Name: Najmush Saquib Ali

Intern-OCT21

Exploratory Data Analysis(EDA)

Dataset Description:

1. **The dataset was released by Aspiring Minds from the Aspiring Mind Employment Outcome 2015 (AMEO).**
2. **The study is primarily limited only to students with engineering disciplines.**
3. **The dataset contains the employment outcomes of engineering graduates as dependent variables (Salary, Job Titles, and Job Locations) along with the standardized scores from three different areas – cognitive skills, technical skills and personality skills.**
4. **The dataset also contains demographic features. The dataset contains around 40 independent variables and 4000 data points.**
5. **The independent variables are both continuous and categorical in nature. The dataset contains a unique identifier for each candidate. Below mentioned table contains the details for the original dataset.**

Importing some neccessary libraries:

In [1]:

```
# It Will ignore some unneccessery warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

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

Importing and exploring the data:

In [3]:

```
df = pd.read_excel("data/aspiring_minds_employability_outcomes_2015.xlsx")
df.head()
```

Out[3]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10perce
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	
3	train	267447	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	

5 rows × 39 columns

In [6]:

```
df.shape #(number of rows , number of columns)
```

Out[6]:

```
(3998, 39)
```

In [7]:

```
df.columns #All columns name
```

Out[7]:

```
Index(['Unnamed: 0', 'ID', 'Salary', 'DOJ', 'DOL', 'Designation', 'JobCity',
       'Gender', 'DOB', '10percentage', '10board', '12graduation',
       '12percentage', '12board', 'CollegeID', 'CollegeTier', 'Degree',
       'Specialization', 'collegeGPA', 'CollegeCityID', 'CollegeCityTier',
       'CollegeState', 'GraduationYear', 'English', 'Logical', 'Quant',
       'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon',
       'ComputerScience', 'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg',
       'CivilEngg', 'conscientiousness', 'agreeableness', 'extraversion',
       'nueroticism', 'openess_to_experience'],
      dtype='object')
```

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3998 entries, 0 to 3997
Data columns (total 39 columns):
 #   Column                 Non-Null Count  Dtype         
---  ------                 --------------  -----         
 0   Unnamed: 0             3998 non-null   object        
 1   ID                     3998 non-null   int64         
 2   Salary                 3998 non-null   int64         
 3   DOJ                    3998 non-null   datetime64[ns]
 4   DOL                    3998 non-null   object        
 5   Designation            3998 non-null   object        
 6   JobCity                3998 non-null   object        
 7   Gender                 3998 non-null   object        
 8   DOB                    3998 non-null   datetime64[ns]
 9   10percentage           3998 non-null   float64       
 10  10board                3998 non-null   object        
 11  12graduation           3998 non-null   int64         
 12  12percentage           3998 non-null   float64       
 13  12board                3998 non-null   object        
 14  CollegeID              3998 non-null   int64         
 15  CollegeTier            3998 non-null   int64         
 16  Degree                 3998 non-null   object        
 17  Specialization         3998 non-null   object        
 18  collegeGPA             3998 non-null   float64       
 19  CollegeCityID          3998 non-null   int64         
 20  CollegeCityTier        3998 non-null   int64         
 21  CollegeState           3998 non-null   object        
 22  GraduationYear         3998 non-null   int64         
 23  English                3998 non-null   int64         
 24  Logical                3998 non-null   int64         
 25  Quant                  3998 non-null   int64         
 26  Domain                 3998 non-null   float64       
 27  ComputerProgramming    3998 non-null   int64         
 28  ElectronicsAndSemicon  3998 non-null   int64         
 29  ComputerScience        3998 non-null   int64         
 30  MechanicalEngg         3998 non-null   int64         
 31  ElectricalEngg         3998 non-null   int64         
 32  TelecomEngg            3998 non-null   int64         
 33  CivilEngg              3998 non-null   int64         
 34  conscientiousness      3998 non-null   float64       
 35  agreeableness          3998 non-null   float64       
 36  extraversion           3998 non-null   float64       
 37  nueroticism            3998 non-null   float64       
 38  openess_to_experience  3998 non-null   float64       
dtypes: datetime64[ns](2), float64(9), int64(18), object(10)
memory usage: 1.2+ MB
```

In [10]:

```
# Number of numeric columns
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
numeric_df = df.select_dtypes(include=numerics)
len(numeric_df.columns)
```

Out[10]:

27

In [11]:

```
# Quick statistical Analysis of the Dataset
df.describe()
```

Out[11]:

	ID	Salary	10percentage	12graduation	12percentage	CollegeID	C
count	3.998000e+03	3.998000e+03	3998.000000	3998.000000	3998.000000	3998.000000	3
mean	6.637945e+05	3.076998e+05	77.925443	2008.087544	74.466366	5156.851426	
std	3.632182e+05	2.127375e+05	9.850162	1.653599	10.999933	4802.261482	
min	1.124400e+04	3.500000e+04	43.000000	1995.000000	40.000000	2.000000	
25%	3.342842e+05	1.800000e+05	71.680000	2007.000000	66.000000	494.000000	
50%	6.396000e+05	3.000000e+05	79.150000	2008.000000	74.400000	3879.000000	
75%	9.904800e+05	3.700000e+05	85.670000	2009.000000	82.600000	8818.000000	
max	1.298275e+06	4.000000e+06	97.760000	2013.000000	98.700000	18409.000000	

8 rows × 27 columns

In [12]:

```
# Checking for null values
df.isnull().sum()
```

Out[12]:

```
Unnamed: 0              0
ID                      0
Salary                  0
DOJ                     0
DOL                     0
Designation             0
JobCity                 0
Gender                  0
DOB                     0
10percentage            0
10board                 0
12graduation            0
12percentage            0
12board                 0
CollegeID               0
CollegeTier             0
Degree                  0
Specialization          0
collegeGPA              0
CollegeCityID           0
CollegeCityTier         0
CollegeState            0
GraduationYear          0
English                 0
Logical                 0
Quant                   0
Domain                  0
ComputerProgramming     0
ElectronicsAndSemicon   0
ComputerScience         0
MechanicalEngg          0
ElectricalEngg          0
TelecomEngg             0
CivilEngg               0
conscientiousness       0
agreeableness           0
extraversion            0
nueroticism             0
openess_to_experience   0
dtype: int64
```

There are no null values in the dataset

In [4]:

```
# We can drop some unneccessary columns to keep our dataset clean
df.drop(['Unnamed: 0','ID','10board','12board','CollegeID','CollegeCityID'],axis=1,inplace=
df.head()
```

Out[4]:

	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	12graduation
0	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	2007
1	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	2007
2	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	2010
3	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	2007
4	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	2008

5 rows × 33 columns

In [14]:

```
df.shape
```

Out[14]:

```
(3998, 33)
```

Exploratory Data Analysis and Visualization:

In [16]:

```
df.columns
```

Out[16]:

```
Index(['Salary', 'DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB',
       '10percentage', '12graduation', '12percentage', 'CollegeTier', 'Degre
e',
       'Specialization', 'collegeGPA', 'CollegeCityTier', 'CollegeState',
       'GraduationYear', 'English', 'Logical', 'Quant', 'Domain',
       'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience',
       'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg',
       'conscientiousness', 'agreeableness', 'extraversion', 'nueroticism',
       'openess_to_experience'],
      dtype='object')
```

Columns I'll Analyze

1. Salary
2. 10percentage
3. 12percentage
4. collegeGPA
5. Gender
6. Designation
7. JobCity
8. Degree
9. Specialization
10. CollegeState
11. GraduationYear
12. English
13. Logical
14. Quant
15. ComputerProgramming
16. ElectronicsAndSemicon
17. ComputerScience
18. MechanicalEngg
19. ElectricalEngg
20. TelecomEngg
21. CivilEngg

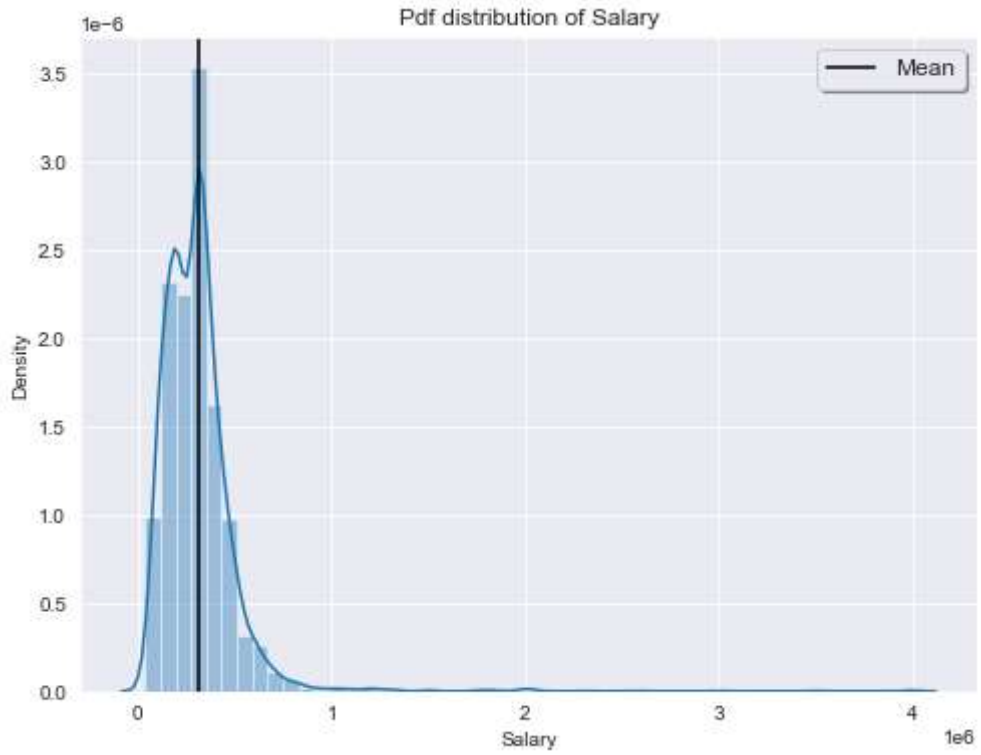
Univariate Analysis:

PDF distribution of Salary:

In [35]:

```
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
sns.distplot(df['Salary'])
plt.axvline(df['Salary'].mean(),color='black', label='Mean')
plt.title("Pdf distribution of Salary")
plt.legend(shadow=True,fontsize="larger")

skew = df['Salary'].skew()
kurt = df['Salary'].kurt()
print('Skewness:{}'.format(round(skew,2)))
print('Kurtosis:{}'.format(round(kurt,2)))
```

```
Skewness:6.45
Kurtosis:80.93
```



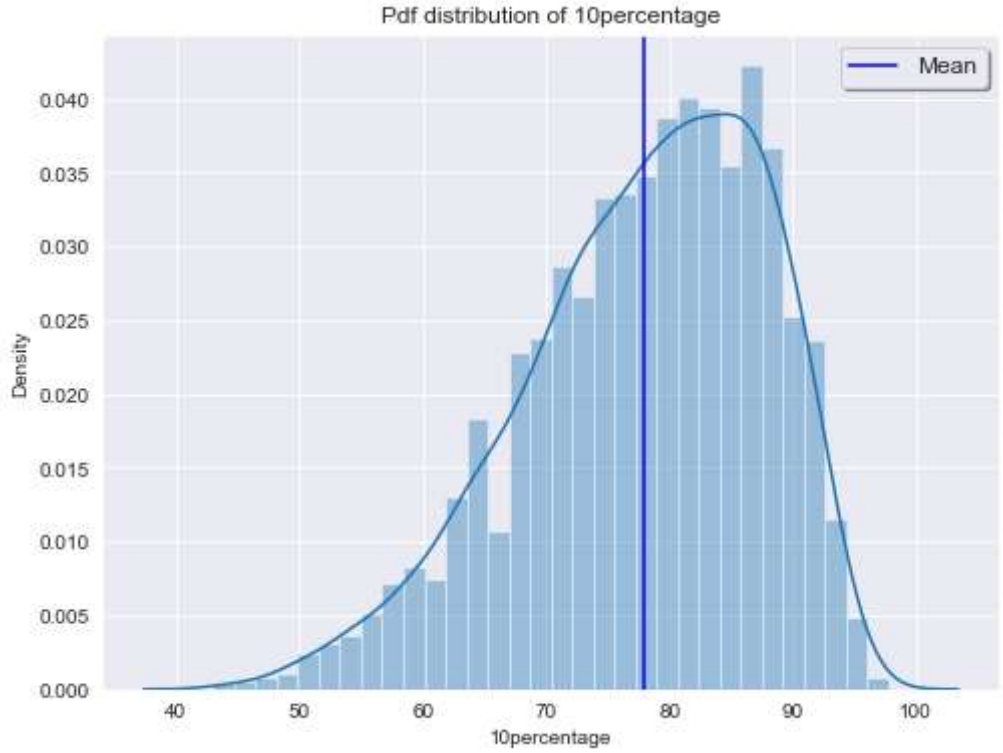
From the above plot as we can see that it's having a long tail at the right side , looks like log normal distribution

PDF distribution of 10percentage:

In [36]:

```
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
sns.distplot(df['10percentage'])
plt.axvline(df['10percentage'].mean(), color="blue", label="Mean")
plt.title("Pdf distribution of 10percentage")
plt.legend(shadow=True,fontsize="larger")

skew = df['10percentage'].skew()
kurt = df['10percentage'].kurt()
print('Skewness:{}'.format(round(skew,2)))
print('Kurtosis:{}'.format(round(kurt,2)))
```

```
Skewness:-0.59
Kurtosis:-0.11
```



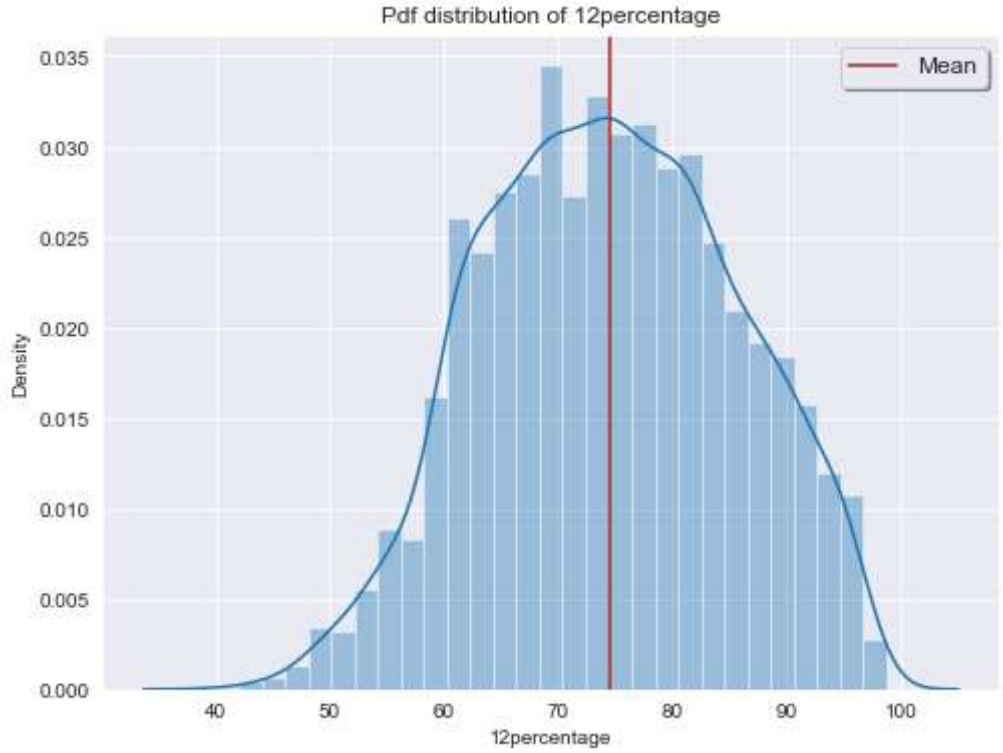
From the above plot as we can see that it's having a thick tail towards the left side looks like a negatively skewed distribution

PDF distribution of 12percentage:

In [38]:

```
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
sns.distplot(df['12percentage'])
plt.axvline(df['12percentage'].mean(), color="brown", label="Mean")
plt.title("Pdf distribution of 12percentage")
plt.legend(shadow=True,fontsize="larger")

skew = df['12percentage'].skew()
kurt = df['12percentage'].kurt()
print('Skewness:{}'.format(round(skew,2)))
print('Kurtosis:{}'.format(round(kurt,2)))
```

```
Skewness:-0.03
Kurtosis:-0.63
```



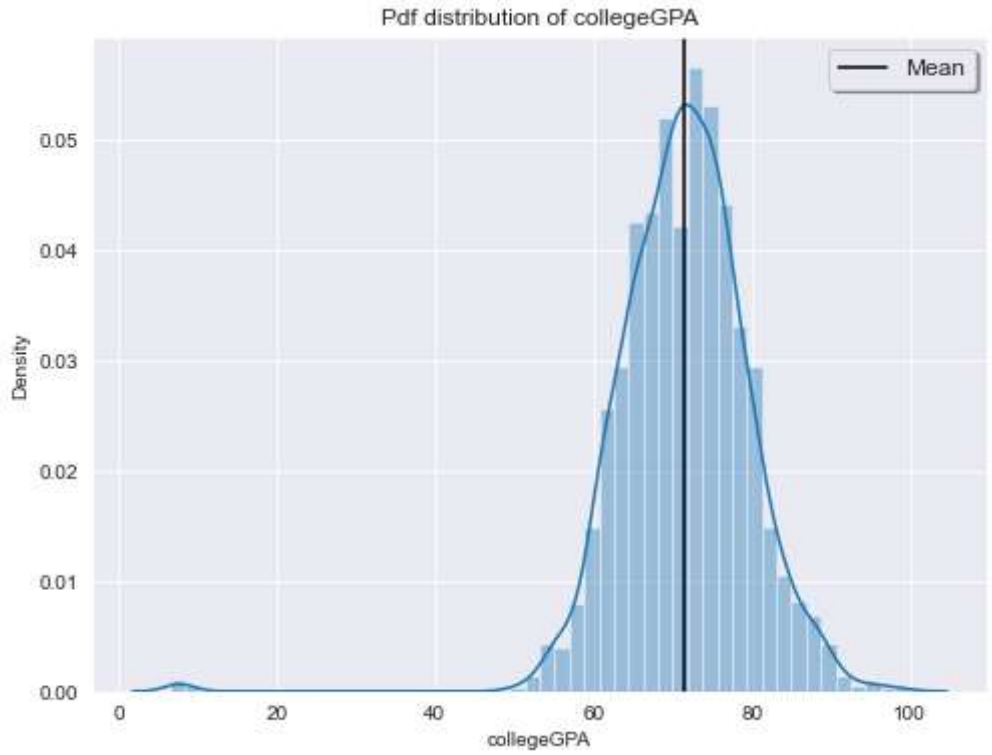
From the above plot as we can see that it's having a tail towards left side looks like a negatively skewed distribution

PDF distribution of collegeGPA:

In [40]:

```
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
sns.distplot(df['collegeGPA'])
plt.axvline(df['collegeGPA'].mean(), color="black", label="Mean")
plt.title("Pdf distribution of collegeGPA")
plt.legend(shadow=True,fontsize="larger")

skew = df['collegeGPA'].skew()
kurt = df['collegeGPA'].kurt()
print('Skewness:{}'.format(round(skew,2)))
print('Kurtosis:{}'.format(round(kurt,2)))
```

```
Skewness:-1.25
Kurtosis:10.23
```

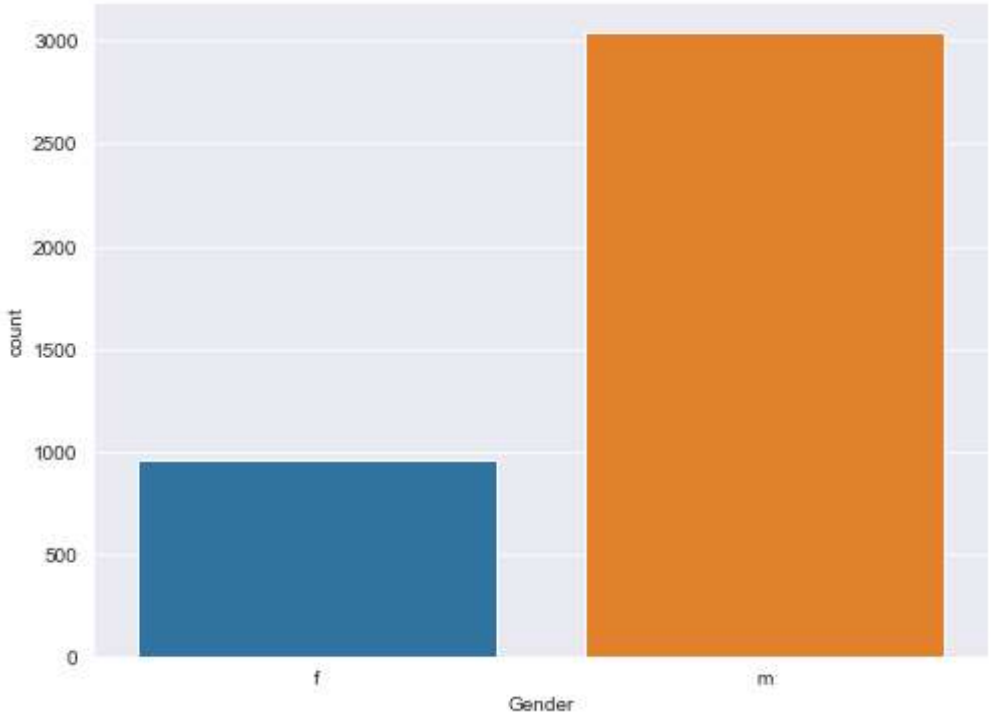


From the above plot we can see that it's having a long tail towards the left looks like highly negativly skwed

Plotting Gender Wise Distribution:

In [49]:

```
plt.figure(figsize=(8,6))
sns.countplot(df['Gender'])
plt.show()
```



From the above plot we can say that there are more male candidates than female

Designation offered in the job and it's count:

In [5]:

```
df['Designation']
```

Out[5]:

```
0           senior quality engineer
1                 assistant manager
2                  systems engineer
3         senior software engineer
4                               get
                  ...
3993              software engineer
3994               technical writer
3995    associate software engineer
3996             software developer
3997        senior systems engineer
Name: Designation, Length: 3998, dtype: object
```

In [6]:

```
designations = df['Designation'].unique()
len(designations)
```

Out[6]:

```
419
```

There are 419 unique values

In [8]:

```
designation_count = df['Designation'].value_counts()
designation_count
```

Out[8]:

```
software engineer           539
software developer          265
system engineer             205
programmer analyst          139
systems engineer            118
                           ...
desktop support engineer      1
full stack developer          1
business systems analyst      1
associate developer           1
operations                    1
Name: Designation, Length: 419, dtype: int64
```

In [11]:

```
# Taking the most frequent designation
designation_count[:15]
```

Out[11]:

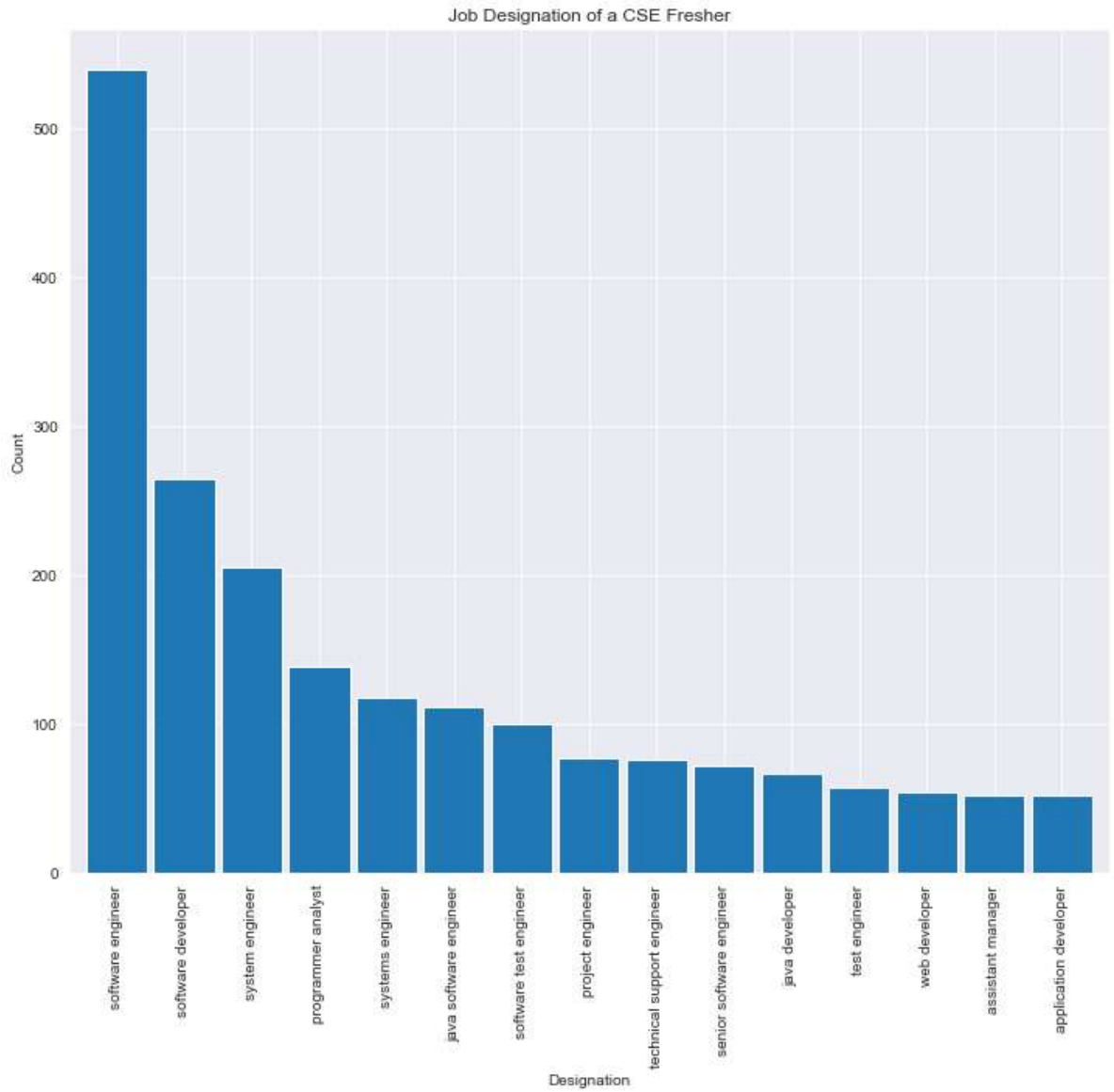
```
software engineer             539
software developer            265
system engineer               205
programmer analyst            139
systems engineer              118
java software engineer        111
software test engineer        100
project engineer               77
technical support engineer     76
senior software engineer       72
java developer                 67
test engineer                  57
web developer                  54
assistant manager              52
application developer          52
Name: Designation, dtype: int64
```

In [42]:

```
#Plotting the most frequent designation
sns.set_style("darkgrid")
plt.figure(figsize=(12,10))
designation_count[:15].plot(kind='bar' , width=0.9)
plt.xlabel('Designation')
plt.ylabel('Count')
plt.title('Job Designation of a CSE Fresher')
```

Out[42]:

```
Text(0.5, 1.0, 'Job Designation of a CSE Fresher')
```



From the above plot we can analyse that most of the candidate choose software engineering

Location of the job (city) and it's count:

In [28]:

```
df['JobCity']
```

Out[28]:

```
0              Bangalore
1                 Indore
2                Chennai
3                Gurgaon
4                Manesar
              ...
3993           New Delhi
3994           Hyderabad
3995           Bangalore
3996    Asifabadbanglore
3997             Chennai
Name: JobCity, Length: 3998, dtype: object
```

In [30]:

```
jobcities = df['JobCity'].unique()
len(jobcities)
```

Out[30]:

```
339
```

There are 339 unique values

In [32]:

```
jobcities_count = df['JobCity'].value_counts()
jobcities_count
```

Out[32]:

```
Bangalore     627
-1            461
Noida         368
Hyderabad     335
Pune          290
             ...
New delhi       1
Rajasthan       1
Ernakulam       1
Dausa           1
bihar           1
Name: JobCity, Length: 339, dtype: int64
```

As you can see above there is an wrong data "-1", we simply gonna replace it with na value and drop it

In [5]:

```
df['JobCity'].replace(-1,np.nan,inplace=True)
df['JobCity'].dropna(inplace=True)
```

In [38]:

```
jobcities_count = df['JobCity'].value_counts()
jobcities_count
```

Out[38]:

```
Bangalore    627
Noida        368
Hyderabad    335
Pune         290
Chennai      272
            ...
New delhi      1
Rajasthan      1
Ernakulam      1
Dausa          1
bihar          1
Name: JobCity, Length: 338, dtype: int64
```

In [39]:

```
# Taking the most frequent cities that a candidate choose to work
jobcities_count[:10]
```

Out[39]:

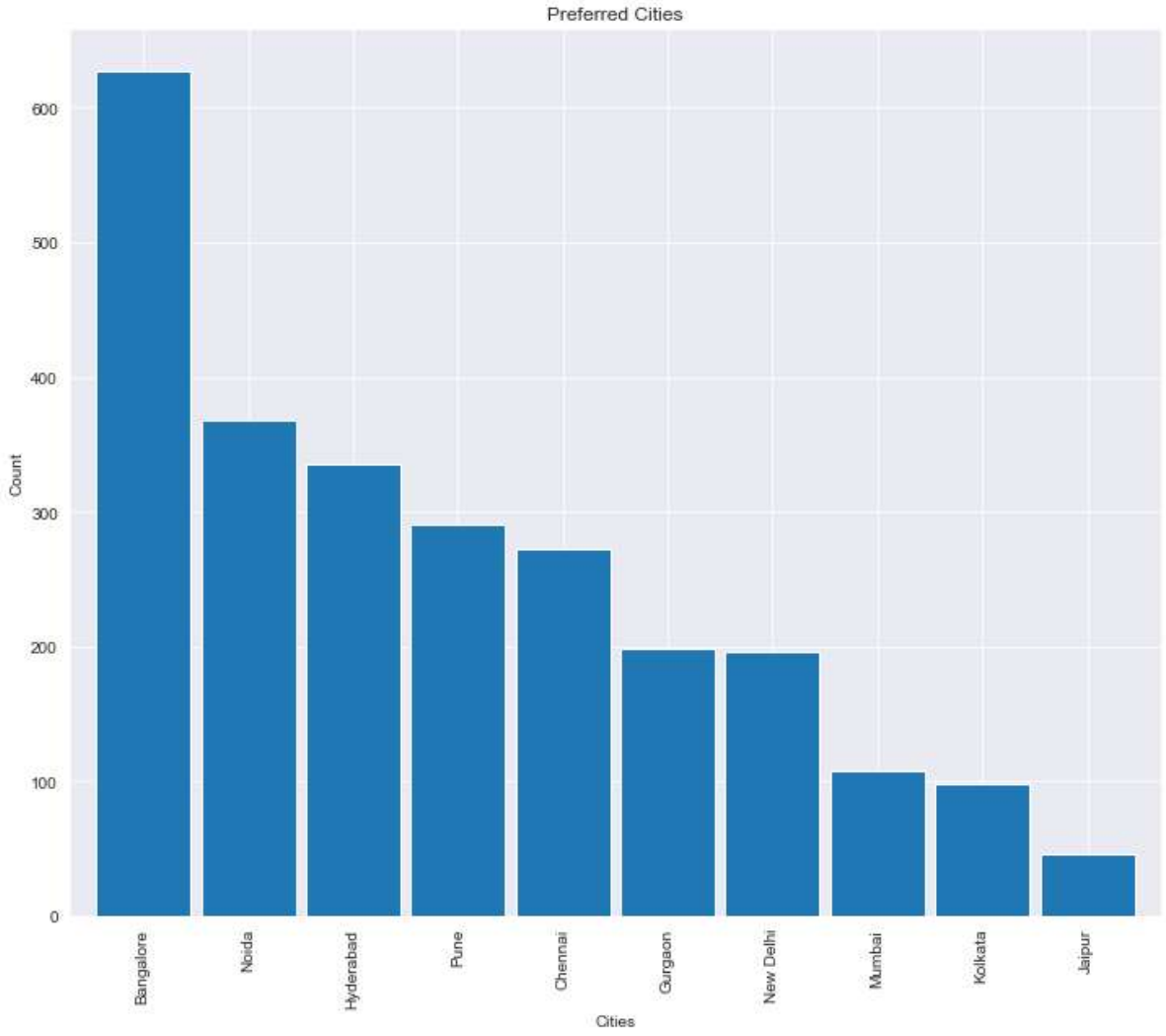
```
Bangalore    627
Noida        368
Hyderabad    335
Pune         290
Chennai      272
Gurgaon      198
New Delhi    196
Mumbai       108
Kolkata       98
Jaipur        46
Name: JobCity, dtype: int64
```

In [44]:

```
#Plotting the most frequent cities
sns.set_style("darkgrid")
plt.figure(figsize=(12,10))
jobcities_count[:10].plot(kind='bar' , width=0.9)
plt.xlabel('Cities')
plt.ylabel('Count')
plt.title('Preferred Cities')
```

Out[44]:

```
Text(0.5, 1.0, 'Preferred Cities')
```



From the above plot we can say that most of the candidates choose Bangalore As their preferred city

Degree obtained/pursued by the candidate and it's count:

In [45]:

```
df['Degree']
```

Out[45]:

```
0       B.Tech/B.E.
1       B.Tech/B.E.
2       B.Tech/B.E.
3       B.Tech/B.E.
4       B.Tech/B.E.
           ...
3993    B.Tech/B.E.
3994    B.Tech/B.E.
3995    B.Tech/B.E.
3996    B.Tech/B.E.
3997    B.Tech/B.E.
Name: Degree, Length: 3998, dtype: object
```

In [47]:

```
uniq_degree = df['Degree'].unique()
len(uniq_degree)
```

Out[47]:

```
4
```

There are 4 unique values

In [48]:

```
degree_count = df['Degree'].value_counts()
degree_count
```

Out[48]:

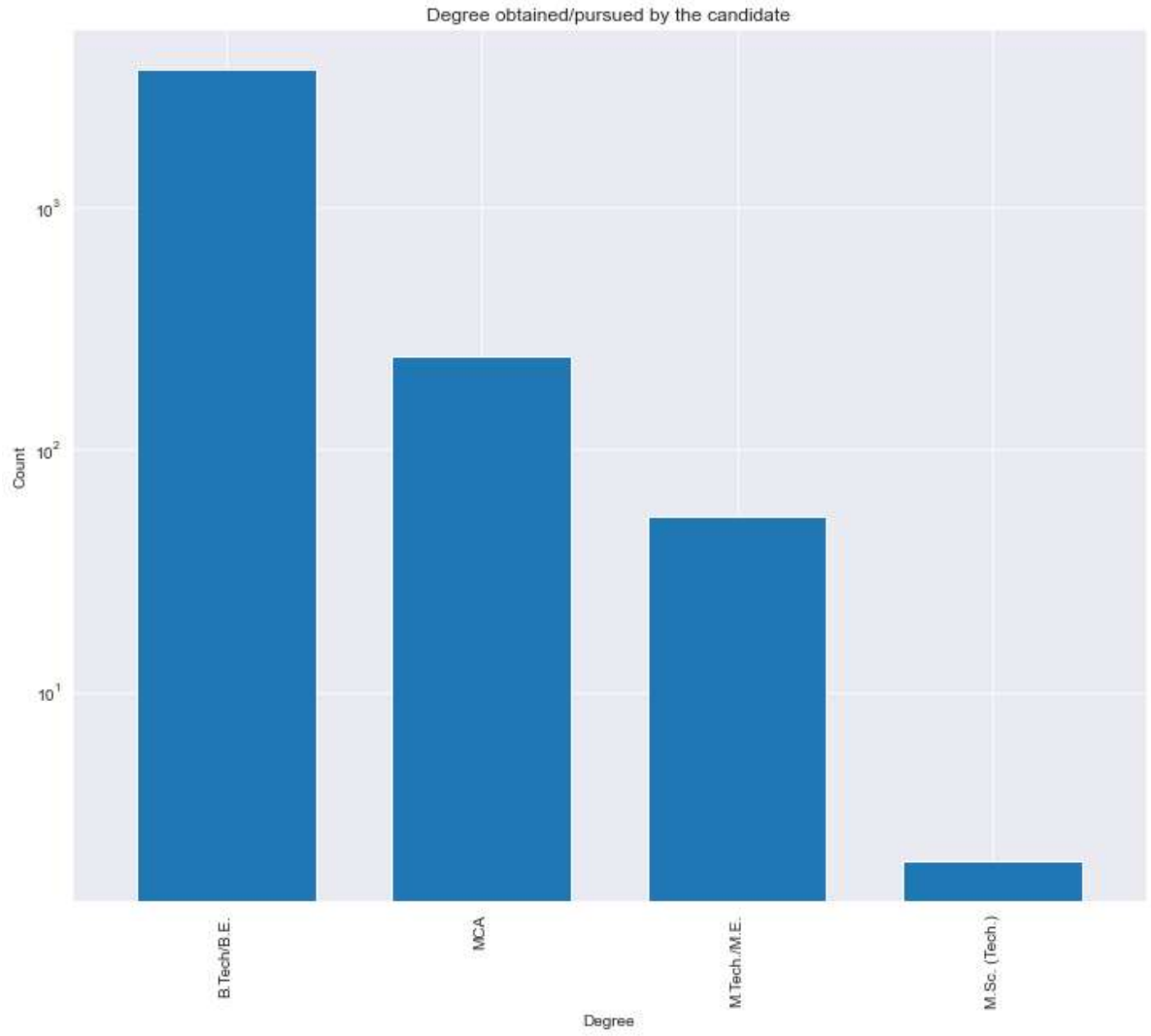
```
B.Tech/B.E.      3700
MCA               243
M.Tech./M.E.       53
M.Sc. (Tech.)       2
Name: Degree, dtype: int64
```

In [52]:

```
# Plotting the most frequent Degree
sns.set_style("darkgrid")
plt.figure(figsize=(12,10))
degree_count.plot(kind='bar' , width=0.7 , logy=True)
plt.xlabel('Degree')
plt.ylabel('Count')
plt.title('Degree obtained/pursued by the candidate')
```

Out[52]:

```
Text(0.5, 1.0, 'Degree obtained/pursued by the candidate')
```



From the above plot we can say that most of the candidates are from B.Tech/B.E

Specialization pursued by the candidate and it's count:

In [54]:

```
df['Specialization']
```

Out[54]:

```
0                            computer engineering
1       electronics and communication engineering
2                          information technology
3                            computer engineering
4       electronics and communication engineering
                          ...
3993                       information technology
3994    electronics and communication engineering
3995                         computer engineering
3996                computer science & engineering
3997                       information technology
Name: Specialization, Length: 3998, dtype: object
```

In [55]:

```
uniq_specialization = df['Specialization'].unique()
len(uniq_specialization)
```

Out[55]:

```
46
```

There are 46 unique values

In [58]:

```
specialization_count = df['Specialization'].value_counts()
specialization_count
```

Out[58]:

```
electronics and communication engineering      880
computer science & engineering                 744
information technology                         660
computer engineering                           600
computer application                           244
mechanical engineering                         201
electronics and electrical engineering         196
electronics & telecommunications               121
electrical engineering                          82
electronics & instrumentation eng               32
civil engineering                               29
information science engineering                 27
electronics and instrumentation engineering     27
instrumentation and control engineering         20
electronics engineering                         19
biotechnology                                   15
other                                           13
industrial & production engineering             10
chemical engineering                             9
applied electronics and instrumentation          9
computer science and technology                  6
telecommunication engineering                    6
mechanical and automation                        5
automobile/automotive engineering                5
mechatronics                                     4
instrumentation engineering                      4
electronics and computer engineering             3
aeronautical engineering                         3
computer science                                 2
metallurgical engineering                        2
information & communication technology           2
electrical and power engineering                 2
biomedical engineering                           2
industrial engineering                           2
internal combustion engine                       1
mechanical & production engineering              1
electronics                                      1
polymer technology                               1
information science                              1
embedded systems technology                      1
control and instrumentation engineering          1
computer networking                              1
ceramic engineering                              1
computer and communication engineering           1
industrial & management engineering              1
power systems and automation                     1
Name: Specialization, dtype: int64
```

In [59]:

```
specialization_count[:10]
```

Out[59]:

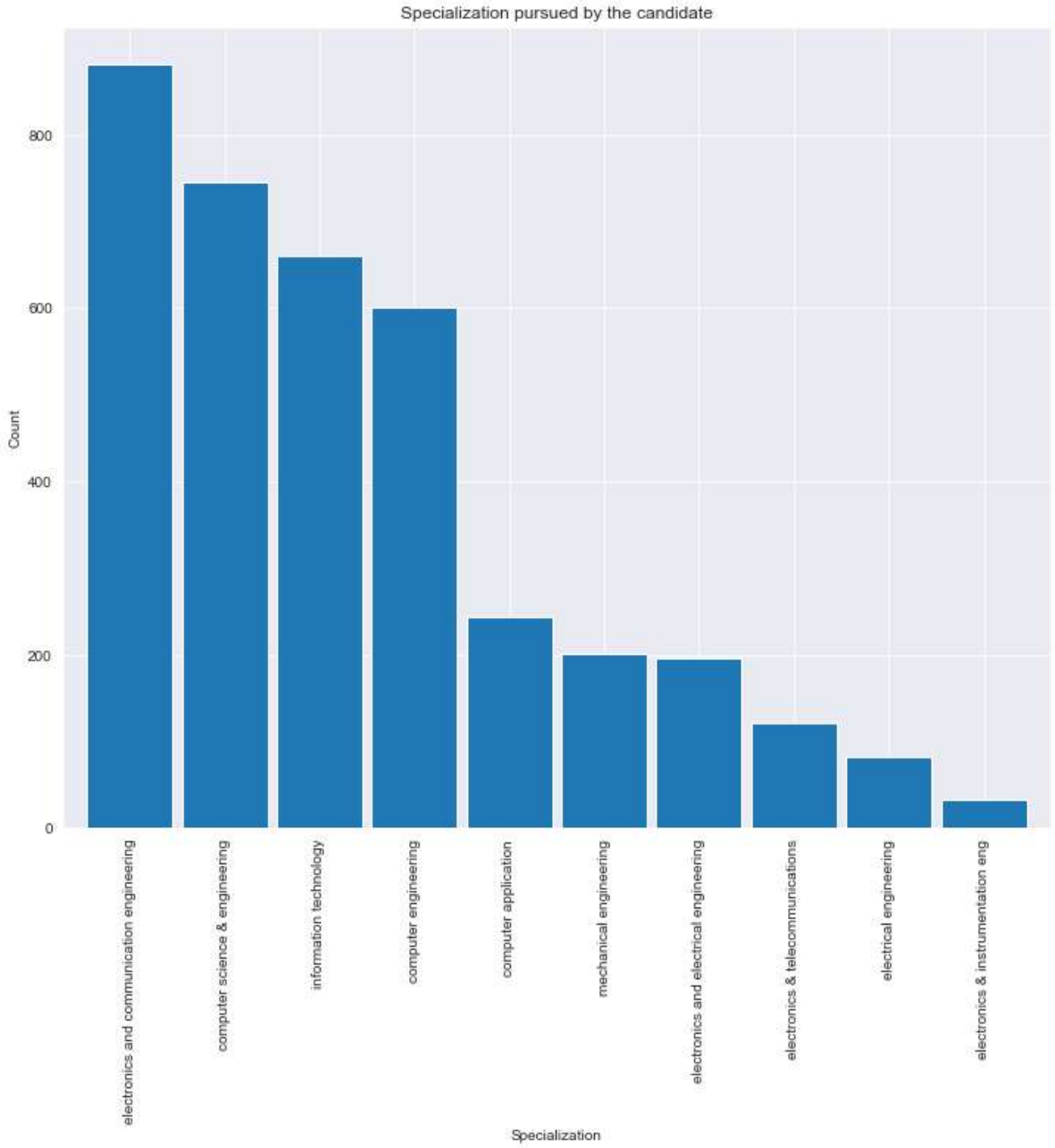
```
electronics and communication engineering    880
computer science & engineering               744
information technology                       660
computer engineering                         600
computer application                         244
mechanical engineering                       201
electronics and electrical engineering       196
electronics & telecommunications             121
electrical engineering                        82
electronics & instrumentation eng             32
Name: Specialization, dtype: int64
```

In [60]:

```
#Plotting the most frequent specialization
sns.set_style("darkgrid")
plt.figure(figsize=(12,10))
specialization_count[:10].plot(kind='bar' , width=0.9)
plt.xlabel('Specialization')
plt.ylabel('Count')
plt.title('Specialization pursued by the candidate')
```

Out[60]:

```
Text(0.5, 1.0, 'Specialization pursued by the candidate')
```



From the above plot we can see that most choosed specialization is Electronics and communication engineering

College State and it's count:

In [6]:

```
df['CollegeState']
```

Out[6]:

```
0       Andhra Pradesh
1       Madhya Pradesh
2        Uttar Pradesh
3                Delhi
4        Uttar Pradesh
             ...
3993           Haryana
3994         Telangana
3995            Orissa
3996         Karnataka
3997        Tamil Nadu
Name: CollegeState, Length: 3998, dtype: object
```

In [7]:

```
uniq_collegestate = df['CollegeState'].unique()
len(uniq_collegestate)
```

Out[7]:

```
26
```

In [8]:

```
college_state_count = df['CollegeState'].value_counts()
college_state_count
```

Out[8]:

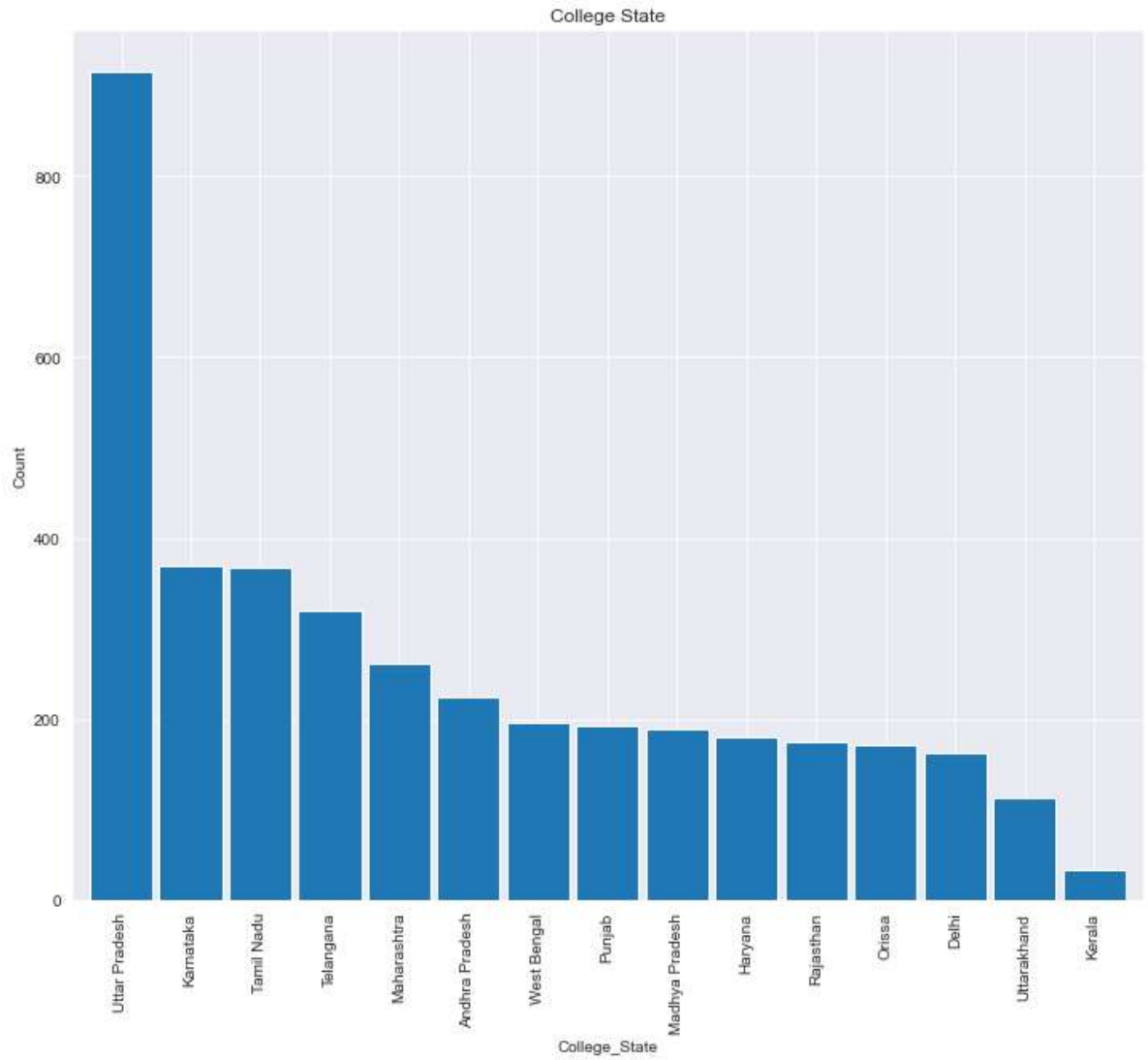
```
Uttar Pradesh        915
Karnataka            370
Tamil Nadu           367
Telangana            319
Maharashtra          262
Andhra Pradesh       225
West Bengal          196
Punjab               193
Madhya Pradesh       189
Haryana              180
Rajasthan            174
Orissa               172
Delhi                162
Uttarakhand          113
Kerala                33
Jharkhand             28
Chhattisgarh          27
Gujarat               24
Himachal Pradesh      16
Bihar                 10
Jammu and Kashmir      7
Union Territory        5
Assam                  5
Sikkim                 3
Meghalaya              2
Goa                    1
Name: CollegeState, dtype: int64
```

In [19]:

```
sns.set_style("darkgrid")
plt.figure(figsize=(12,10))
college_state_count[:15].plot(kind='bar' , width=0.9)
plt.xlabel('College_State')
plt.ylabel('Count')
plt.title('College State Distribution')
```

Out[19]:

```
Text(0.5, 1.0, 'College State')
```

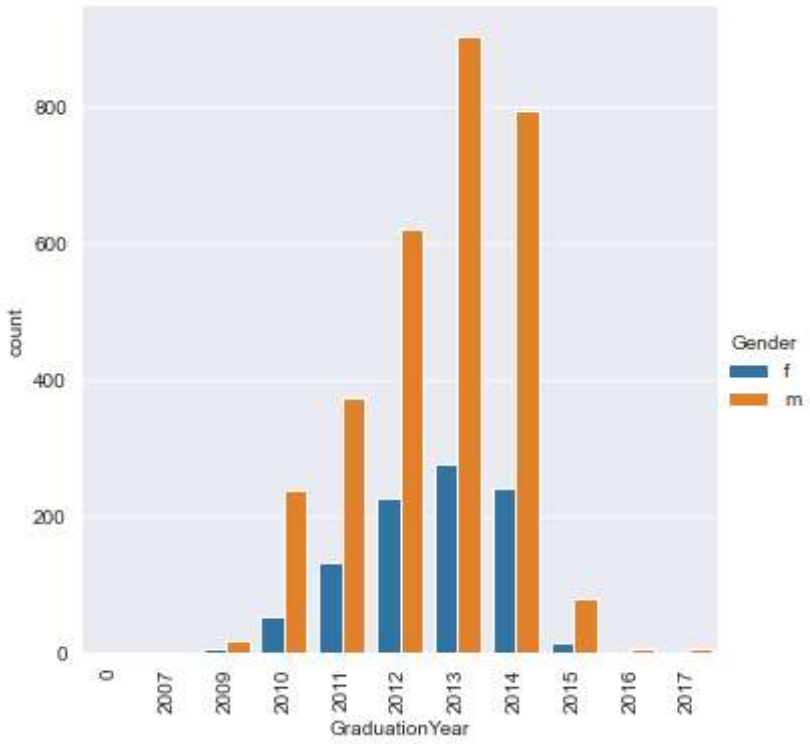


From the above plot we can see that Uttar Pradesh is having most of the preferrence colleges

Yearwise Employment groupby Gender:

In [70]:

```
sns.catplot(x = "GraduationYear",hue="Gender",data = df,kind='count')
plt.xticks(rotation=90)
plt.show()
```



From the above plot we can say that in 2013 joining of employee is higher , in 2017 joining of employee is lower and in all the years joining of male emplyoee is higher then female employees

Detect the outliers in each numerical column:

In [21]:

```
df.columns
```

Out[21]:

```
Index(['Salary', 'DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB',
       '10percentage', '12graduation', '12percentage', 'CollegeTier', 'Degre
e',
       'Specialization', 'collegeGPA', 'CollegeCityTier', 'CollegeState',
       'GraduationYear', 'English', 'Logical', 'Quant', 'Domain',
       'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience',
       'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg',
       'conscientiousness', 'agreeableness', 'extraversion', 'nueroticism',
       'openess_to_experience'],
      dtype='object')
```

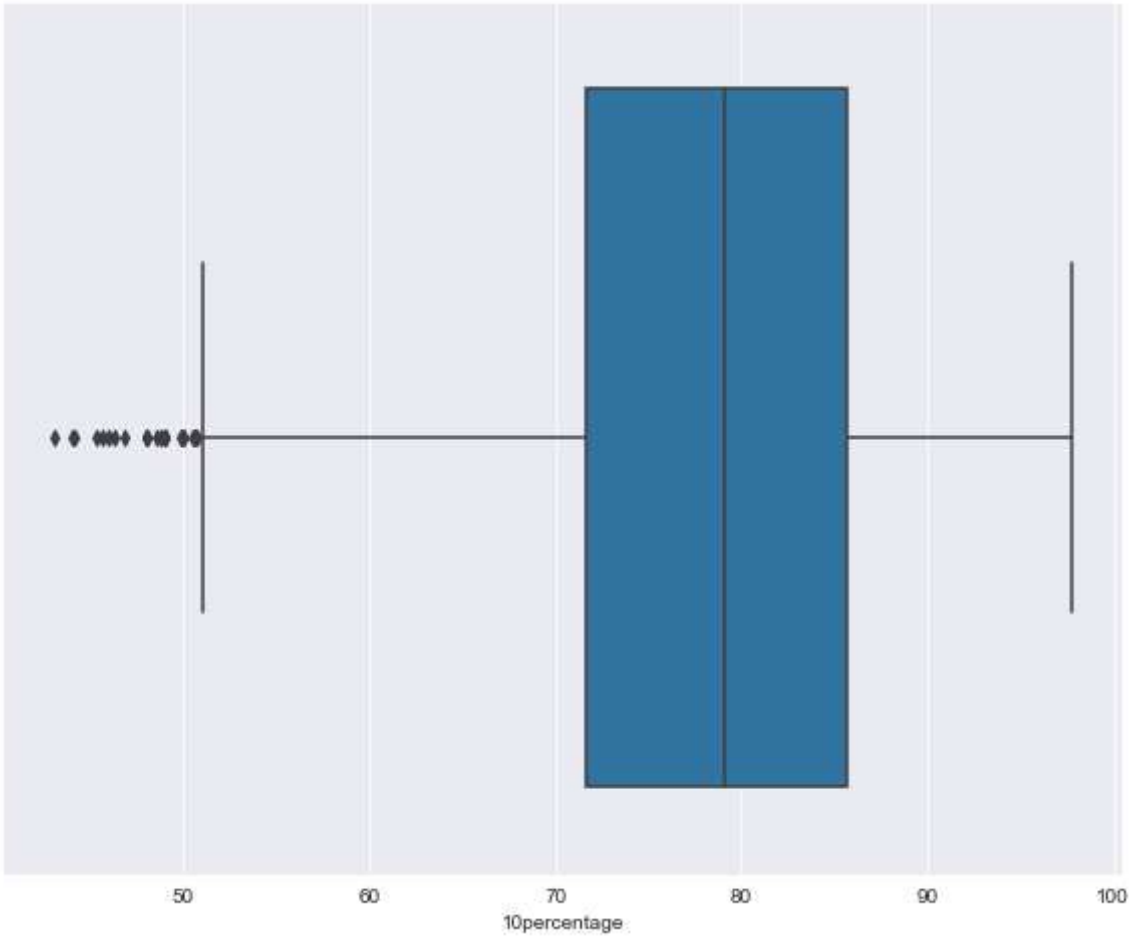
Finding the outliers in 10percentage column:

In [23]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['10percentage'])
```

Out[23]:

```
<AxesSubplot:xlabel='10percentage'>
```



From the above plot we can say that there are some outliers

In [24]:

```
# Outliers
df['10percentage'][df['10percentage']<52].reset_index()
```

Out[24]:

	index	10percentage
0	108	51.00
1	245	50.60
2	466	44.16
3	490	44.00
4	491	45.60
5	502	48.00
6	600	49.00
7	613	48.00
8	887	51.20
9	898	49.00
10	919	48.80
11	1064	49.00
12	1102	49.00
13	1169	48.50
14	1193	48.00
15	1235	50.60
16	1334	43.00
17	1838	50.00
18	1845	49.00
19	1955	45.33
20	1976	46.24
21	2024	51.36
22	2037	48.00
23	2123	51.00
24	2215	50.50
25	2217	51.00
26	2292	50.00
27	2432	50.00
28	2563	51.60
29	2655	50.66
30	2885	46.80
31	2982	50.00
32	3284	50.00
33	3425	50.00

	index	10percentage
34	3507	51.83
35	3525	51.42
36	3690	46.00
37	3743	49.90

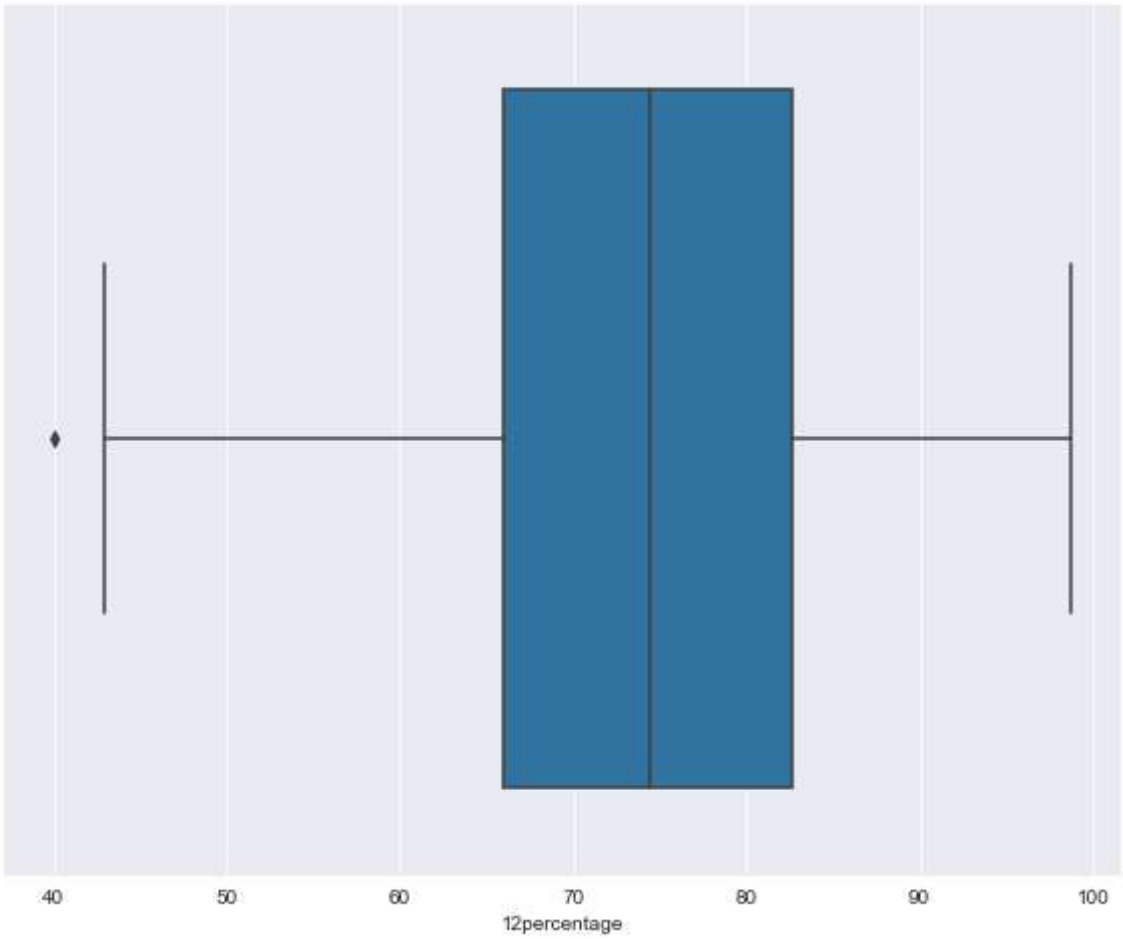
Finding the outliers in 12percentage column:

In [25]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['12percentage'])
```

Out[25]:

```
<AxesSubplot:xlabel='12percentage'>
```



From the above plot we can say that there are some outliers

In [28]:

```
# outliers
df['10percentage'][df['10percentage']<44].reset_index()
```

Out[28]:

	index	10percentage
0	1334	43.0

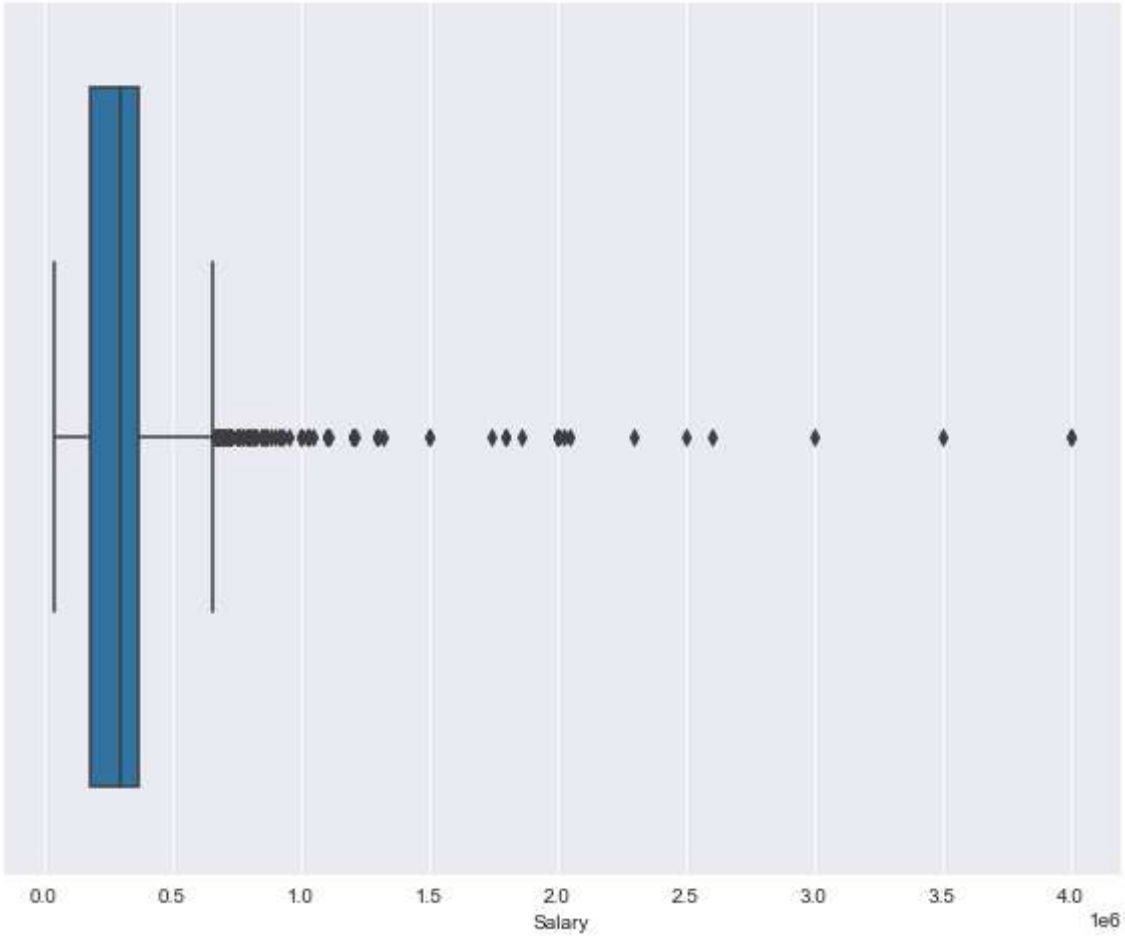
Finding the outliers in Salary column:

In [29]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['Salary'])
```

Out[29]:

```
<AxesSubplot:xlabel='Salary'>
```



From the above plot we can say that there are some outliers

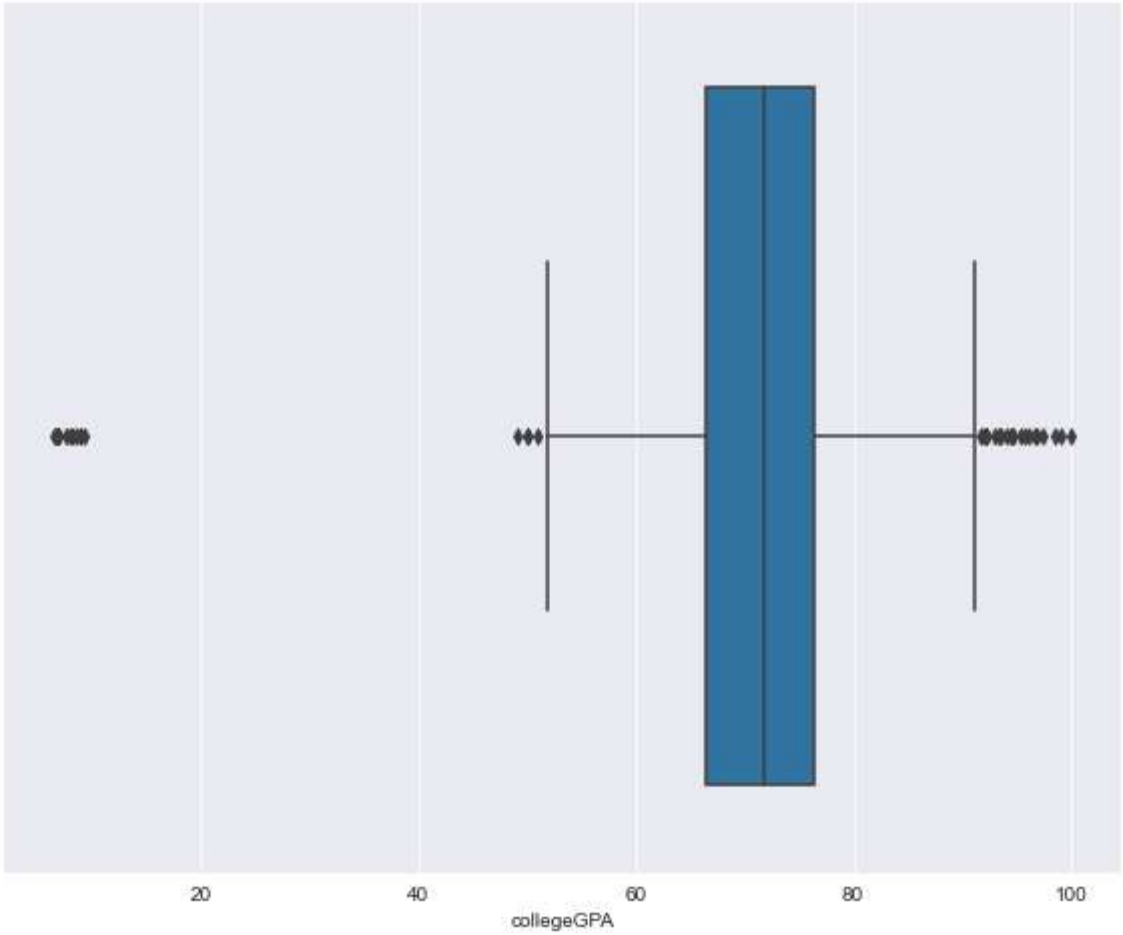
Finding the outliers in collegeGPA column:

In [30]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['collegeGPA'])
```

Out[30]:

```
<AxesSubplot:xlabel='collegeGPA'>
```



From the above plot we can say that there are some outliers

In [33]:

```
# Outliers
df['collegeGPA'][(df['collegeGPA']<53) | (df['collegeGPA']>93)].reset_index()
```

Out[33]:

	index	collegeGPA
0	7	8.58
1	138	6.63
2	324	52.00
3	614	93.60
4	690	99.93
5	788	6.80
6	874	94.50
7	907	50.00
8	968	52.00
9	1134	96.00
10	1264	97.30
11	1345	93.30
12	1400	51.80
13	1419	6.85
14	1439	8.07
15	1510	96.70
16	1685	94.70
17	1767	7.56
18	2125	52.00
19	2151	6.95
20	2152	95.30
21	2229	8.13
22	2293	9.30
23	2410	52.00
24	2662	8.88
25	2691	8.89
26	2703	94.00
27	2836	49.07
28	2988	94.60
29	3151	98.40
30	3276	95.70
31	3293	51.00
32	3308	6.45
33	3323	96.90

	index	collegeGPA
34	3448	50.00
35	3590	52.00
36	3850	99.00

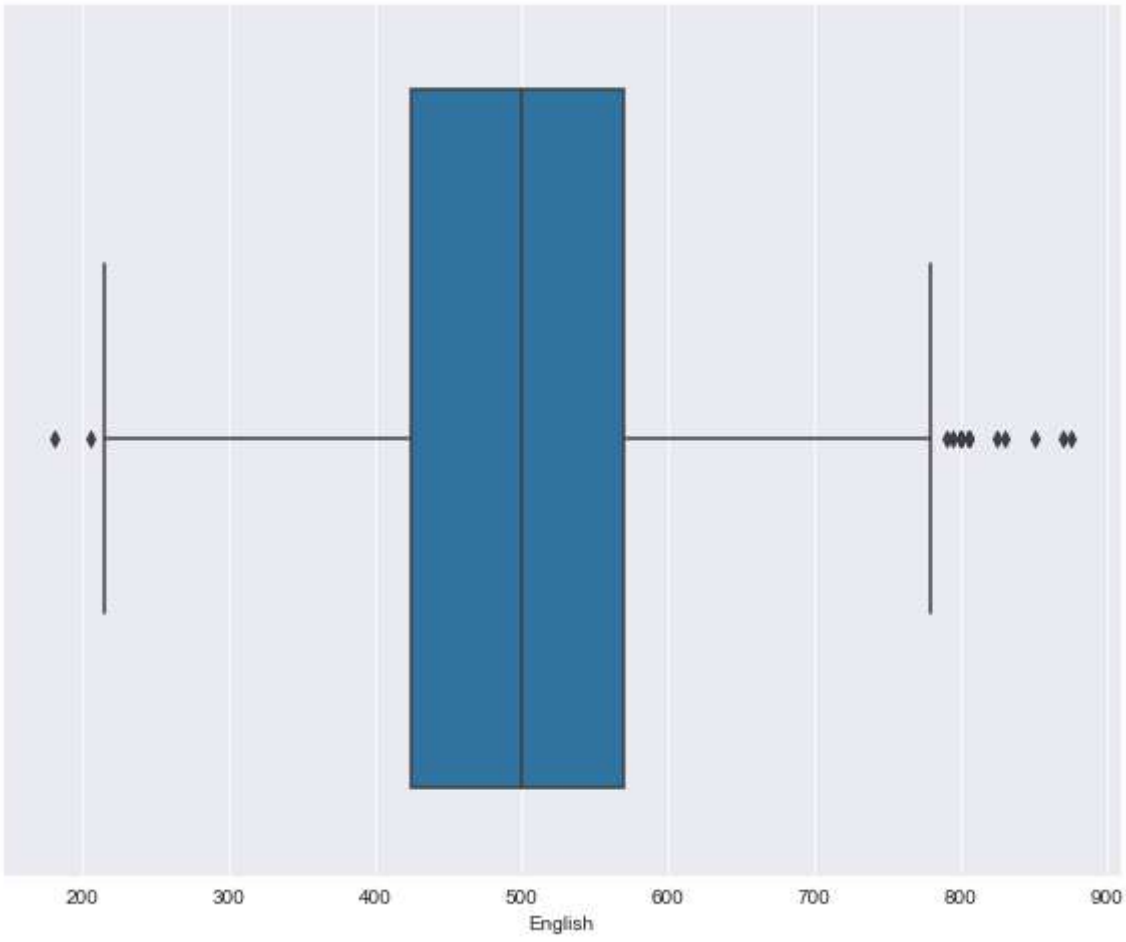
Finding the outliers in English column:

In [34]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['English'])
```

Out[34]:

<AxesSubplot:xlabel='English'>



From the above plot we can say that there are some outliers

In [35]:

```
# Outliers
df['English'][(df['English']<220) | (df['English']>790)].reset_index()
```

Out[35]:

	index	English
0	275	875
1	444	825
2	624	215
3	668	870
4	847	800
5	935	205
6	1183	805
7	1217	180
8	1450	830
9	1519	795
10	2077	800
11	2122	800
12	2273	215
13	2385	805
14	2596	805
15	3044	850

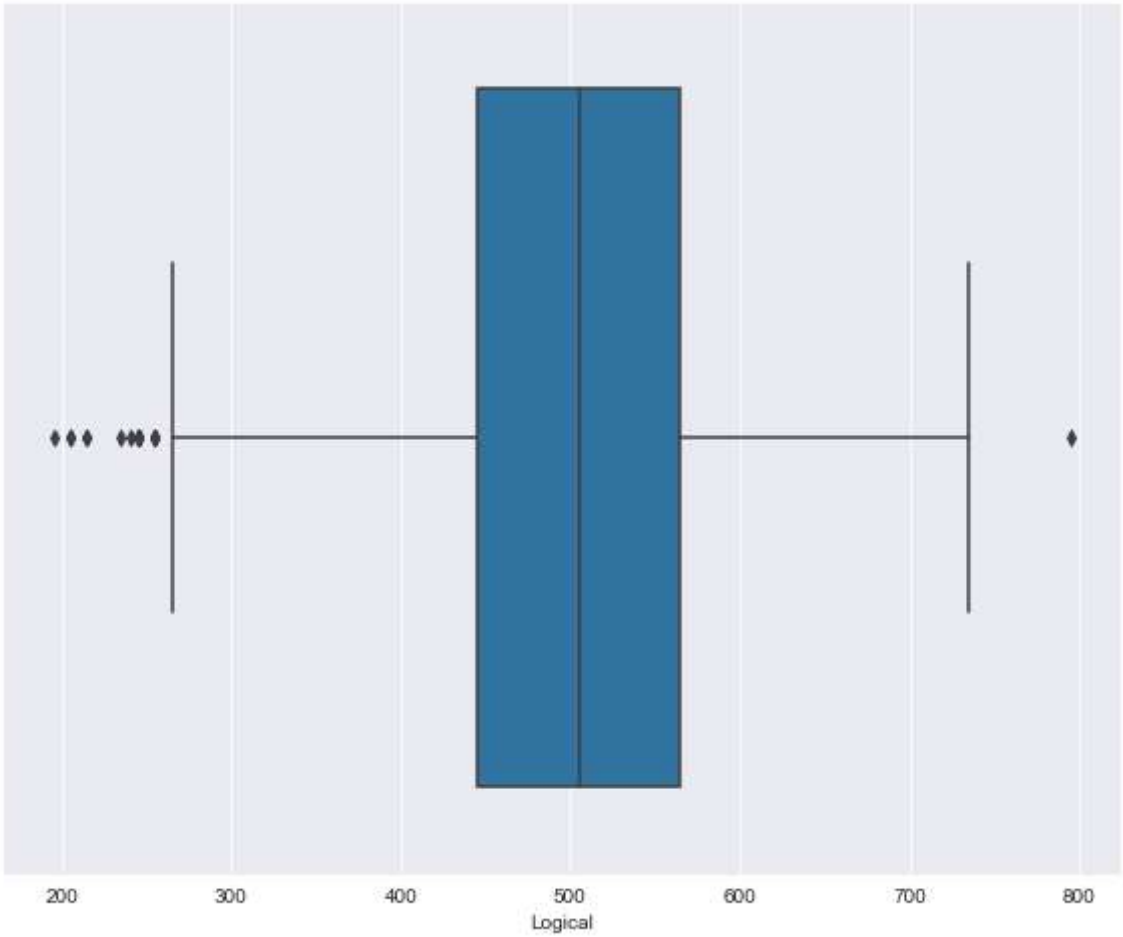
Finding the outliers in Logical column:

In [36]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['Logical'])
```

Out[36]:

```
<AxesSubplot:xlabel='Logical'>
```



From the above plot we can say that there are some outliers

In [37]:

```
# Outliers
df['Logical'][(df['Logical']<260) | (df['Logical']>790)].reset_index()
```

Out[37]:

	index	Logical
0	101	255
1	133	205
2	207	245
3	345	215
4	628	215
5	1014	795
6	1160	255
7	1439	245
8	2141	255
9	2265	255
10	2796	240
11	2830	195
12	2891	245
13	3119	245
14	3159	245
15	3784	205
16	3876	235
17	3953	245

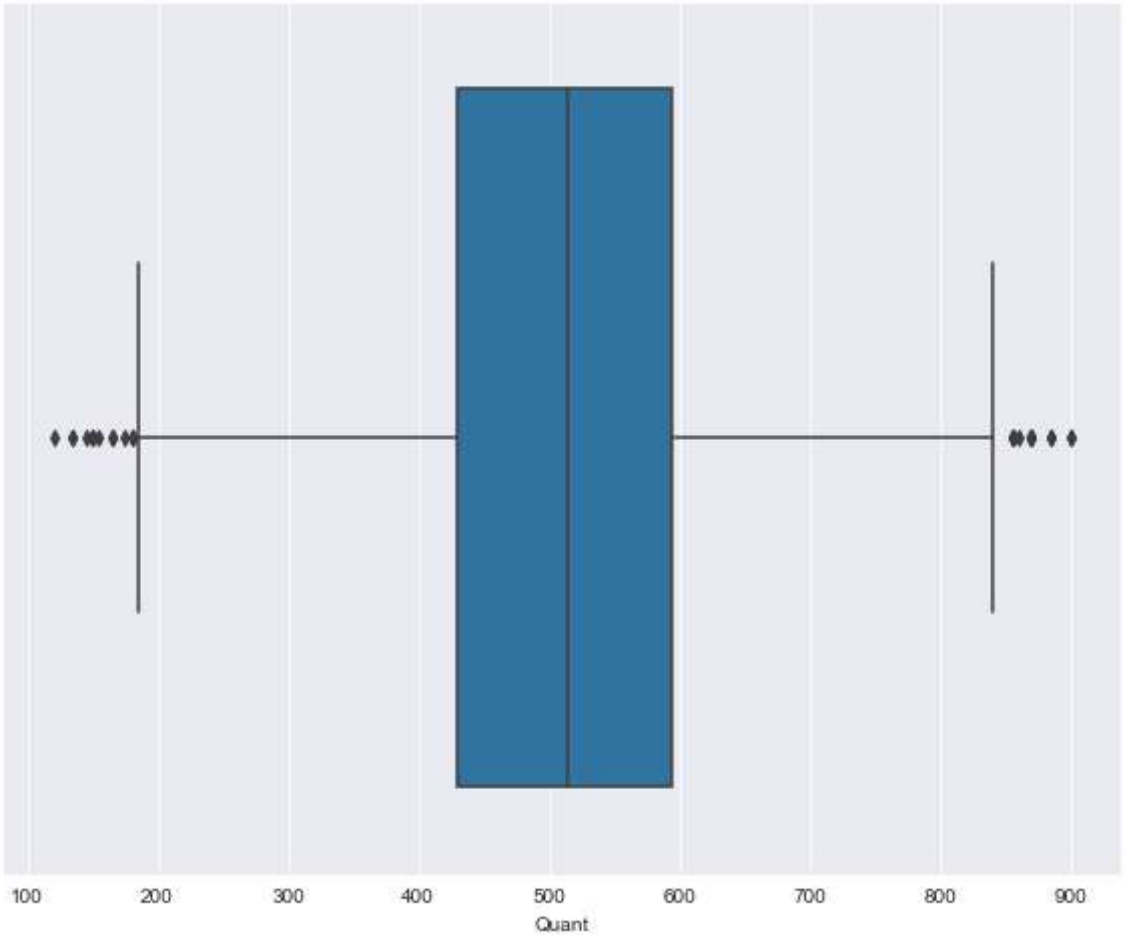
Finding the outliers in Quant column:

In [38]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['Quant'])
```

Out[38]:

```
<AxesSubplot:xlabel='Quant'>
```



From the above plot we can say that there are some outliers

In [40]:

```
# Outliers
df['Quant'][(df['Quant']<190) | (df['Quant']>850)].reset_index()
```

Out[40]:

	index	Quant
0	195	870
1	522	165
2	564	180
3	698	860
4	783	180
5	899	870
6	1148	175
7	1310	870
8	1661	870
9	1815	135
10	2007	855
11	2134	145
12	2155	870
13	2411	885
14	2437	120
15	2490	900
16	2616	135
17	2932	165
18	2957	900
19	2988	885
20	3165	855
21	3239	855
22	3258	150
23	3283	150
24	3544	185
25	3616	155

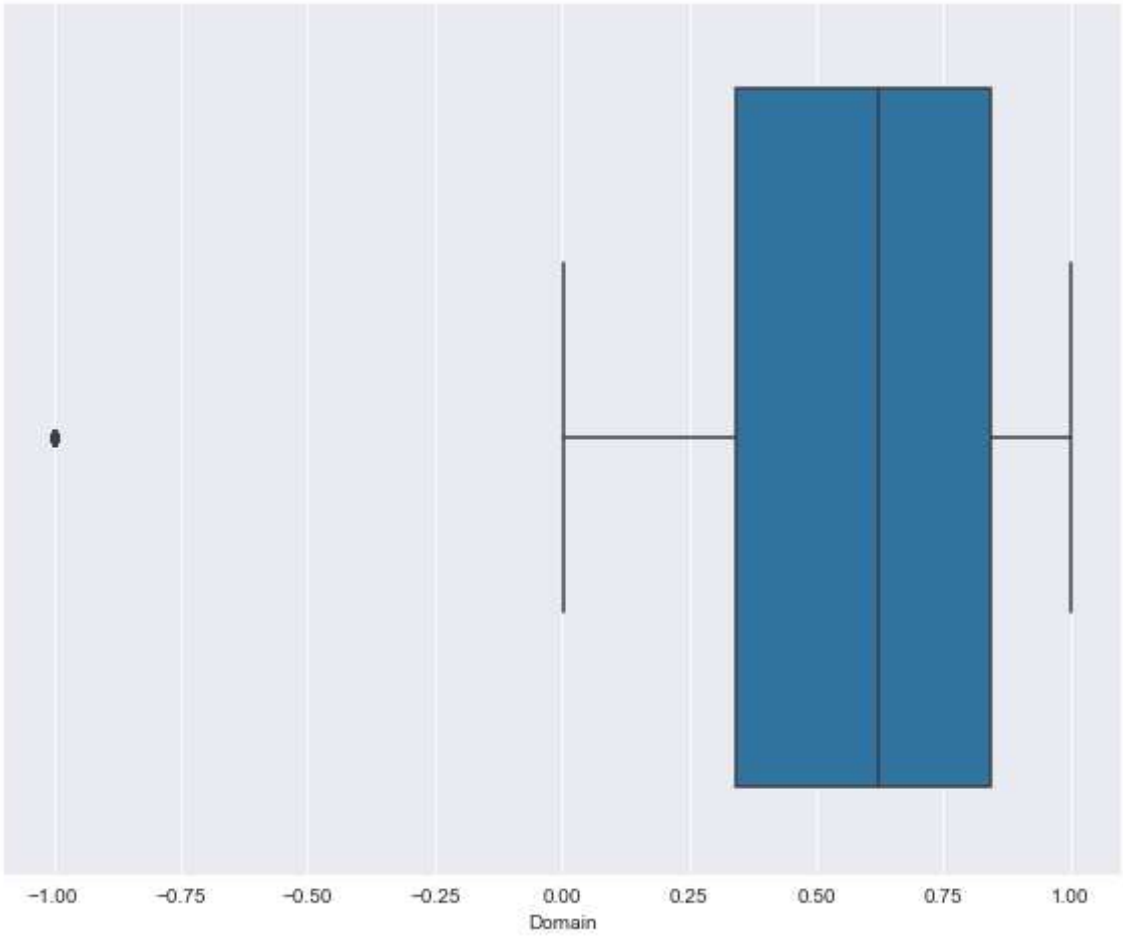
Finding the outliers in Domain column:

In [41]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['Domain'])
```

Out[41]:

```
<AxesSubplot:xlabel='Domain'>
```



In [43]:

```
# Outliers
df['Domain'][df['Domain'] <= -1].value_counts().reset_index()
```

Out[43]:

	index	Domain
0	-1.0	246

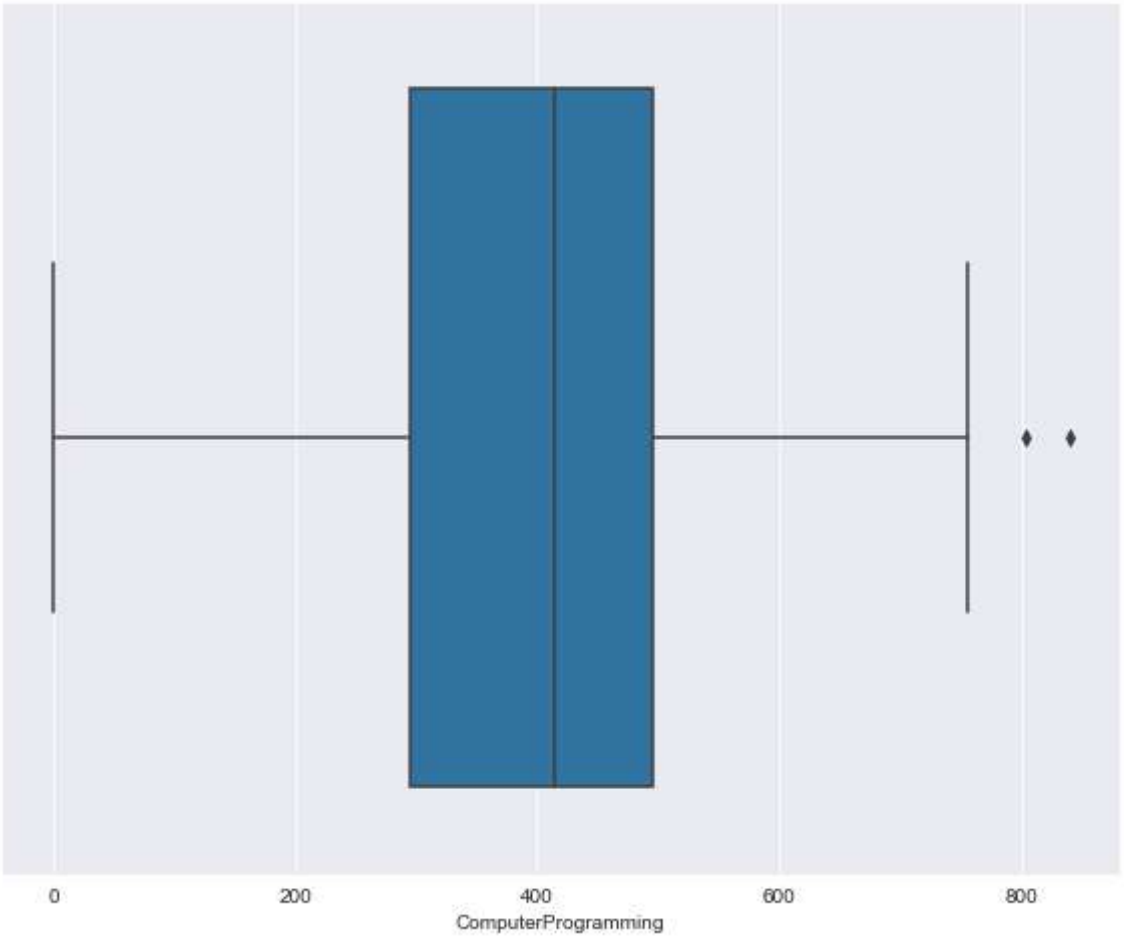
Finding the outliers in ComputerProgramming column:

In [44]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['ComputerProgramming'])
```

Out[44]:

```
<AxesSubplot:xlabel='ComputerProgramming'>
```



In [45]:

```
# Outliers
df['ComputerProgramming'][df['ComputerProgramming'] > 800].reset_index()
```

Out[45]:

	index	ComputerProgramming
0	64	804
1	1711	840

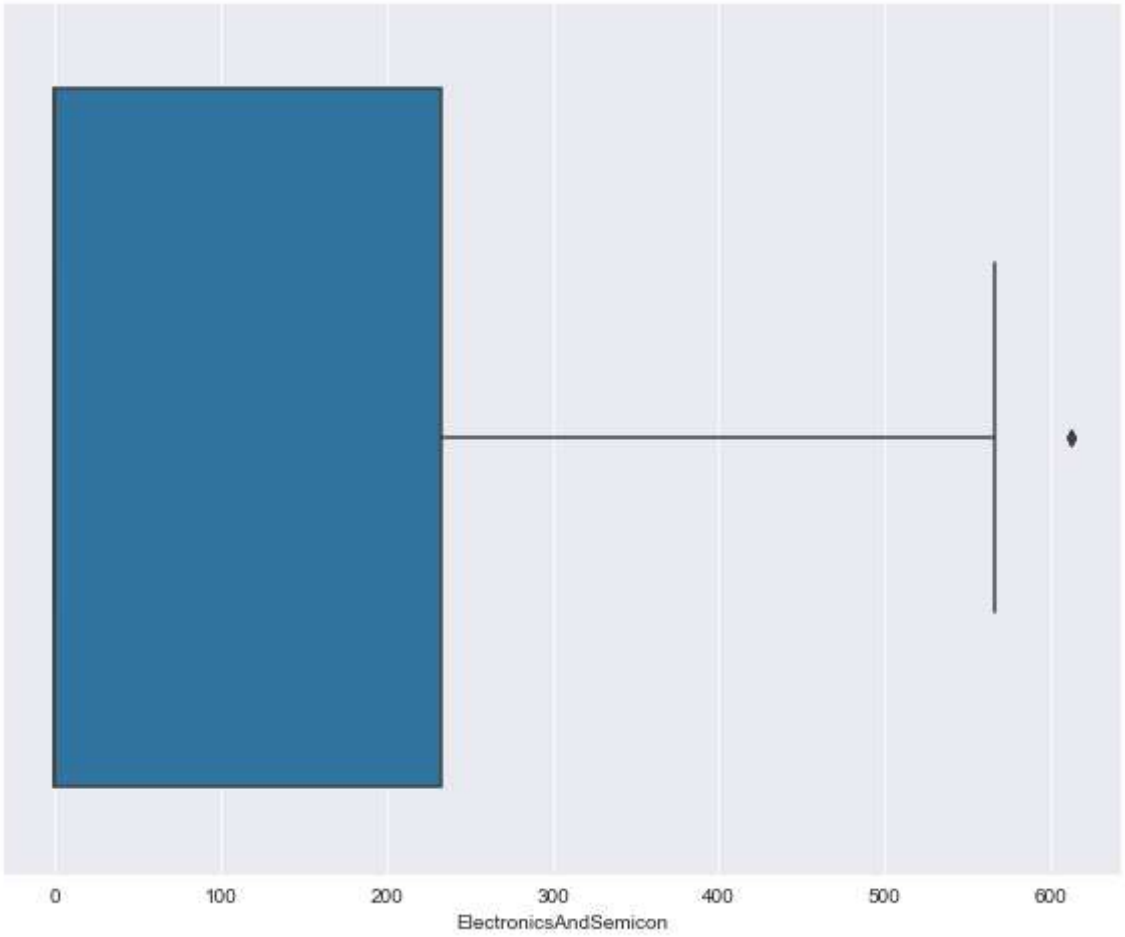
Finding the outliers in ElectronicsAndSemicon column:

In [46]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['ElectronicsAndSemicon'])
```

Out[46]:

```
<AxesSubplot:xlabel='ElectronicsAndSemicon'>
```



In [47]:

```
# Outliers
df['ElectronicsAndSemicon'][df['ElectronicsAndSemicon'] > 600].reset_index()
```

Out[47]:

	index	ElectronicsAndSemicon
0	1557	612
1	3889	612

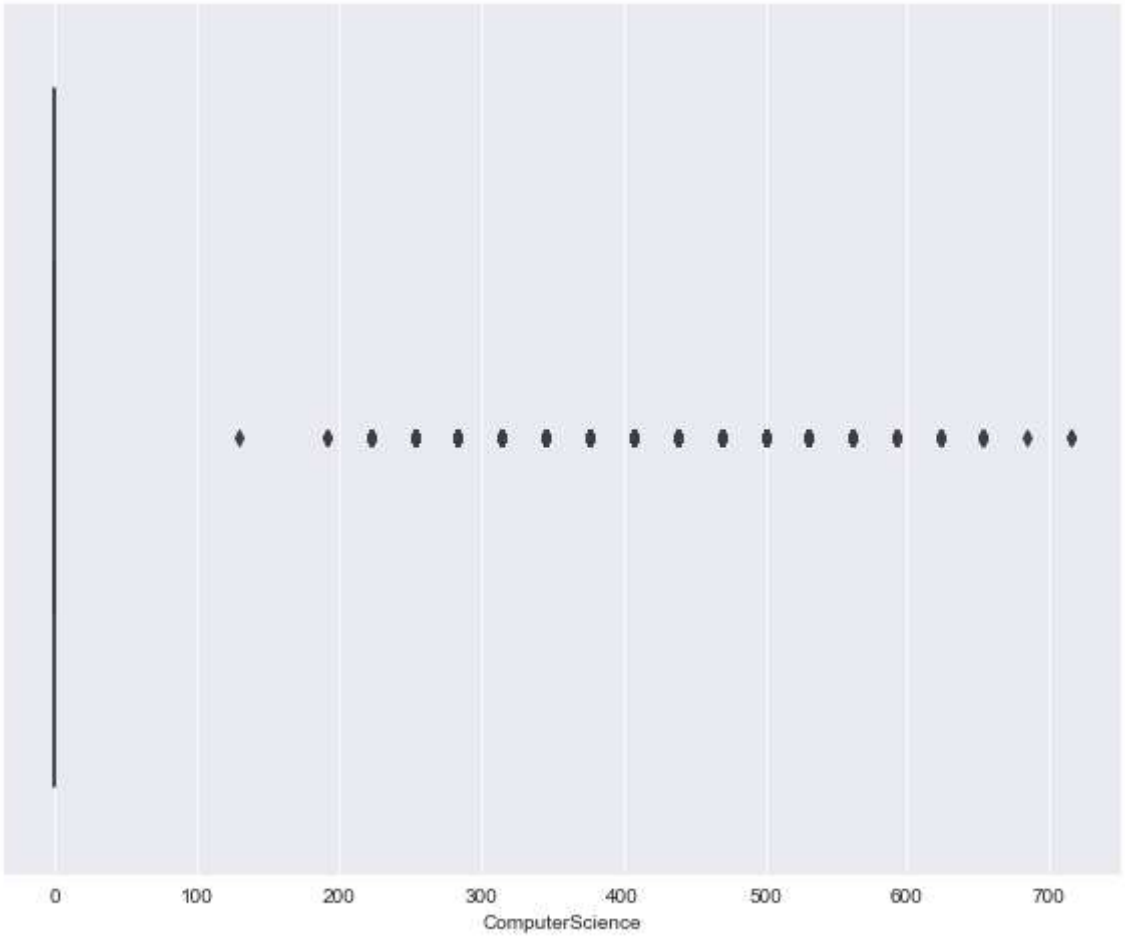
Frequency distribution of ComputerScience column:

In [48]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['ComputerScience'])
```

Out[48]:

```
<AxesSubplot:xlabel='ComputerScience'>
```



In [52]:

```
df['ComputerScience'][df['ComputerScience'] > 100].value_counts().reset_index()
```

Out[52]:

	index	ComputerScience
0	407	128
1	376	123
2	346	118
3	438	111
4	469	80
5	315	77
6	500	64
7	284	50
8	530	45
9	253	30
10	561	22
11	223	14
12	592	14
13	623	10
14	653	9
15	192	3
16	715	2
17	684	1
18	130	1

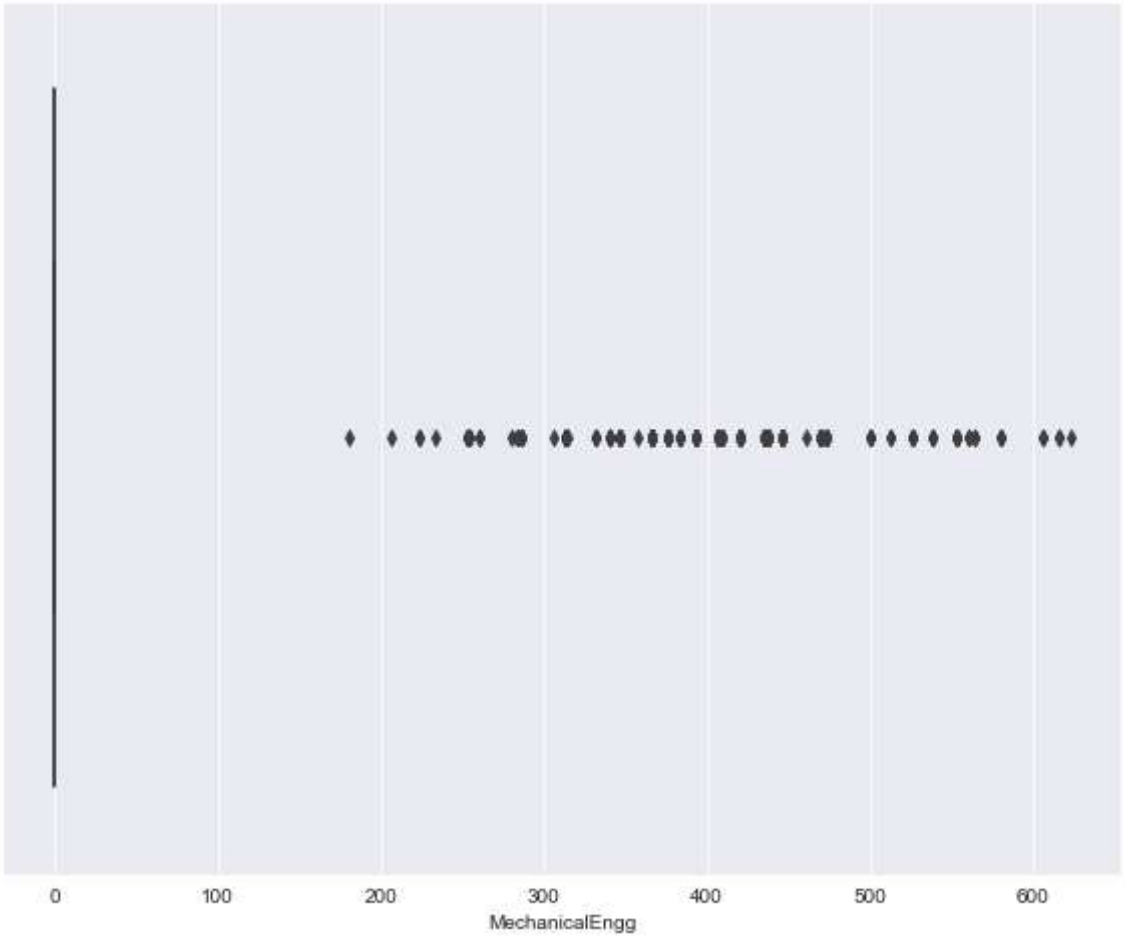
Frequency distribution of MechanicalEngg column:

In [51]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['MechanicalEngg'])
```

Out[51]:

```
<AxesSubplot:xlabel='MechanicalEngg'>
```



In [55]:

```
df['MechanicalEngg'][df['MechanicalEngg'] > 170].value_counts().reset_index()
```

Out[55]:

	index	MechanicalEngg
0	366	20
1	446	16
2	438	15
3	420	14
4	313	13
5	393	13
6	376	13
7	407	12
8	346	11
9	473	10
10	469	10
11	553	8
12	435	7
13	340	6
14	383	6
15	409	5
16	526	5
17	286	5
18	500	4
19	253	4
20	254	3
21	332	3
22	580	3
23	538	3
24	284	3
25	512	2
26	561	2
27	616	2
28	606	2
29	564	2
30	260	2
31	223	2
32	280	1
33	233	1
34	623	1

	index	MechanicalEngg
35	206	1
36	461	1
37	180	1
38	315	1
39	358	1
40	306	1

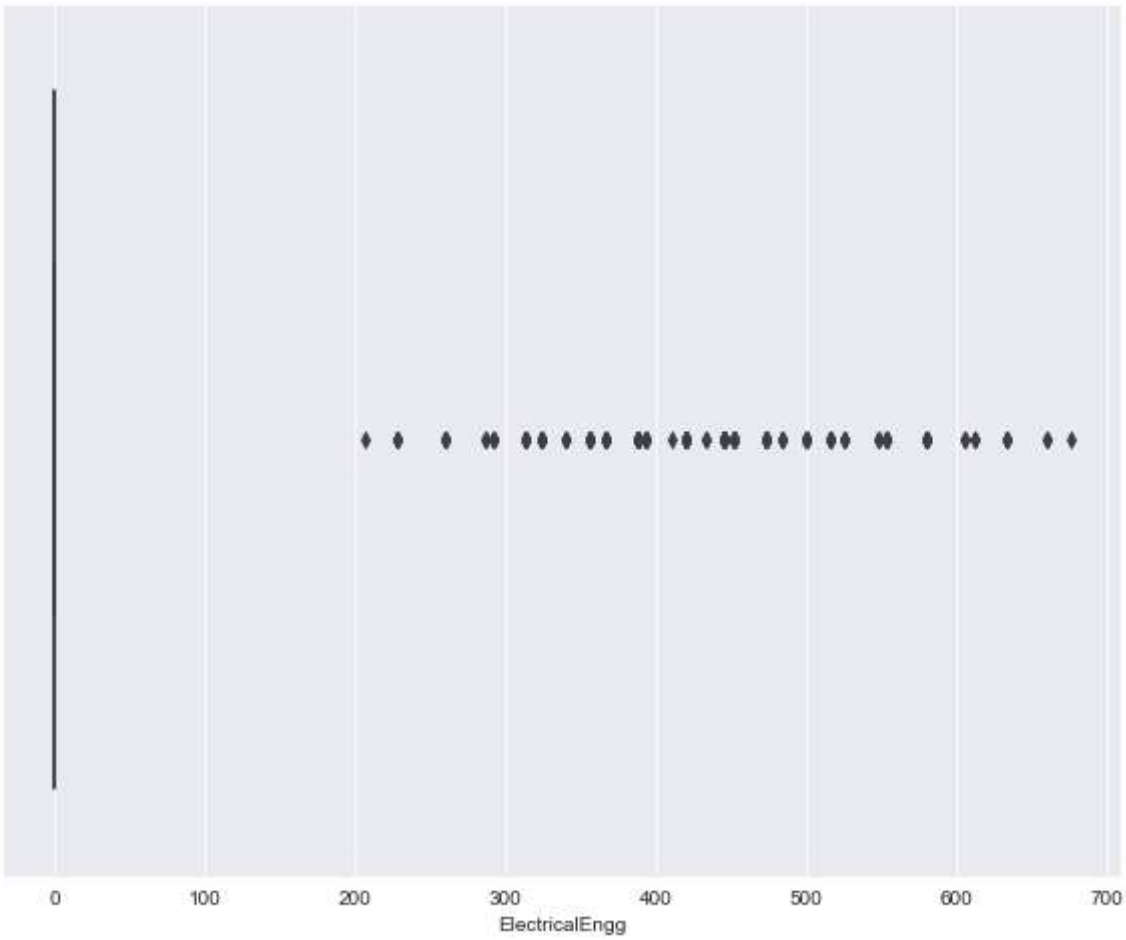
Frequency Distribution of ElectricalEngg column:

In [56]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['ElectricalEngg'])
```

Out[56]:

```
<AxesSubplot:xlabel='ElectricalEngg'>
```



In [57]:

```
df['ElectricalEngg'][df['ElectricalEngg'] > 200].value_counts().reset_index()
```

Out[57]:

	index	ElectricalEngg
0	420	16
1	446	14
2	388	10
3	473	10
4	452	10
5	356	9
6	500	8
7	580	8
8	393	7
9	324	7
10	366	7
11	553	6
12	313	6
13	516	5
14	260	4
15	292	4
16	633	3
17	526	3
18	228	3
19	340	3
20	484	3
21	660	2
22	433	2
23	606	2
24	548	2
25	286	2
26	612	2
27	411	1
28	676	1
29	206	1

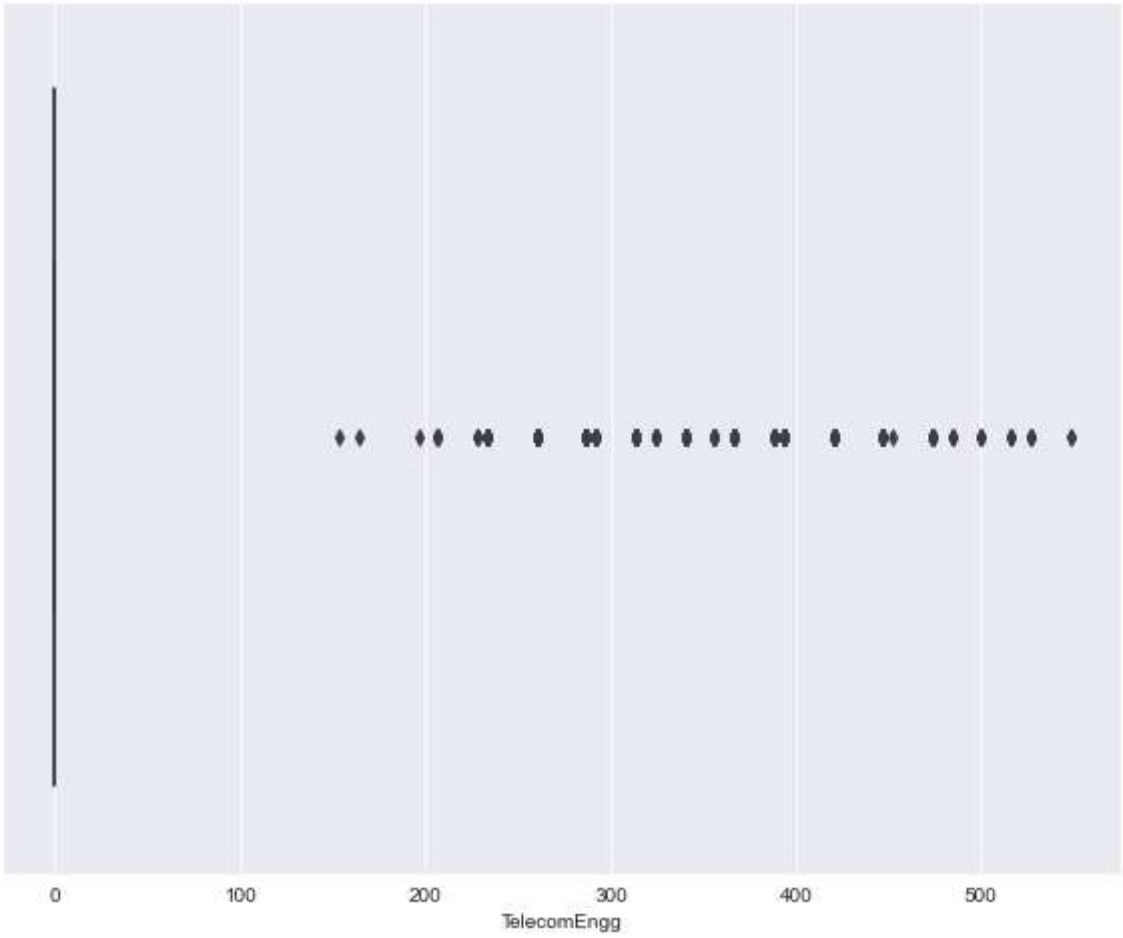
Frequency Distribution of TelecomEngg column:

In [58]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(df['TelecomEngg'])
```

Out[58]:

```
<AxesSubplot:xlabel='TelecomEngg'>
```



In [59]:

```
df['TelecomEngg'][df['TelecomEngg'] > 150].value_counts().reset_index()
```

Out[59]:

	index	TelecomEngg
0	393	45
1	366	43
2	260	35
3	313	34
4	340	32
5	286	31
6	420	26
7	446	16
8	388	14
9	233	14
10	473	13
11	292	13
12	356	12
13	324	11
14	206	8
15	500	5
16	526	5
17	484	4
18	516	4
19	228	3
20	548	2
21	196	1
22	164	1
23	153	1
24	452	1

Bivariate Analysis:

Correlation between the features using corr() function:

In [6]:

```
df.corr()
```

Out[6]:

	Salary	10percentage	12graduation	12percentage	CollegeTier	colle
Salary	1.000000	0.177373	-0.161383	0.170254	-0.179332	0.
10percentage	0.177373	1.000000	0.269957	0.643378	-0.126042	0.
12graduation	-0.161383	0.269957	1.000000	0.259166	0.027691	0.
12percentage	0.170254	0.643378	0.259166	1.000000	-0.100771	0.
CollegeTier	-0.179332	-0.126042	0.027691	-0.100771	1.000000	-0.
collegeGPA	0.130103	0.312538	0.086001	0.346137	-0.086781	1.
CollegeCityTier	0.015384	0.116707	-0.003016	0.130462	-0.101494	0.
GraduationYear	-0.010053	-0.013799	0.014457	-0.012933	-0.005557	0.
English	0.178219	0.350780	0.147925	0.212888	-0.183843	0.
Logical	0.179275	0.316014	0.105887	0.243571	-0.182811	0.
Quant	0.230627	0.317640	0.001379	0.312413	-0.251103	0.
Domain	0.104656	0.078563	-0.034163	0.074099	-0.061436	0.
ComputerProgramming	0.115665	0.053600	-0.047995	0.080818	-0.073644	0.
ElectronicsAndSemicon	0.000665	0.085179	-0.005891	0.117112	-0.031573	0.
ComputerScience	-0.100720	-0.018933	0.293439	-0.043534	0.001053	0.
MechanicalEngg	0.018475	0.050364	0.035459	0.037635	-0.021548	-0.
ElectricalEngg	-0.047598	0.074419	0.123751	0.064001	0.002594	0.
TelecomEngg	-0.022691	0.049378	0.023470	0.044201	0.000007	-0.
CivilEngg	0.037639	0.030002	-0.004727	0.005910	-0.033722	-0.
conscientiousness	-0.064148	0.067657	0.103329	0.058299	0.055174	0.
agreeableness	0.057423	0.136645	0.041182	0.103998	-0.038055	0.
extraversion	-0.010213	-0.004679	0.061956	-0.007486	0.009970	-0.
nueroticism	-0.054685	-0.132496	-0.074369	-0.094369	0.023778	-0.
openess_to_experience	-0.011312	0.036692	-0.015069	0.006332	-0.019179	0.

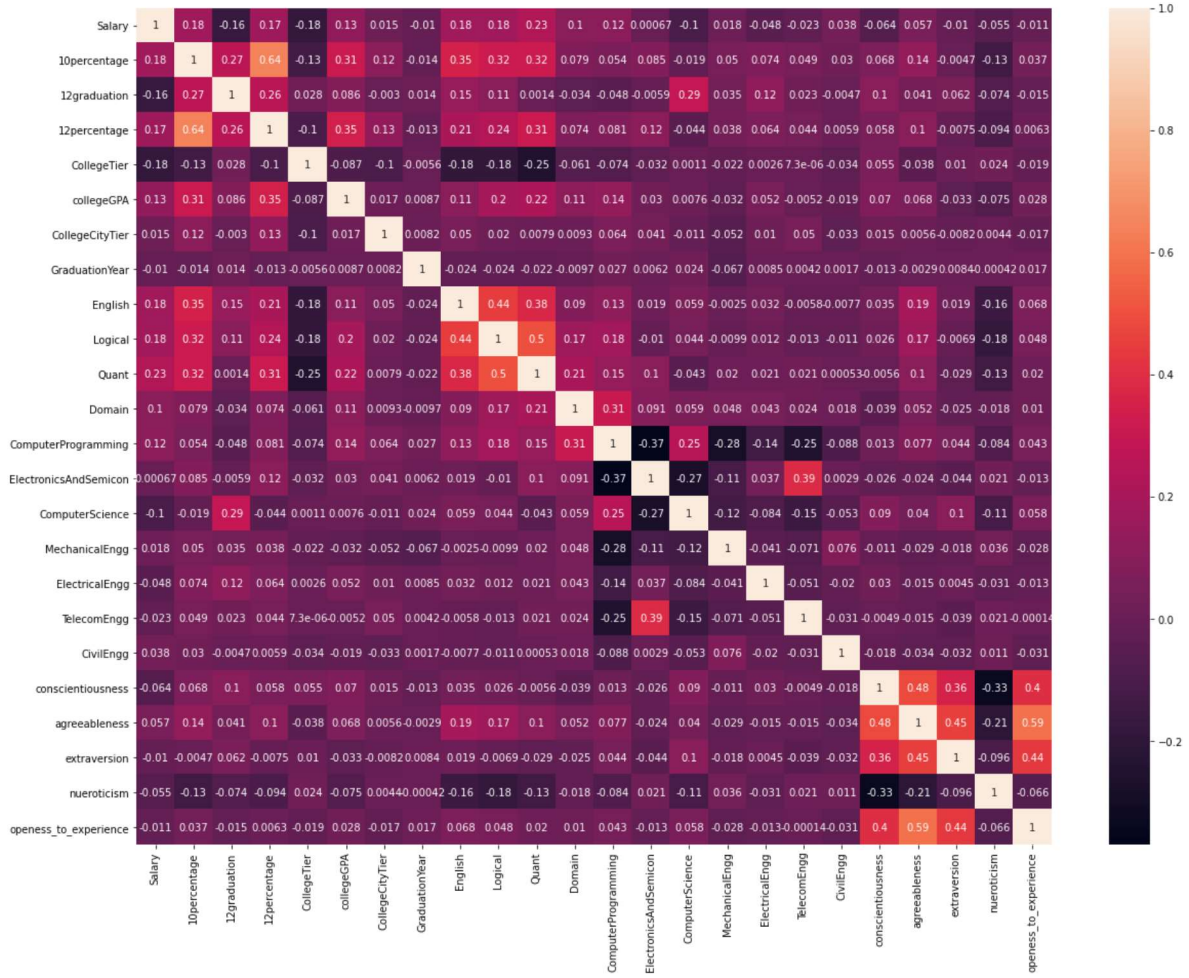
24 rows × 24 columns

As we can see above the most of the data points are between -1 to +1 where -1 means negatively correlated and +1 means highly positive relation

Correlation between the features using Heatmap:

In [12]:

```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr() , annot=True)
plt.show()
```



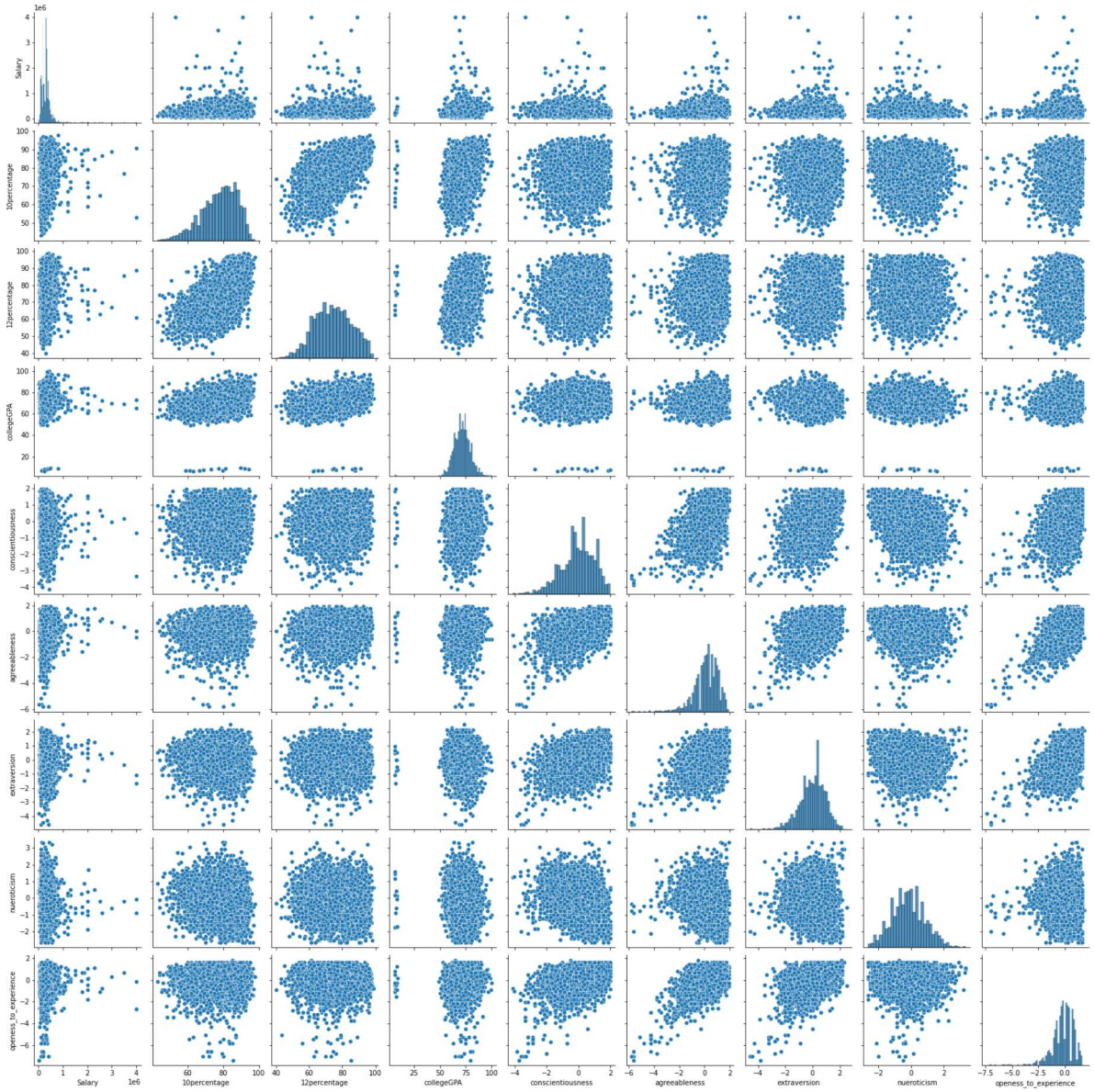
1. **From the above plot we can say that 10percentage and 12percentage is highly positively correlated with each other means if a student got a better marks in 10th standard then it is highly possible that he or she can get the better marks in 12th standard also**
2. **We can also say that the subjects English,Logical,Quant are also positively correlated with each other means if a student get better marks in one subject then it is highly possible that he or she will get better marks in other subjects also**
3. **As we can see that Conscientiousness, Agreeableness, Extraversion, and Openess_to_experience are also highly correlated with each other**

Correlation between the features using pairplot:

In [21]:

```
plt.figure(figsize=(20,15))
data = df[['Salary','10percentage','12percentage','collegeGPA', 'conscientiousness','agreea
sns.pairplot(data)
plt.show()
```

```
<Figure size 1440x1080 with 0 Axes>
```



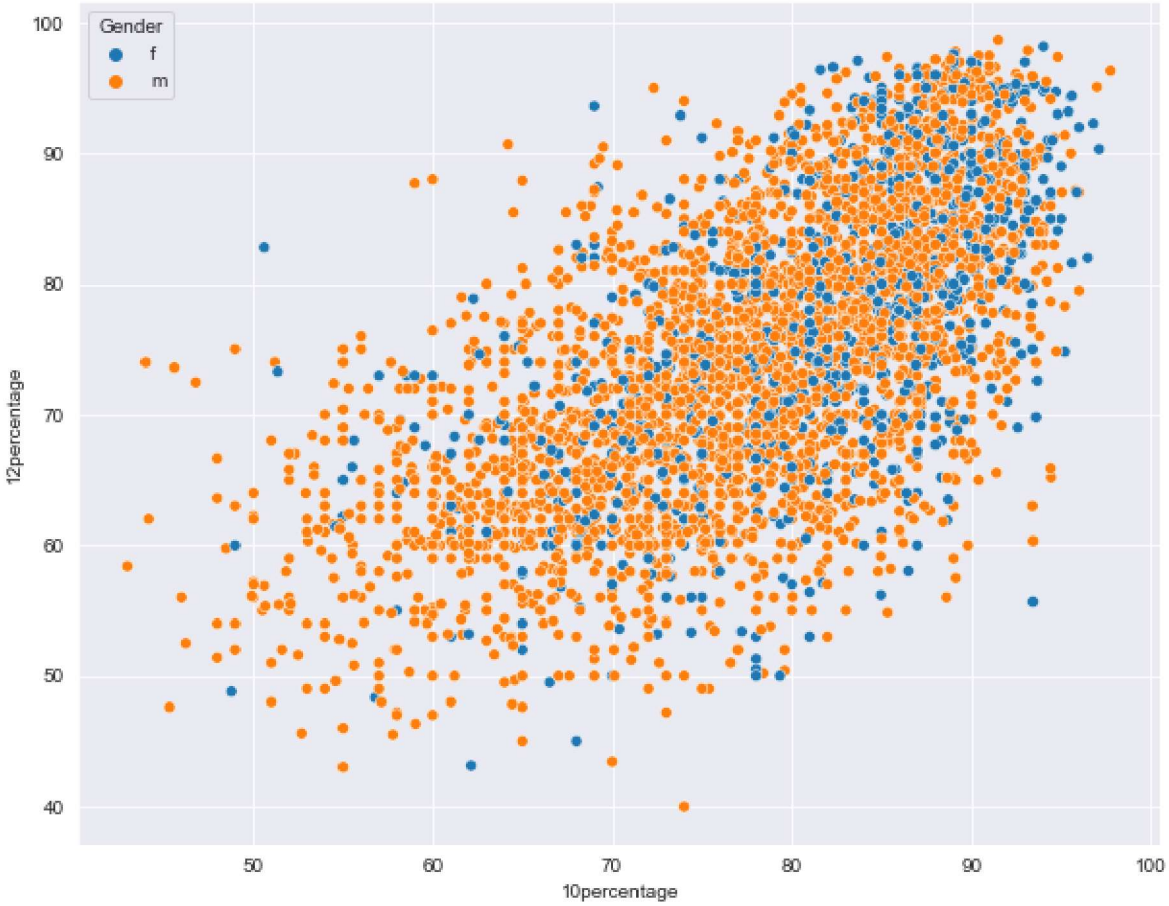
Correlation between the features using scatterplot:

In [23]:

```
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.scatterplot(data=df, x="10percentage", y="12percentage",hue="Gender")
```

Out[23]:

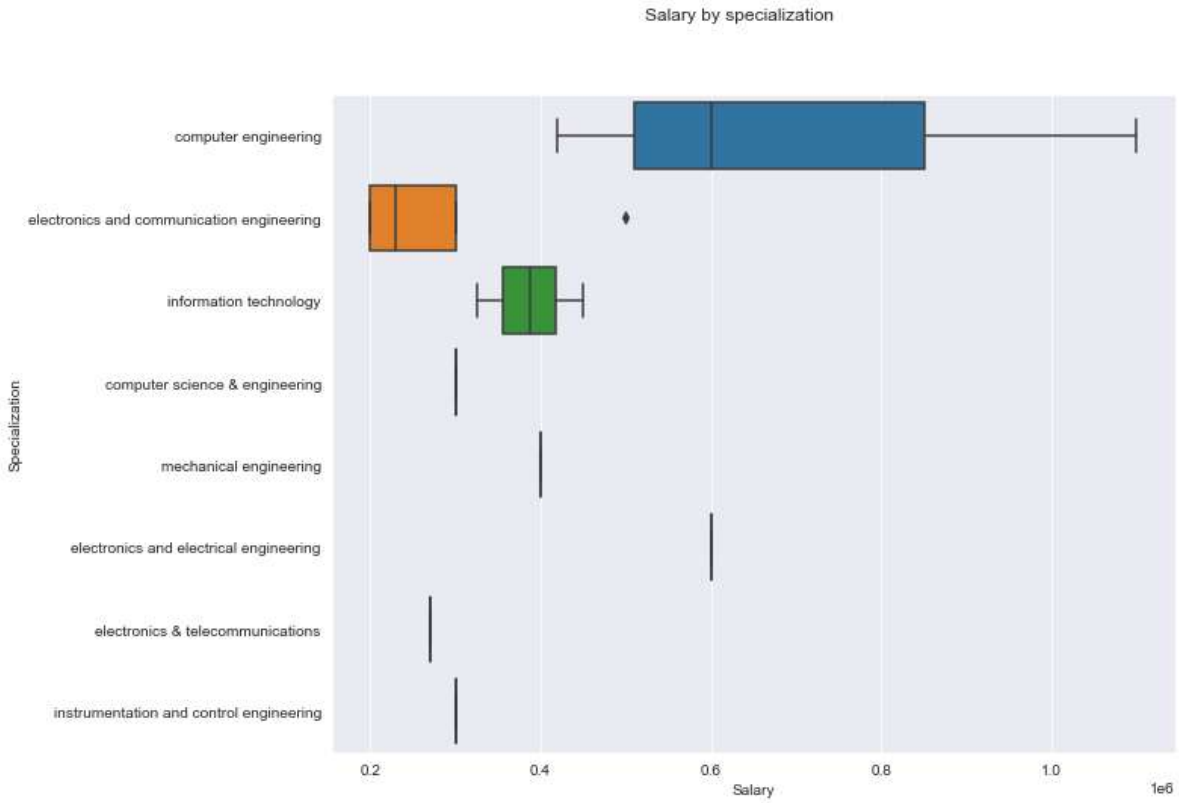
```
<AxesSubplot:xlabel='10percentage', ylabel='12percentage'>
```



Correlation between the features using boxplot:

In [27]:

```
# Salary by Specialization
plt.figure(figsize=(10,8))
sns.set_style("darkgrid")
sns.boxplot(x=df['Salary'].iloc[:15], y = df['Specialization'].iloc[:15])
plt.suptitle('Salary by specialization')
plt.show()
```



Research Questions:

Times of India article dated Jan 18, 2019 states that "After doing your Computer Science Engineering if you take up jobs as a Programming Analyst, Software Engineer, Hardware Engineer and Associate Engineer you can earn up to 2.5-3 lakhs as a fresh graduate." Test this claim with the data.

Hypothesis Testing:

In [46]:

```
df.head()
```

Out[46]:

	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	12graduation
0	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	2007
1	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	2007
2	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	2010
3	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	2007
4	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	2008

5 rows × 34 columns

In [28]:

```
# Normalize Salary for Better Visualization
df['n_sal']=df['Salary']/100000
```

In [29]:

```
df[['Designation','Specialization']][df['Designation']=='hardware engineer']
```

Out[29]:

	Designation	Specialization
197	hardware engineer	electrical engineering
802	hardware engineer	electronics and communication engineering
839	hardware engineer	electronics and communication engineering
1886	hardware engineer	electronics and communication engineering
2070	hardware engineer	electronics and communication engineering
2533	hardware engineer	electronics and communication engineering
3438	hardware engineer	electronics engineering
3547	hardware engineer	electronics and communication engineering

In [32]:

```
print('Average Salary :')
print('Programmer Analyst :',round(df['n_sal'][(df['GraduationYear']==2014) & (df['Designat
print('Software Engineer :',round(df['n_sal'][(df['GraduationYear']==2014) & (df['Designati
print('Hardware Engineer :',round(df['n_sal'][(df['GraduationYear']==2014) &(df['Designatio
print('Associate Engineer :',round(df['n_sal'][(df['GraduationYear']==2014) &(df['Designati
```

```
Average Salary :
Programmer Analyst : 3.02
Software Engineer : 3.4
Hardware Engineer : nan
Associate Engineer : 3.33
```

In [33]:

```
# Sample Data for Required Employees
sample = [3.16,3.6,0,3.5]
sample = np.array(sample)
```

In [34]:

```
# Necessary variables initialization ex- sample mean
sample_size = len(sample)
sample_mean = np.mean(sample)
sample_mean
```

Out[34]:

```
2.565
```

In [36]:

```
# Sample Standard Devation
import math
sample_std = math.sqrt(sum([(i-sample_mean)**2 for i in sample]) / 3)
print('Sample Standard Deviation :', sample_std)
```

```
Sample Standard Deviation : 1.7203391138571102
```

In [37]:

```
# Calulating T-Score
def t_score(pop_mean, sample_mean, sample_std, sample_size):
    numerator = sample_mean - pop_mean
    denomenator = sample_std / (sample_size**0.5)
    return numerator / denomenator
```

In [38]:

```
# Necessary variables initialization ex- sample mean,population mean
pop_mean = 2.75
sample_mean = 3.34
sample_std = 0.21
sample_size = 4
```

In [39]:

```
# Calling T-score Function
t_sc = t_score(pop_mean, sample_mean, sample_std, sample_size)
print('t-score :', t_sc)
```

```
t-score : 5.619047619047618
```

In [41]:

```
# Setting the Confidence Level

# Two Tail - Deciding the Significance Level & Calculating the t-critical value
from scipy.stats import t
confidence_level = 0.95
alpha = 1 - confidence_level
t_critical = t.ppf(1-alpha/2, df = 3)
print('t_critical :', t_critical)
```

```
t_critical : 3.182446305284263
```

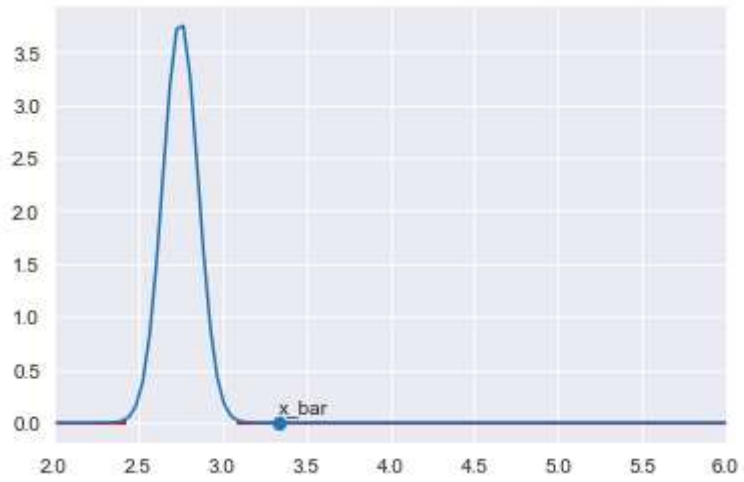
In [43]:

```
# Visualizing the Sampling Distribution with Rejection Regions
from scipy.stats import norm
# Defining the x min & x max
x_min = 2
x_max =6

# Defining the Sampling Distribution mean & std
mean = pop_mean
std = sample_std / (sample_size**0.5)

# Ploting the graph and setting the x limits
x = np.linspace(x_min, x_max, 100)
y = norm.pdf(x, mean, std)
plt.xlim(x_min, x_max)
plt.plot(x, y)

# Computing the left and right critical values of Two tailed Test
t_critical_left = pop_mean + (-t_critical * std)
t_critical_right = pop_mean + (t_critical * std)

print('t_critical_left :', t_critical_left)
print('t_critical_right :', t_critical_right)

# Shading the left rejection region
x_left = np.linspace(x_min, t_critical_left, 100)
y_left = norm.pdf(x_left, mean, std)
plt.fill_between(x_left, y_left, color='red')

# Shading the right rejection region
x_right = np.linspace(t_critical_right, x_max, 100)
y_right = norm.pdf(x_right, mean, std)
plt.fill_between(x_right, y_right, color='red')

plt.scatter(sample_mean, 0)
plt.annotate("x_bar", (sample_mean, 0.1))
```

```
t_critical_left : 2.4158431379451524
t_critical_right : 3.0841568620548476
```

Out[43]:

```
Text(3.34, 0.1, 'x_bar')
```



In [44]:

```
# Compairing the Table Value and T-score value

# Conclusion using t-test

if np.abs(t_sc) > t_critical:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
```

```
Reject Null Hypothesis
```

In [45]:

```
# Conclusion using p-test
p_value = 2 * (1.0 - norm.cdf(np.abs(t_sc)))

print("p_value = ", p_value)

if p_value < alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
```

```
p_value =  1.9201293444126577e-08
Reject Null Hypothesis
```

Feature Transformation:

Column Standardization for Numerical Features:

In [47]:

```
# Column standardization using MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [48]:

```
# Standardizing Salary Column
scaled_sal = scaler.fit_transform(data['Salary'].values.reshape(-1,1))

# First 20 Scaled Values (Salary column)
print(scaled_sal[:20])
```

```
[[0.09709962]
 [0.11727617]
 [0.07313997]
 [0.26860025]
 [0.04161412]
 [0.0668348 ]
 [0.0668348 ]
 [0.09205549]
 [0.14249685]
 [0.04918033]
 [0.14249685]
 [0.10466583]
 [0.0592686 ]
 [0.04161412]
 [0.0668348 ]
 [0.07944515]
 [0.07313997]
 [0.05422446]
 [0.02143758]
 [0.07313997]]
```

In [49]:

```
# Standardizing 10th percent Column
scaled_10 = scaler.fit_transform(data['10percentage'].values.reshape(-1,1))

# First 20 Scaled Values (Salary column)
print(scaled_10[:20])
```

```
[[0.75420015]
 [0.7742878 ]
 [0.7669832 ]
 [0.7779401 ]
 [0.63915267]
 [0.8568298 ]
 [0.78670562]
 [0.89481373]
 [0.85829072]
 [0.62089116]
 [0.83272462]
 [0.69393718]
 [0.74141709]
 [0.32505478]
 [0.69758948]
 [0.48466034]
 [0.7815924 ]
 [0.32505478]
 [0.4017531 ]
 [0.65741417]]
```

In [50]:

```
# Standardizing 12th percent Column
scaled_12 = scaler.fit_transform(data['12percentage'].values.reshape(-1,1))

# First 20 Scaled Values (Salary column)
print(scaled_12[:20])
```

```
[[0.95059625]
 [0.76660988]
 [0.48040886]
 [0.7427598 ]
 [0.62691652]
 [0.80068143]
 [0.46848382]
 [0.86882453]
 [0.87223169]
 [0.54855196]
 [0.74446337]
 [0.78364566]
 [0.50817717]
 [0.38160136]
 [0.67972743]
 [0.41618399]
 [0.59284497]
 [0.45417376]
 [0.42248722]
 [0.37819421]]
```

Column Standardization for Categorical Features:

In [51]:

```
# One-hot Encoding of Gender column
dummies = pd.get_dummies(df[['Gender']])
dummies
```

Out[51]:

	Gender_f	Gender_m
0	1	0
1	0	1
2	1	0
3	0	1
4	0	1
...	...	...
3993	0	1
3994	1	0
3995	0	1
3996	1	0
3997	1	0

3998 rows × 2 columns

In [53]:

```
df1 = pd.concat([df,dummies],axis='columns')
df1.head()
```

Out[53]:

	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	12graduation
0	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	2007
1	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	2007
2	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	2010
3	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	2007
4	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	2008

5 rows × 36 columns

In [56]:

```
finaldf = df1.drop(['Gender','Gender_f'],axis='columns')
finaldf
```

Out[56]:

	Salary	DOJ	DOL	Designation	JobCity	DOB	10percentage	12graduatior
0	420000	2012-06-01	present	senior quality engineer	Bangalore	1990-02-19	84.30	2007
1	500000	2013-09-01	present	assistant manager	Indore	1989-10-04	85.40	2007
2	325000	2014-06-01	present	systems engineer	Chennai	1992-08-03	85.00	2010
3	1100000	2011-07-01	present	senior software engineer	Gurgaon	1989-12-05	85.60	2007
4	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	1991-02-27	78.00	2008
...	...	...	...	...	...	...	...	..
3993	280000	2011-10-01	2012-10-01 00:00:00	software engineer	New Delhi	1987-04-15	52.09	2006
3994	100000	2013-07-01	2013-07-01 00:00:00	technical writer	Hyderabad	1992-08-27	90.00	2009
3995	320000	2013-07-01	present	associate software engineer	Bangalore	1991-07-03	81.86	2008
3996	200000	2014-07-01	2015-01-01 00:00:00	software developer	Asifabadbanglore	1992-03-20	78.72	2010
3997	400000	2013-02-01	present	senior systems engineer	Chennai	1991-02-26	70.60	2008

3998 rows × 34 columns

In [ ]: