\rahul\anaconda3\lib\site-packages\numpy\core\_asarray.py:83: Comple
xWarning: Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



Q 2.9 Explain the business implication of using the Principal Component Analysis for this case study. How may PCs help in the further analysis? [Hint: Write Interpretations of the Principal Components Obtained]

ANS :-Mention in report .....thanks

In [ ]:

```
```