Dataset Discription:The Given data set is Aspiring Minds from the Aspiring Mind Employment Outcome 2015 (AMEO) and it contains approximately 4000 rows and 40 columns The dataset contains the employment outcomes of all engineering graduates these employees having different types of deppendent variables like Designation,JobCity,10percentage,Salary etc and the data set contains the both numerical and categorical data

In [ ]:
```
#Importing libraries

import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
```

In [ ]:
```
#importing dataset

from google.colab import files
dataset= files.upload()
```

Choose Files No file selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving data.xlsx to data.xlsx
```

In [ ]:
```
df = pd.read_excel("data.xlsx")
df.head()
```

Out[ ]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	ComputerScience	MechanicalEngg
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	...	-1	-1
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	...	-1	-1
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	...	-1	-1
3	train	267447	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	...	-1	-1
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	...	-1	-1

5 rows × 39 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(3998, 39)
```

In [ ]:
```
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 [ ]:
```
df.describe()
```

Out[ ]:

	ID	Salary	10percentage	12graduation	12percentage	CollegeID	CollegeTier	collegeGPA	CollegeCityID	Colleg
count	3.998000e+03	3.998000e+03	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	399
mean	6.637945e+05	3.076998e+05	77.925443	2008.087544	74.466366	5156.851426	1.925713	71.486171	5156.851426	
std	3.632182e+05	2.127375e+05	9.850162	1.653599	10.999933	4802.261482	0.262270	8.167338	4802.261482	
min	1.124400e+04	3.500000e+04	43.000000	1995.000000	40.000000	2.000000	1.000000	6.450000	2.000000	
25%	3.342842e+05	1.800000e+05	71.680000	2007.000000	66.000000	494.000000	2.000000	66.407500	494.000000	
50%	6.396000e+05	3.000000e+05	79.150000	2008.000000	74.400000	3879.000000	2.000000	71.720000	3879.000000	
75%	9.904800e+05	3.700000e+05	85.670000	2009.000000	82.600000	8818.000000	2.000000	76.327500	8818.000000	
max	1.298275e+06	4.000000e+06	97.760000	2013.000000	98.700000	18409.000000	2.000000	99.930000	18409.000000	

8 rows × 27 columns

In [ ]:
```
#Dropping the unwanted columns

df.drop("Unnamed: 0", axis=1, inplace=True)
df.drop("DOB", axis=1, inplace=True)
df.drop("CollegeID", axis=1, inplace=True)
```

In [ ]:
```
print((df['10percentage'] <=10).sum())
print((df['12percentage'] <=10).sum())
print((df['collegeGPA'] <=10).sum())
```

```
0
0
12
```

In [ ]:
```
#converting 12 entries off college gpa to percentage
df.loc[df['collegeGPA']<=10,'collegeGPA'] = (df.loc[df['collegeGPA']<=10,'collegeGPA']/10)*100
```

In [ ]:
```
print((df==0).sum()[(df==0).sum() > 0])
```

```
10board             350
12board             359
CollegeCityTier    2797
GraduationYear        1
dtype: int64
```

In [ ]:
```
df = df[~df['GraduationYear'].isin([-1, 0])]
```

In [ ]:
```
print((df==0).sum()[(df==0).sum() > 0])
```

```
10board             350
12board             359
CollegeCityTier    2796
dtype: int64
```

In [ ]:
```
df['DOJ'] = pd.to_datetime(df['DOJ'])
df['YearsExperience'] = (df['DOJ'].dt.year - df['GraduationYear']).astype(int)
print(df[['DOJ', 'GraduationYear', 'YearsExperience']])
```

```
            DOJ  GraduationYear  YearsExperience
0    2012-06-01          2011.0                1
1    2013-09-01          2012.0                1
2    2014-06-01          2014.0                0
3    2011-07-01          2011.0                0
4    2014-03-01          2012.0                2
...         ...             ...              ...
3993 2011-10-01          2010.0                1
3994 2013-07-01          2013.0                0
3995 2013-07-01          2012.0                1
3996 2014-07-01          2014.0                0
3997 2013-02-01          2012.0                1

[3998 rows x 3 columns]
```

In [ ]:
```
df.head()
```

Out[ ]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	Domain	ComputerProgramming	E
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	...	0.635979	445.000000	
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	...	0.960603	451.301278	
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	...	0.450877	395.000000	
3	train	267447	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	...	0.974396	615.000000	
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	...	0.124502	451.301278	

5 rows × 36 columns

In [ ]:
```
#convert "present" to today's date
df['DOL'] = df['DOL'].replace('present', datetime.now().date())
```

In [ ]:
```
# convert the column to datetime data type
df['DOL'] = pd.to_datetime(df['DOL'])
df['DOJ'] = pd.to_datetime(df['DOJ'])
```

In [ ]:
```
#Droping the unwanted columns
df= df.drop(columns = ['MechanicalEngg','ElectricalEngg','TelecomEngg','CivilEngg'])
```

In [ ]:
```
df['10board'] = df['10board'].astype(str).replace({'0':np.nan})
df['12board'] = df['12board'].astype(str).replace({'0':np.nan})
df['GraduationYear'] = df['GraduationYear'].replace({0:np.nan})
df['JobCity'] = df['JobCity'].astype(str).replace({'-1':np.nan})
df['Domain'] = df['Domain'].replace({-1:np.nan})

df['ComputerProgramming'] = df['ComputerProgramming'].replace({-1:np.nan})
df['ElectronicsAndSemicon'] = df['ElectronicsAndSemicon'].replace({-1:0})
df['ComputerScience'] = df['ComputerScience'].replace({-1:0})
```

In [ ]:
```
#imputing values in numerical columns

df['Domain'].fillna(df['Domain'].mean(), inplace = True)
df['ComputerProgramming'].fillna(df['ComputerProgramming'].mean(), inplace = True)
```

In [ ]:
```
#imputing values in categorical columns

df['10board'].fillna(df['10board'].mode()[0], inplace = True)
```

```
df['12board'].fillna(df['12board'].mode()[0], inplace = True)
df['GraduationYear'].fillna(df['GraduationYear'].mode()[0], inplace = True)
df['JobCity'].fillna(df['JobCity'].mode()[0], inplace = True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	Quant	Domain	ComputerProgram
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	...	525	0.635979	445.0(
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	...	780	0.960603	451.3(
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	...	370	0.450877	395.0(
3	train	267447	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	...	625	0.974396	615.0(
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	...	465	0.124502	451.3(

5 rows × 35 columns

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3998 entries, 0 to 3997
Data columns (total 35 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   float64
 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   float64
 28  ElectronicsAndSemicon  3998 non-null   int64
 29  ComputerScience        3998 non-null   int64
 30  conscientiousness      3998 non-null   float64
 31  agreeableness          3998 non-null   float64
 32  extraversion           3998 non-null   float64
 33  nueroticism            3998 non-null   float64
 34  openess_to_experience  3998 non-null   float64
dtypes: datetime64[ns](2), float64(11), int64(12), object(10)
memory usage: 1.1+ MB
```

Seprating the Columns

In [ ]:
```
#Numerical columns
numerical_col = df.select_dtypes(include=['float64', 'int64']).columns
num_df=df[numerical_col]
```

In [ ]:
```
num_df.columns
```

Out[ ]:
```
Index(['ID', 'Salary', '10percentage', '12graduation', '12percentage',
       'CollegeID', 'CollegeTier', 'collegeGPA', 'CollegeCityID',
       'CollegeCityTier', 'GraduationYear', 'English', 'Logical', 'Quant',
       'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon',
       'ComputerScience', 'conscientiousness', 'agreeableness', 'extraversion',
       'nueroticism', 'openess_to_experience', 'YearsExperience'],
      dtype='object')
```

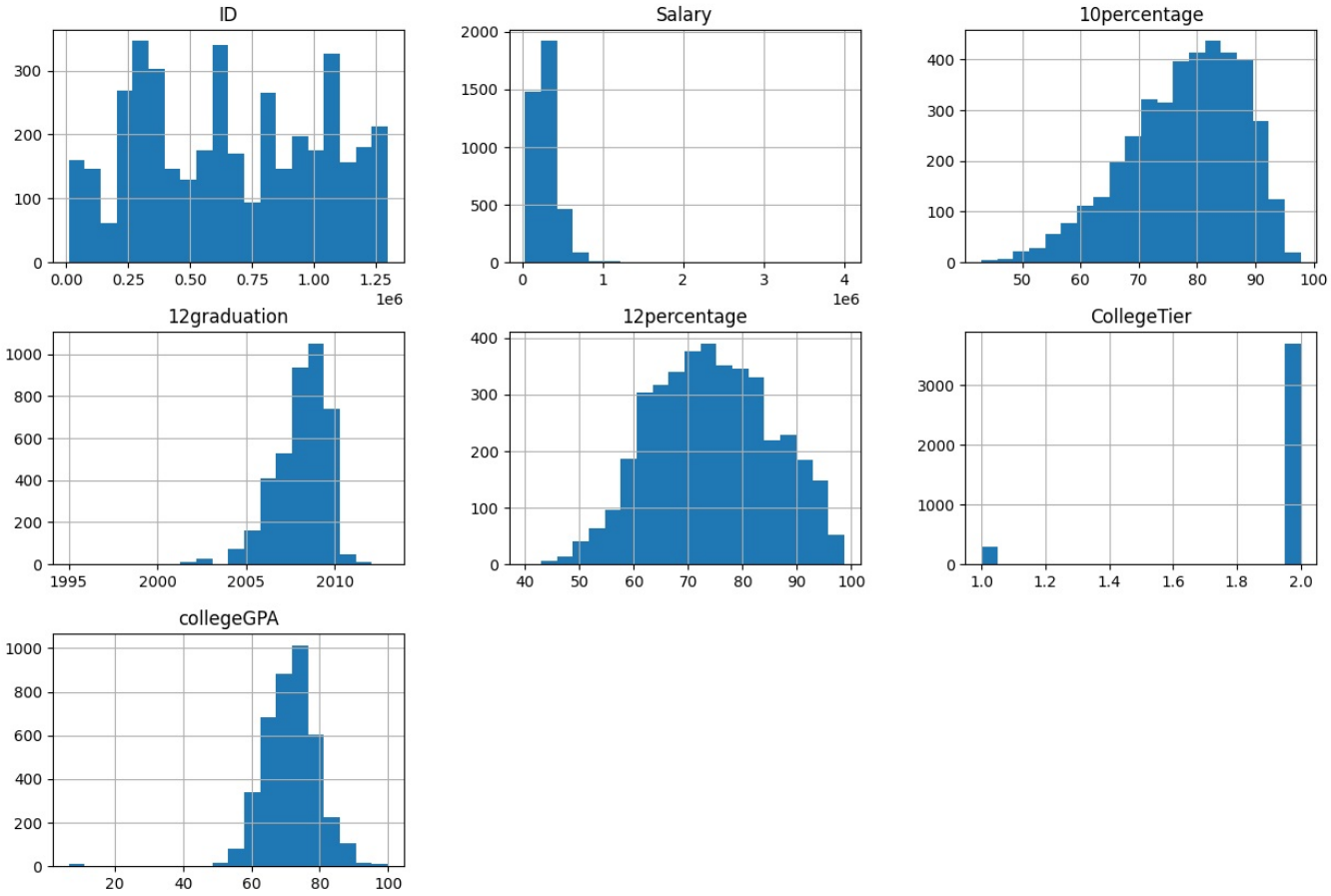
In [ ]:
```
#Categorical Columns
categorical_col= df.select_dtypes(include=['object']).columns
cat_df=df[categorical_col]
```

In [ ]:
```
cat_df.columns
```

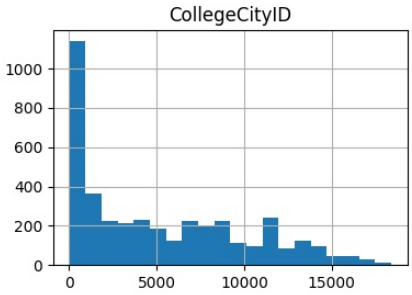
Out[ ]:
```
Index(['Unnamed: 0', 'DOL', 'Designation', 'JobCity', 'Gender', '10board',
       '12board', 'Degree', 'Specialization', 'CollegeState'],
      dtype='object')
```

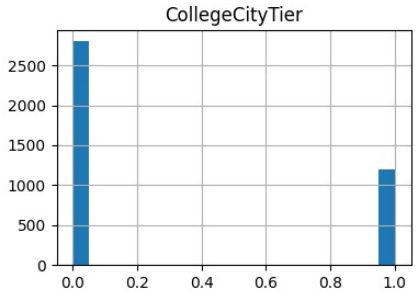
Univariate Analysis of Numerical Features

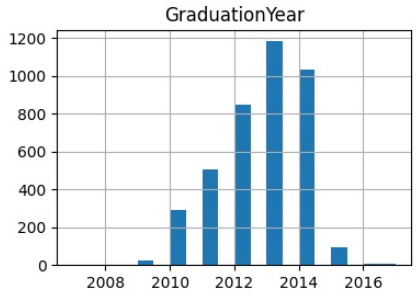
In [ ]:
```
# Histograms Plots
num_cols = ['ID', 'Salary' , '10percentage', '12graduation', '12percentage', 'CollegeTier', 'collegeGPA']
df[num_cols].hist(bins=20, figsize=(15, 10))
plt.show()
```

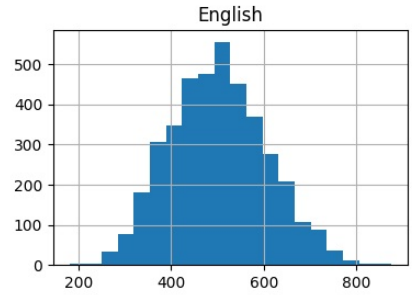


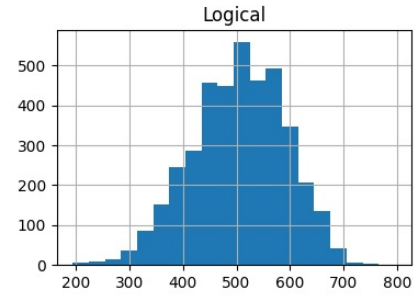
In [ ]:
```
num_cols = ['CollegeCityID', 'CollegeCityTier', 'GraduationYear', 'English','Logical', 'Quant', 'Domain']
df[num_cols].hist(bins=20, figsize=(15, 10))
plt.show()
```

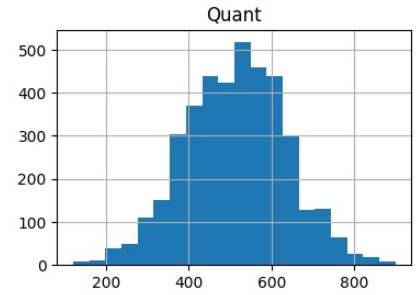


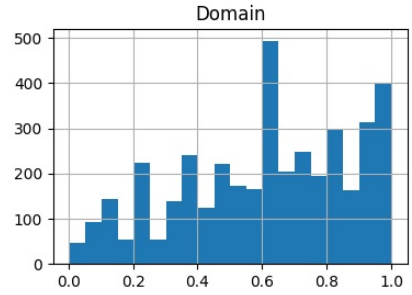




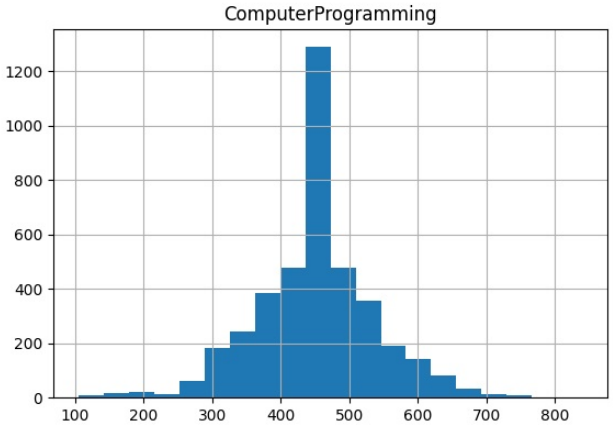


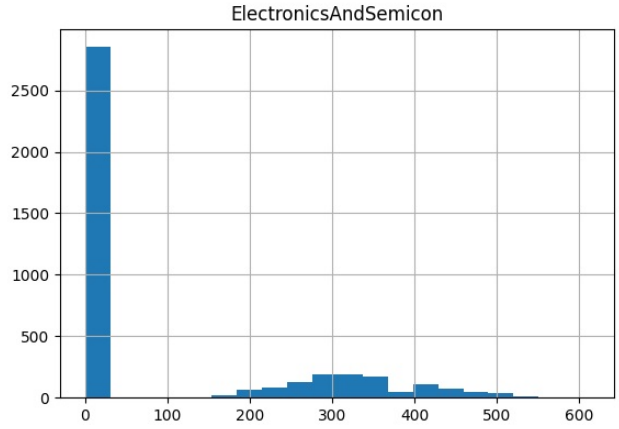


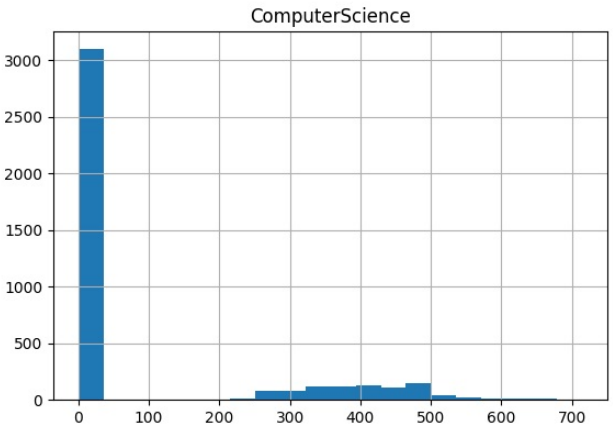




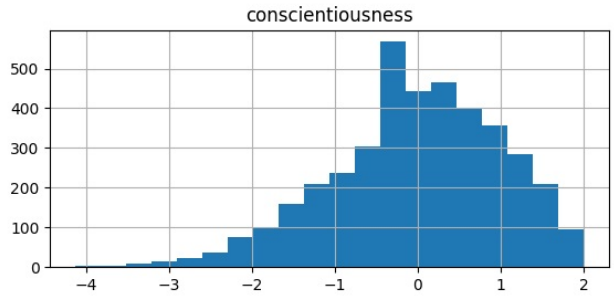
In [ ]:
```
num_cols = ['ComputerProgramming','ElectronicsAndSemicon', 'ComputerScience']
df[num_cols].hist(bins=20, figsize=(15, 10))
plt.show()
```

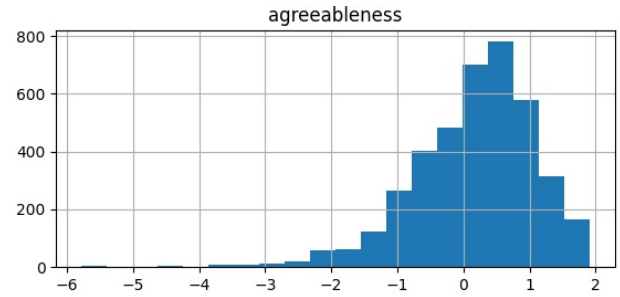


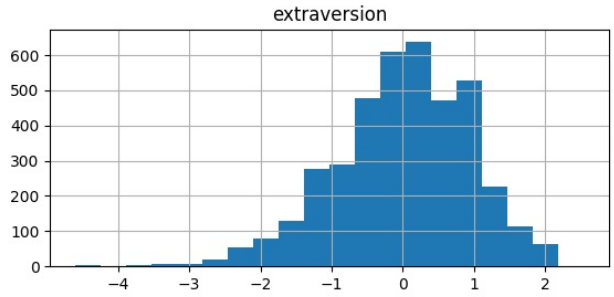


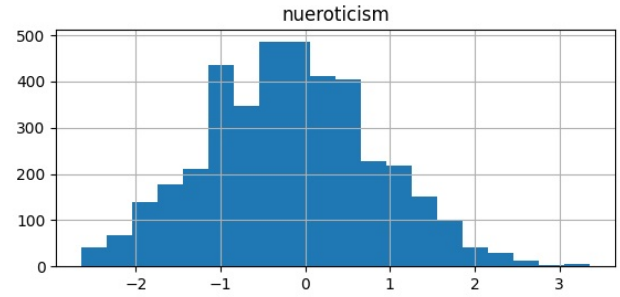


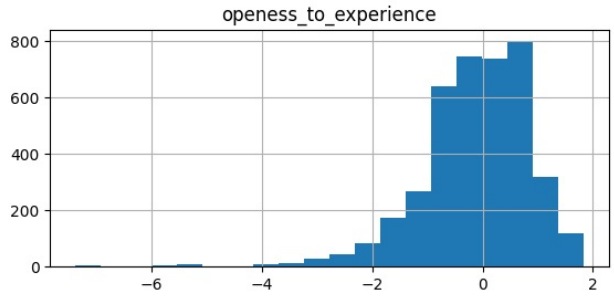
In [ ]:
```
num_cols = ['conscientiousness','agreeableness', 'extraversion', 'nueroticism','openess_to_experience', 'YearsE
df[num_cols].hist(bins=20, figsize=(15, 10))
plt.show()
```

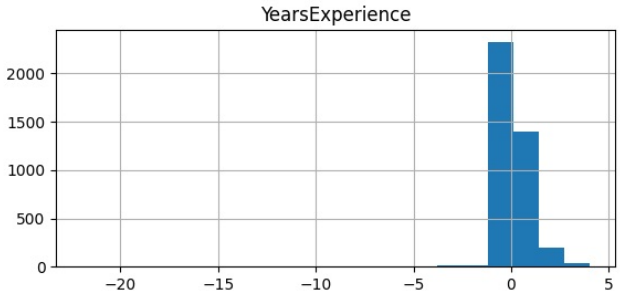




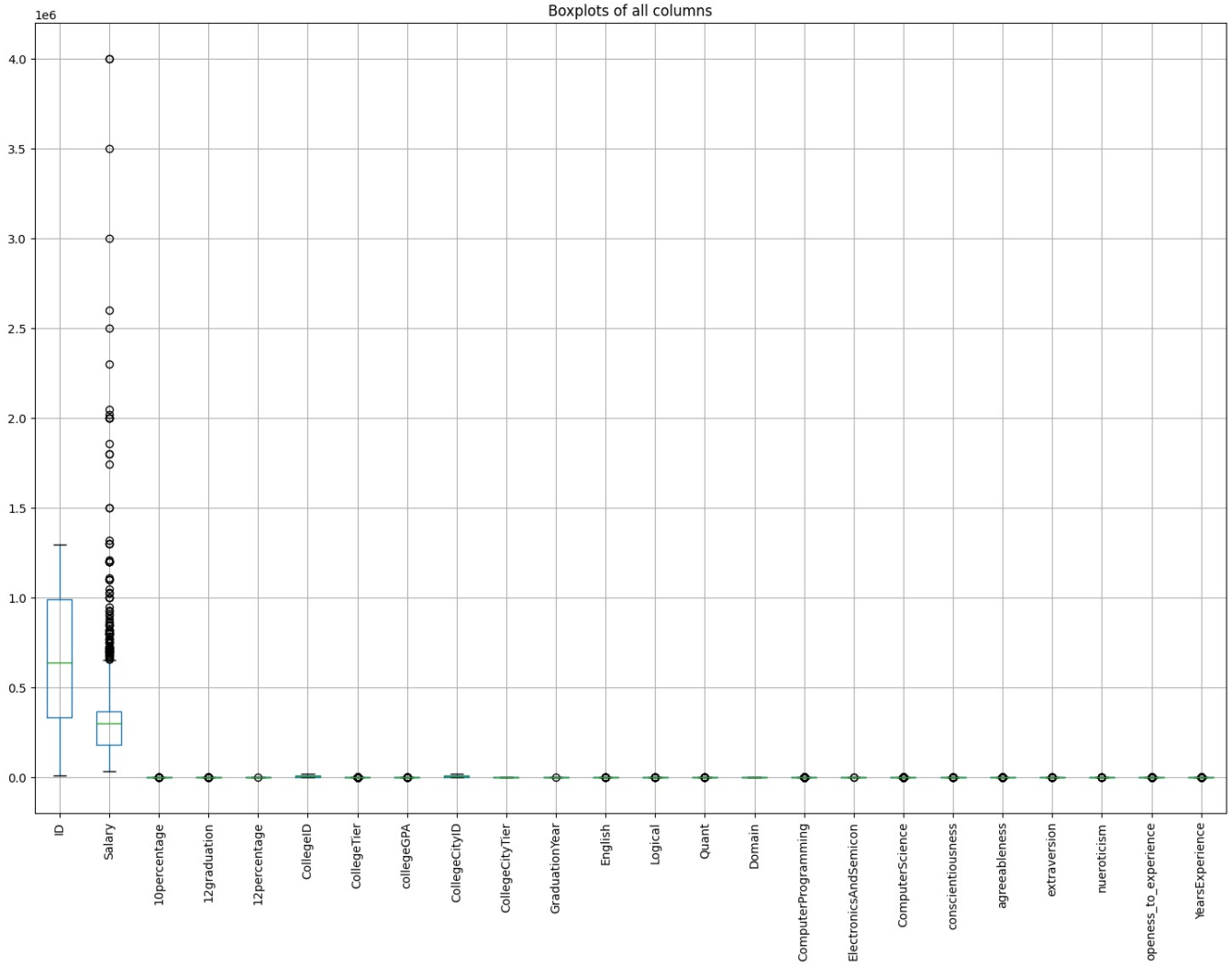






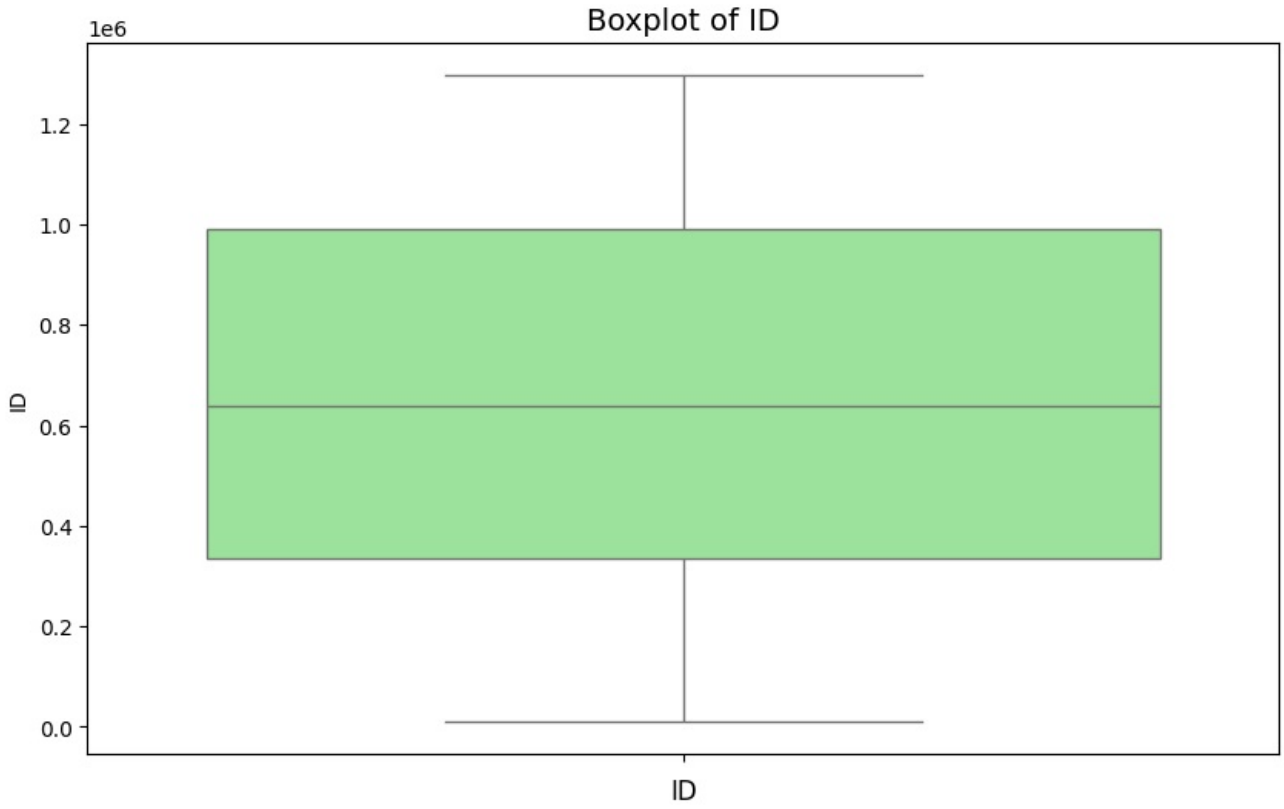


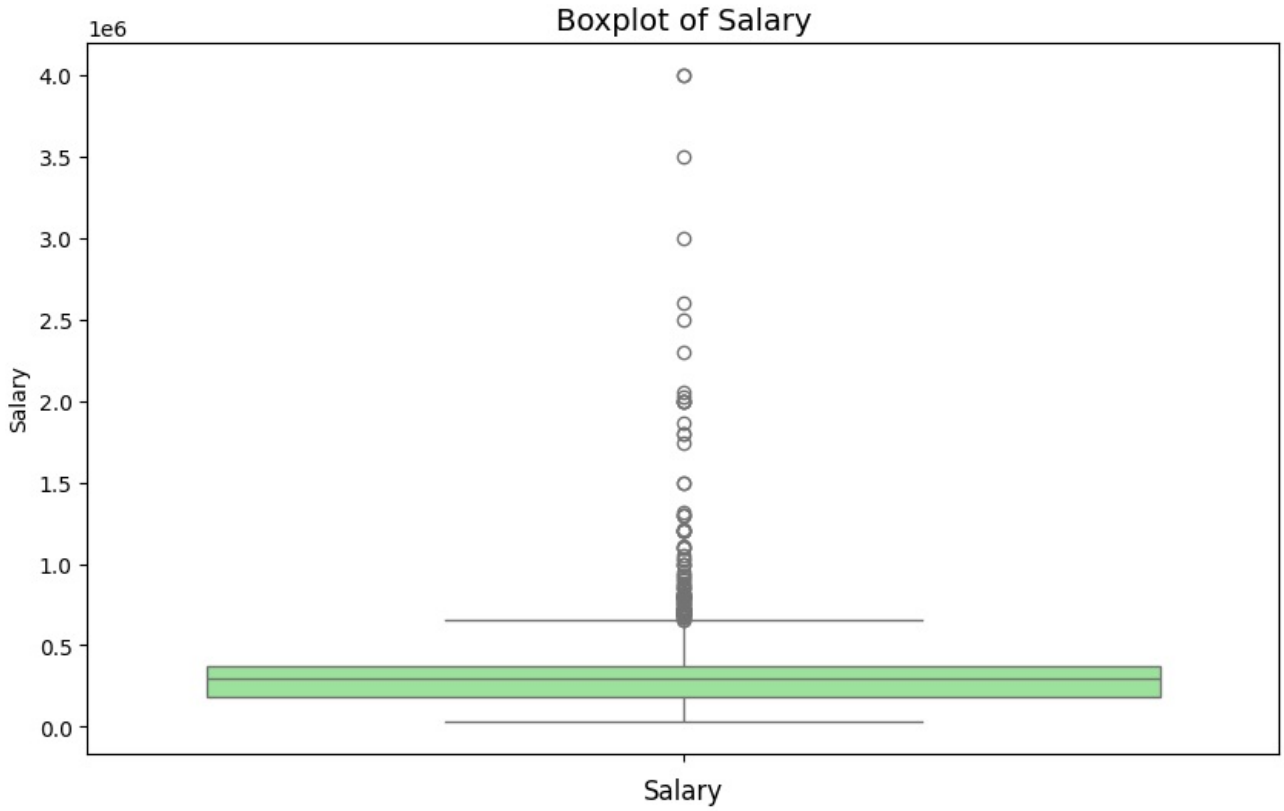
In [ ]:
```
#Boxplots of all columns
plt.figure(figsize=(18, 12))
df.boxplot(rot=90)
plt.title("Boxplots of all columns")
plt.show()
```

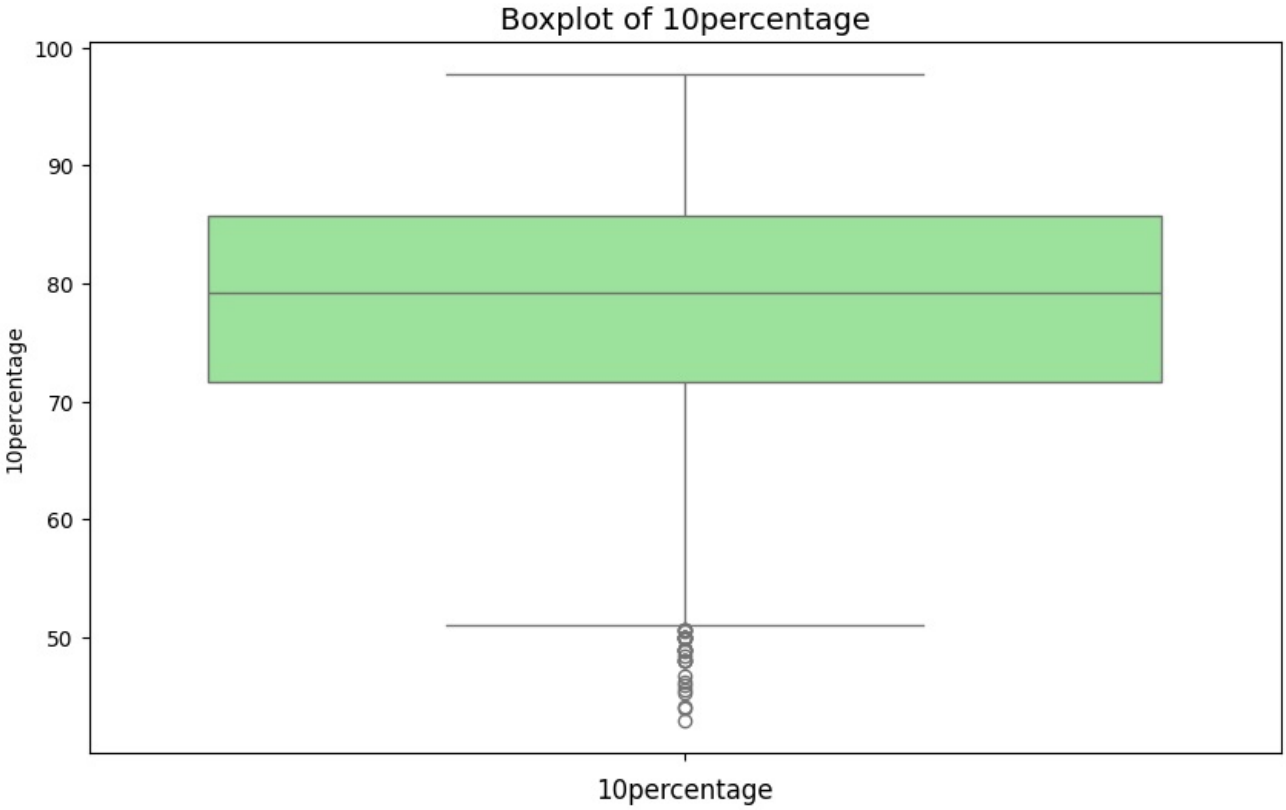


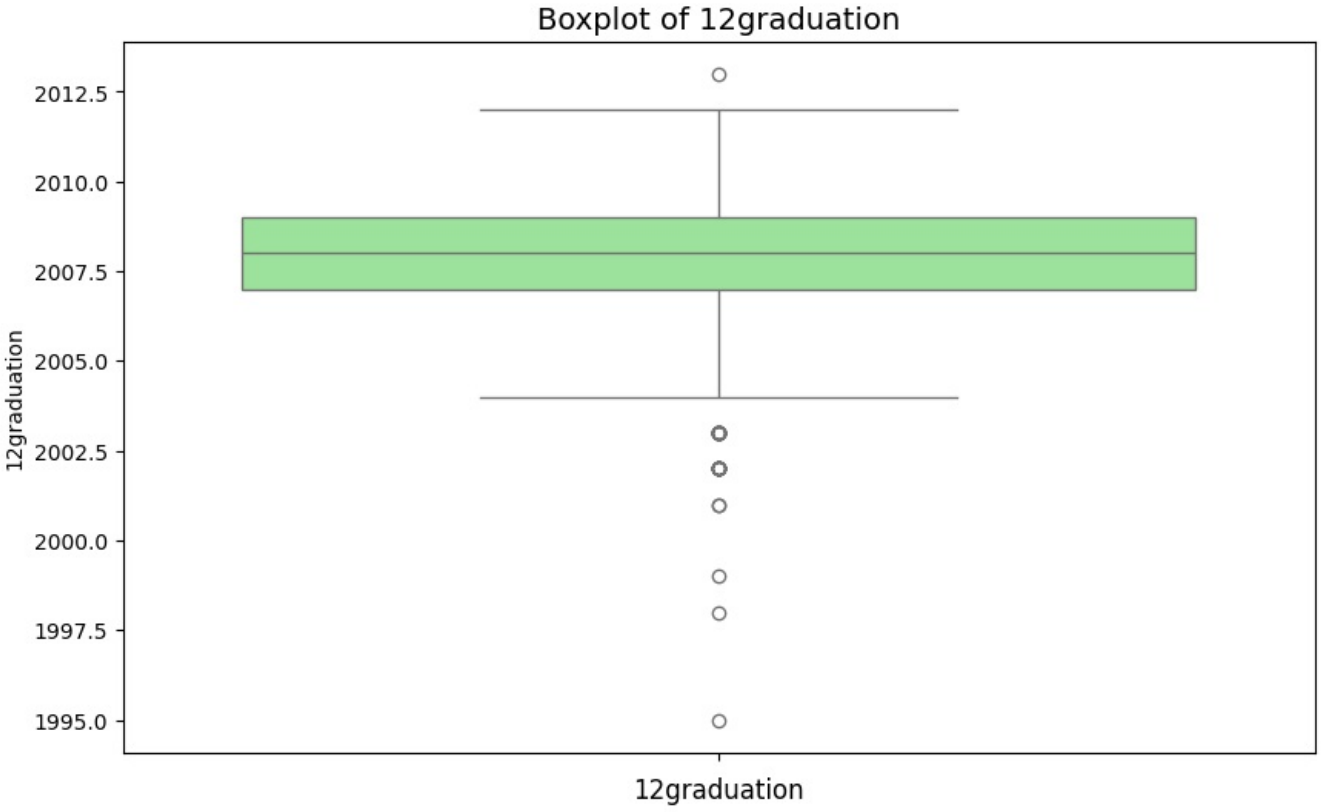
In [ ]:
```
for column in numerical_col:
```

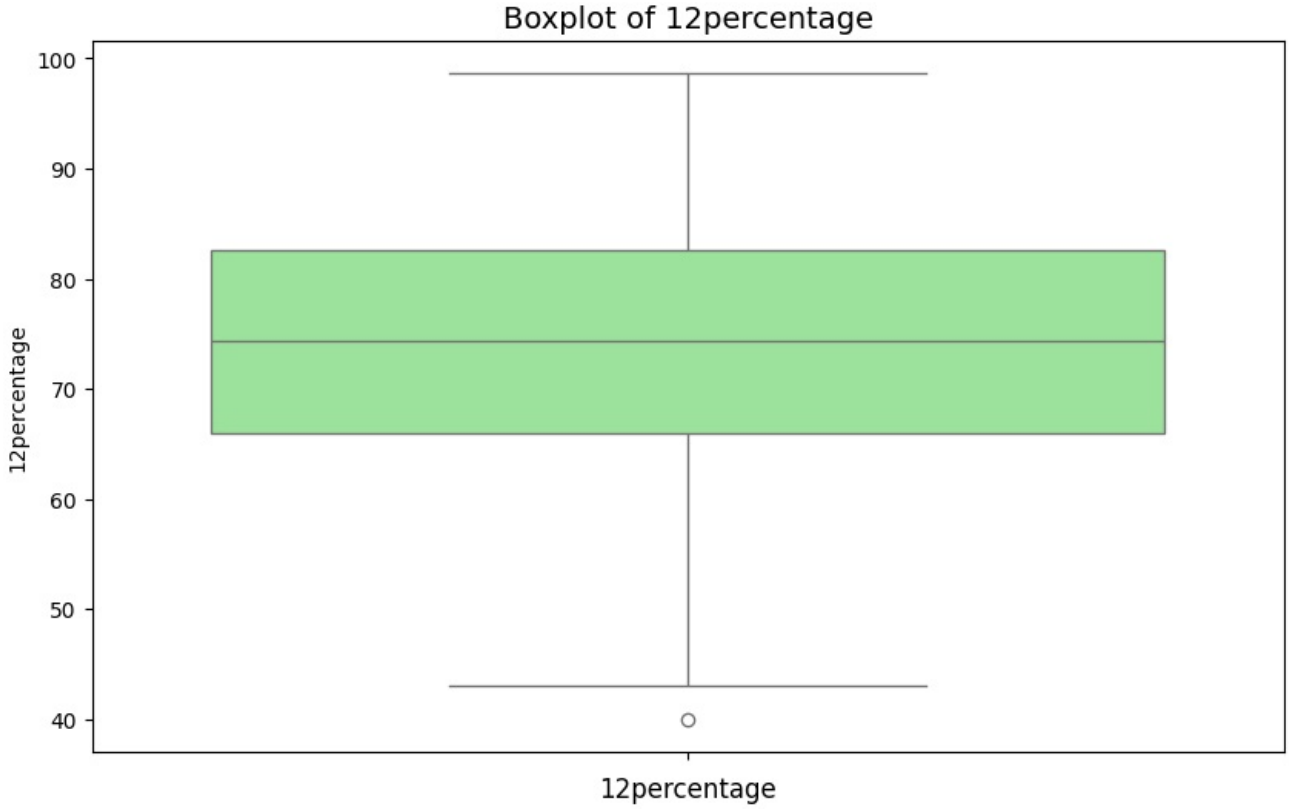
```
for column in numerical_col:
    plt.figure(figsize=(10, 6))
    sns.boxplot(df[column], color='Lightgreen')
    plt.title(f"Boxplot of {column}", fontsize=14)
    plt.xlabel(column, fontsize=12)
    plt.xticks(fontsize=10)
    plt.show()
```

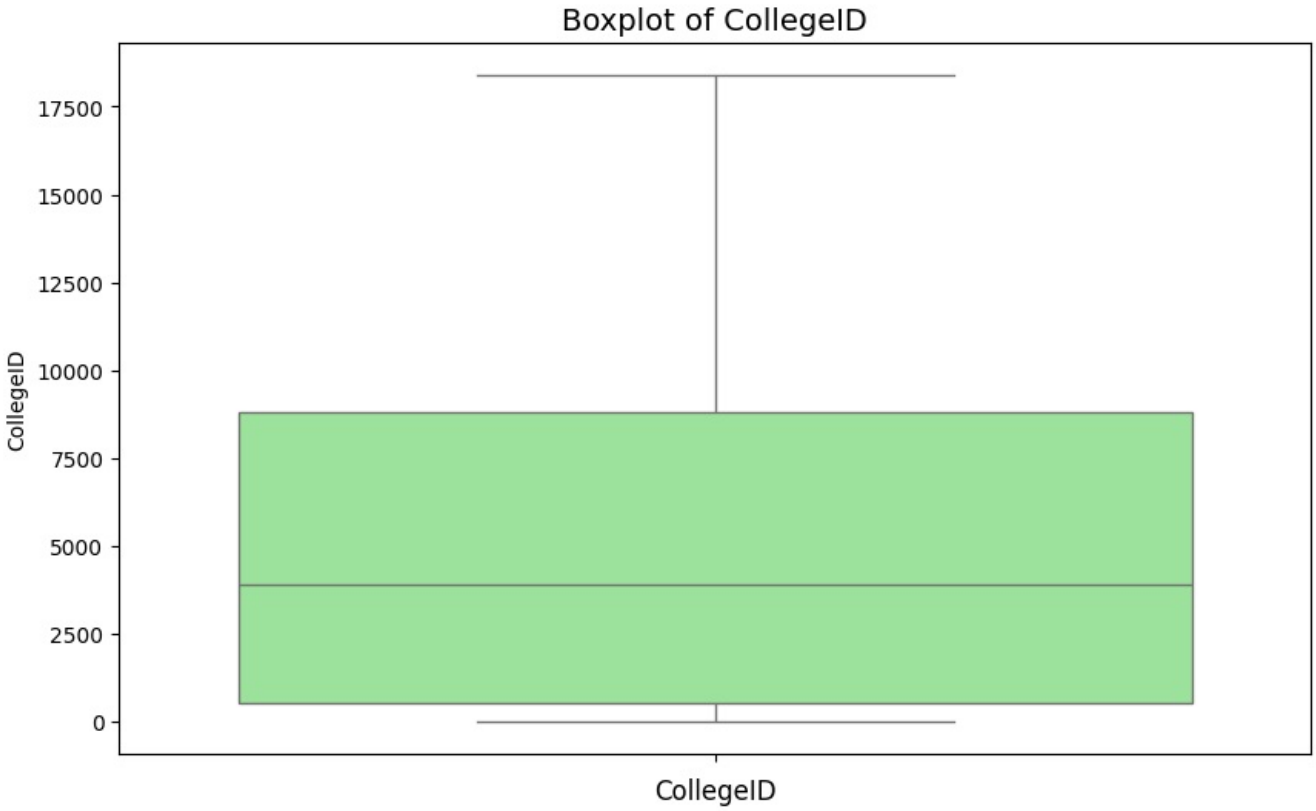


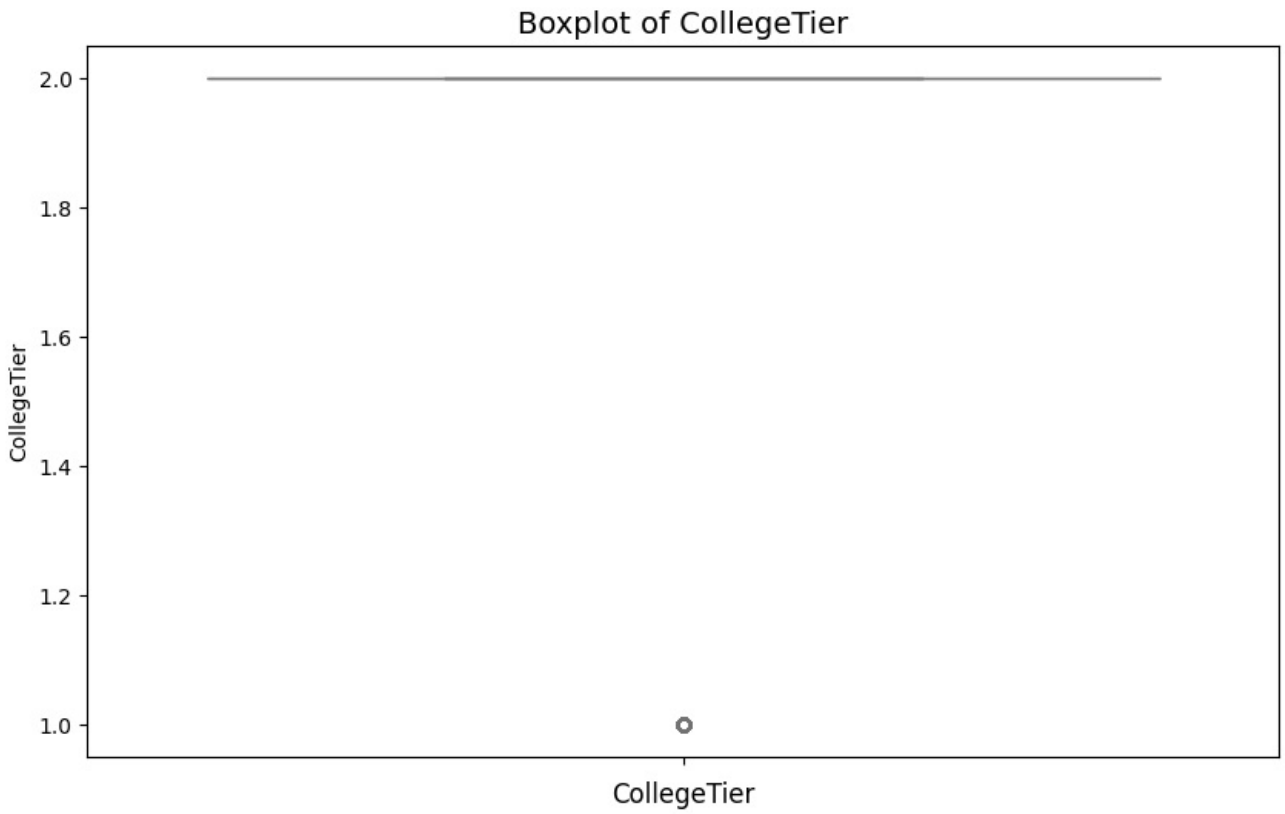


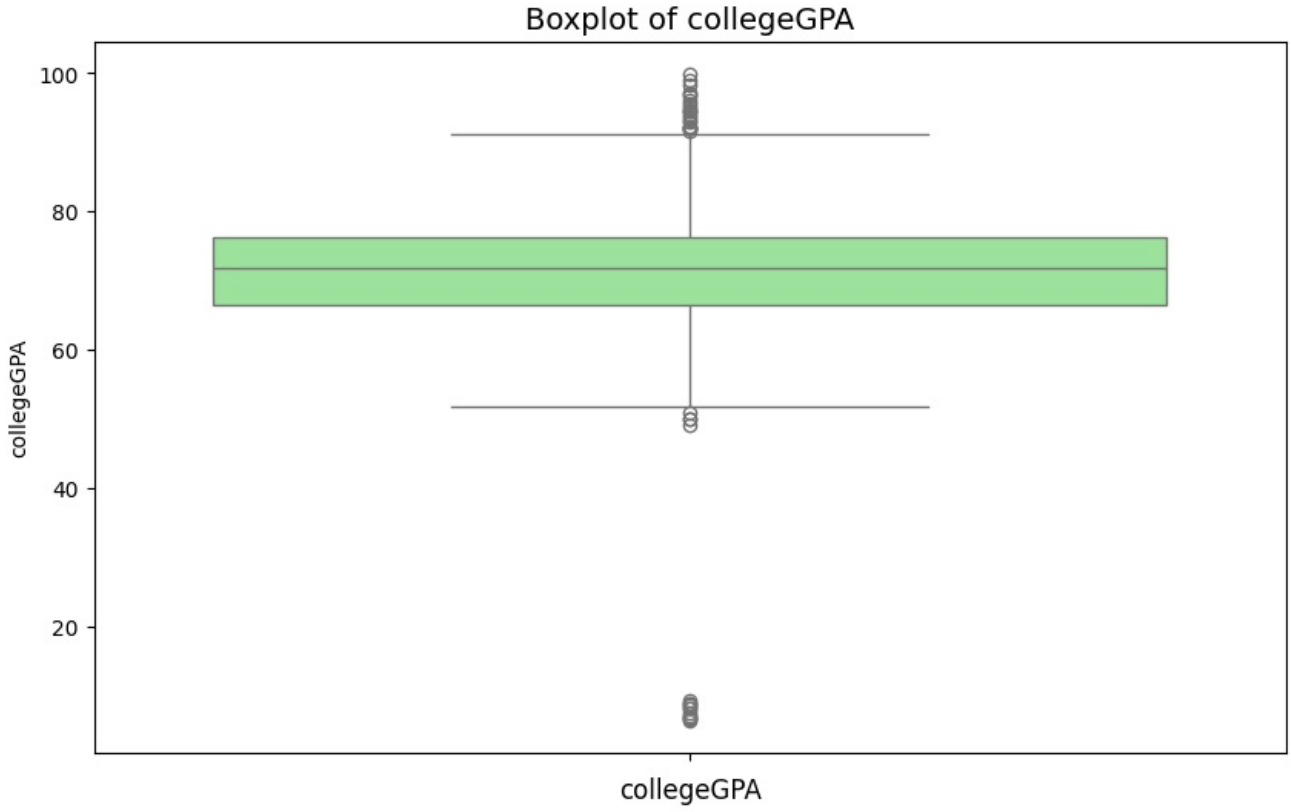

Boxplot of 10percentage
100
90
80
70
60
50
10percentage
10percentage

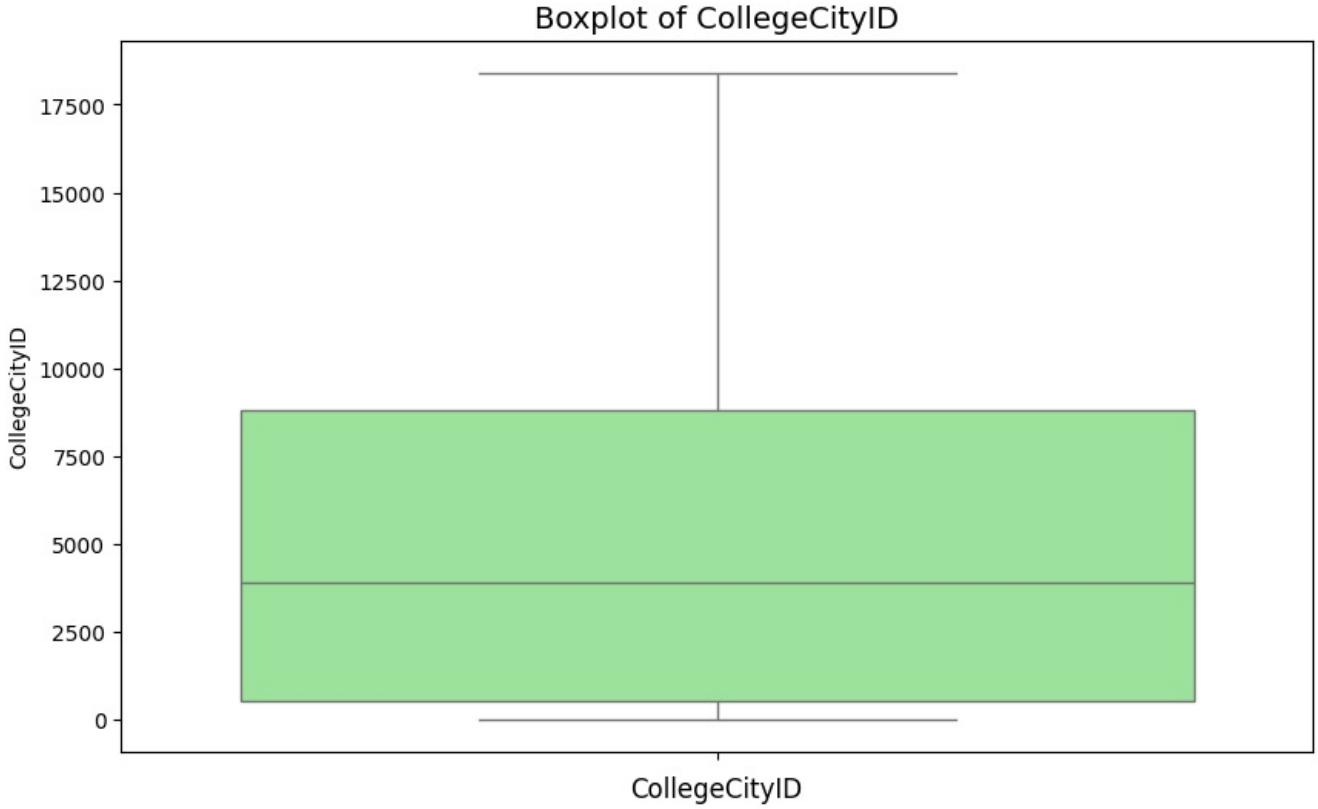

Boxplot of 12graduation
2012.5
2010.0
2007.5
2005.0
2002.5
2000.0
1997.5
1995.0
12graduation
12graduation

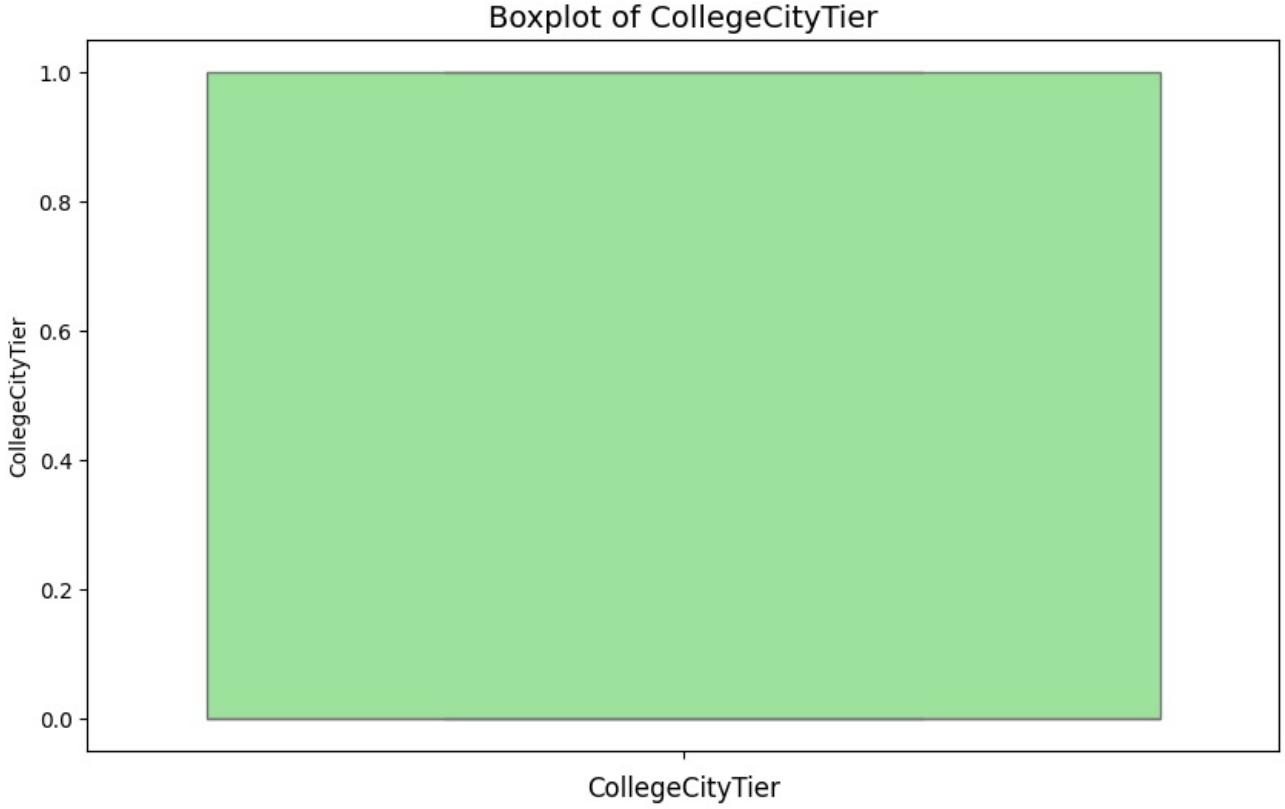

Boxplot of 12percentage
100
90
80
70
60
50
40
12percentage
12percentage

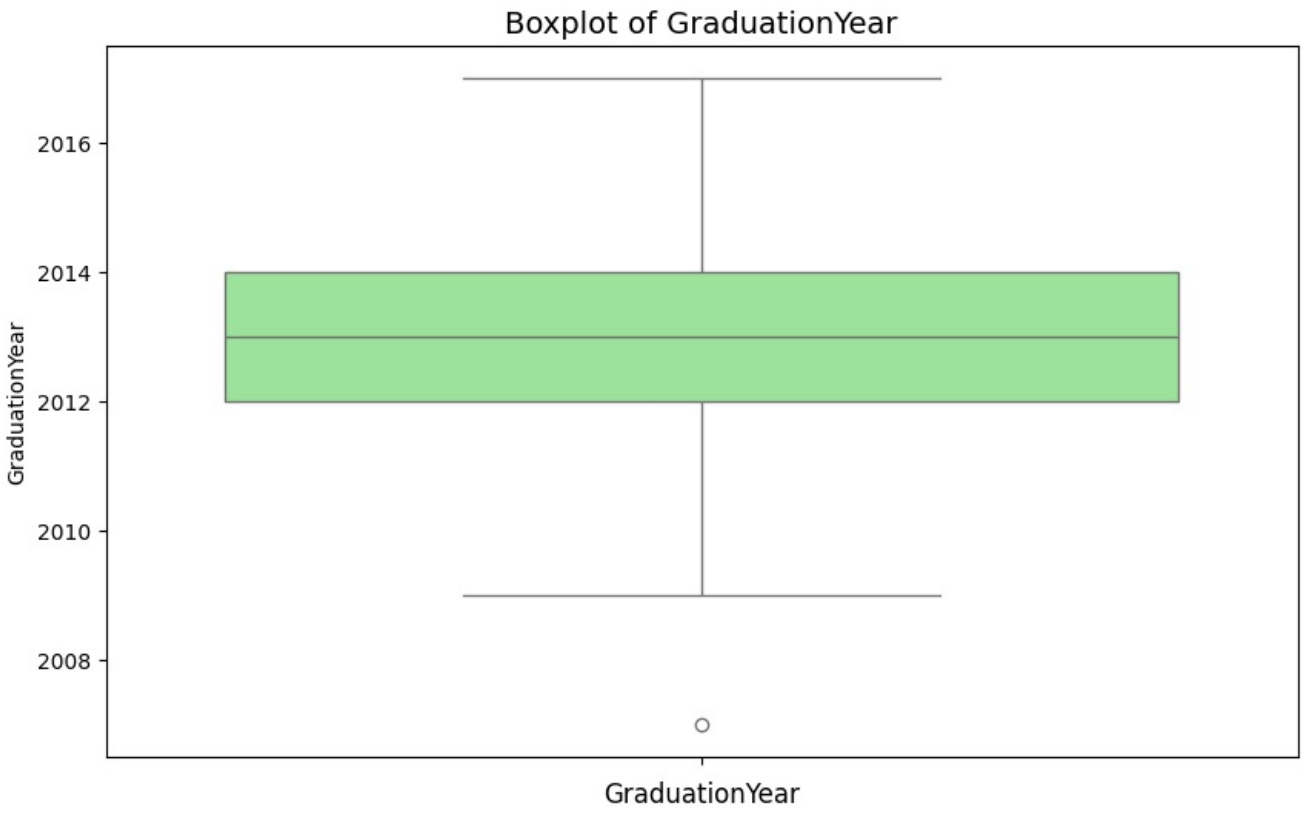

Boxplot of CollegeID
17500
15000
12500
10000
7500
5000
2500
0
CollegeID
CollegeID

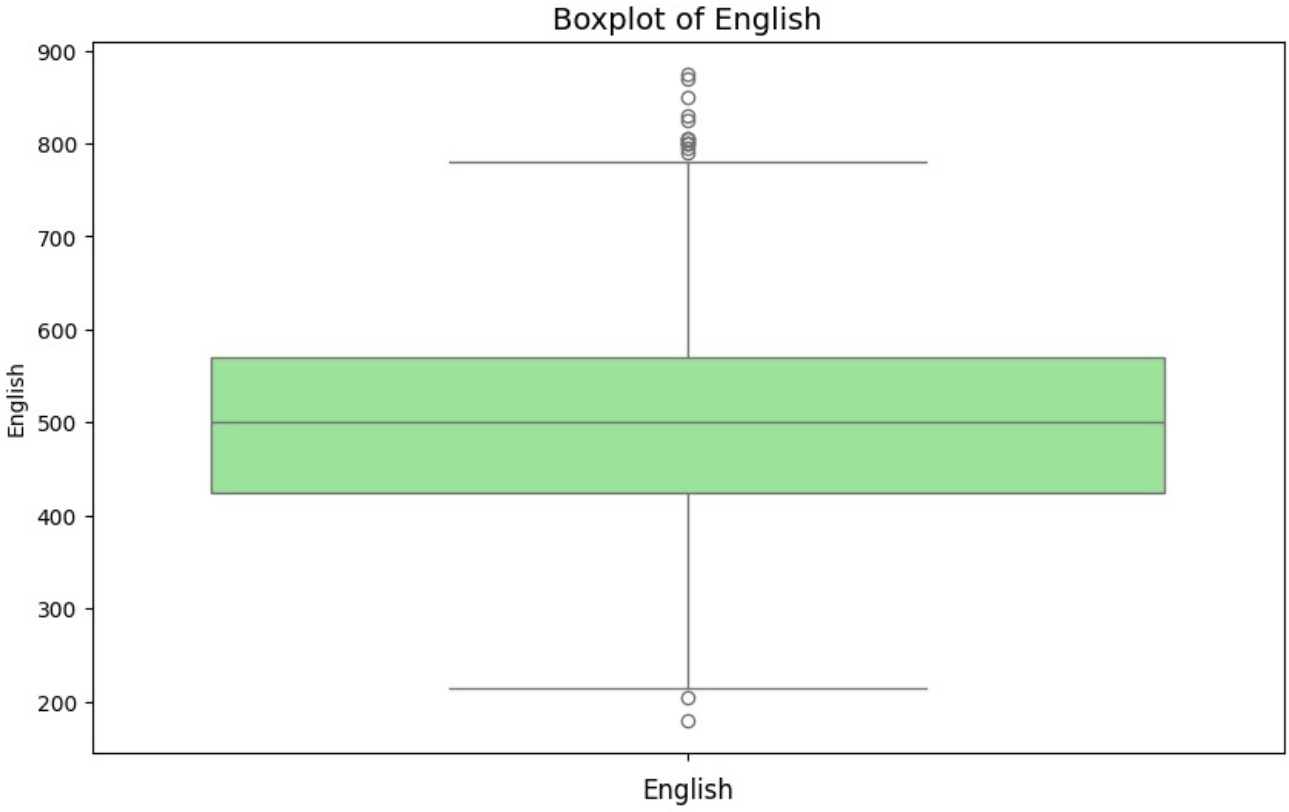

Boxplot of CollegeTier
2.0
1.8
1.6
1.4
1.2
1.0
CollegeTier
CollegeTier

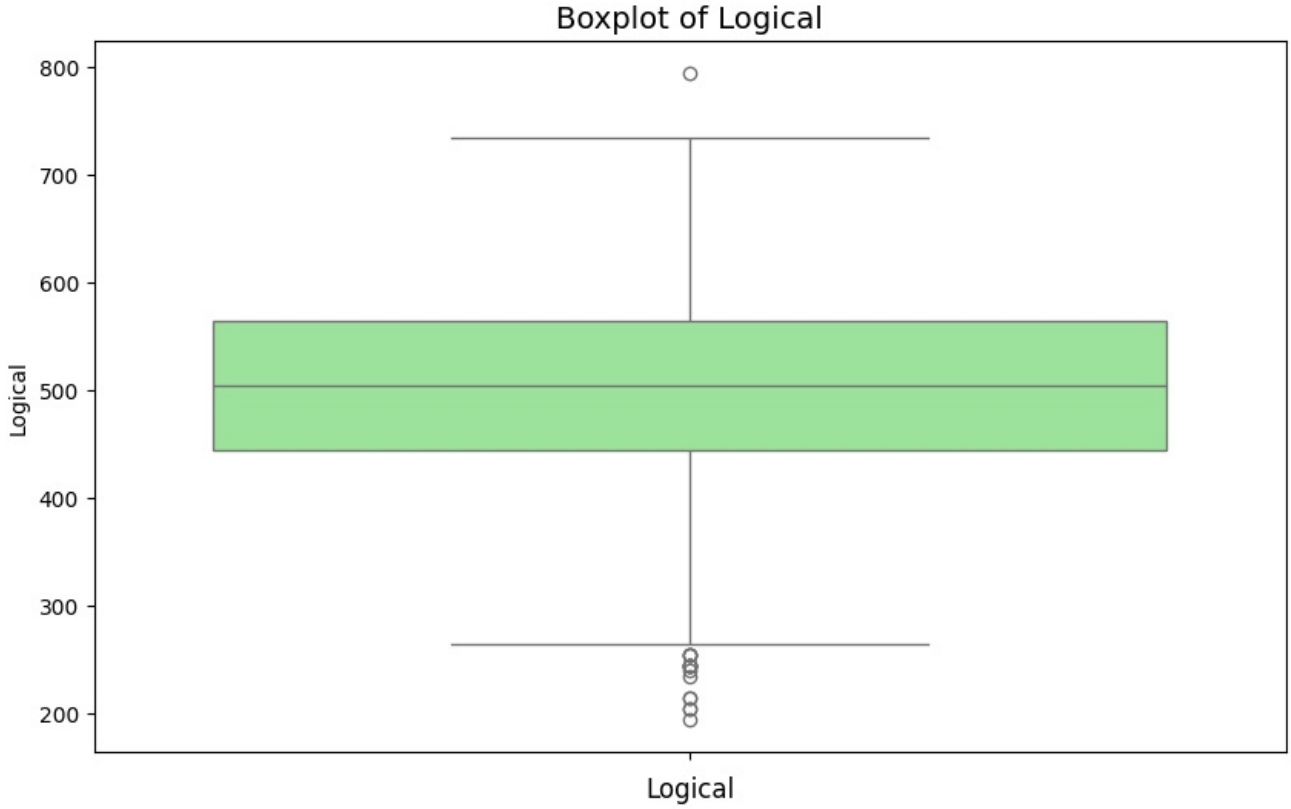

Boxplot of collegeGPA
100
80
60
40
20
collegeGPA
collegeGPA

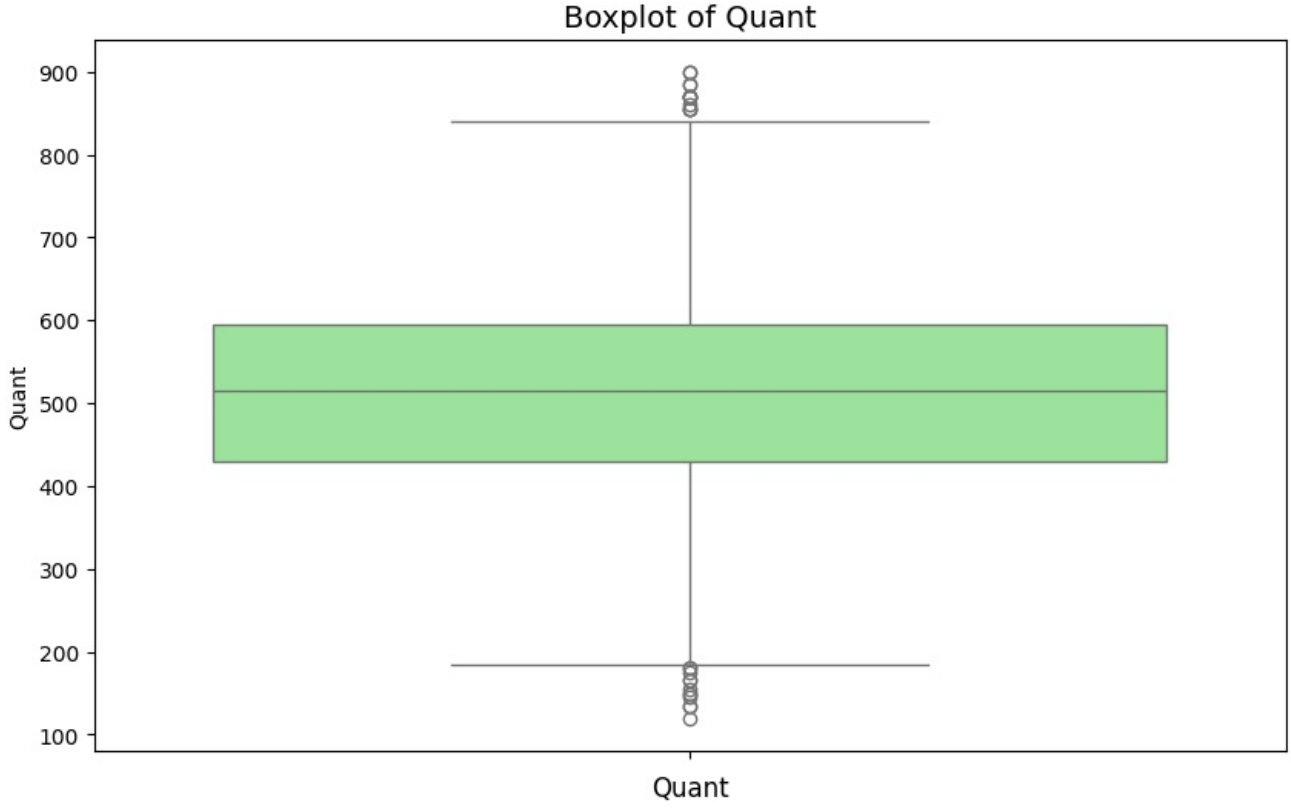

Boxplot of CollegeCityID
17500
15000
12500
10000
7500
5000
2500
0
CollegeCityID
CollegeCityID

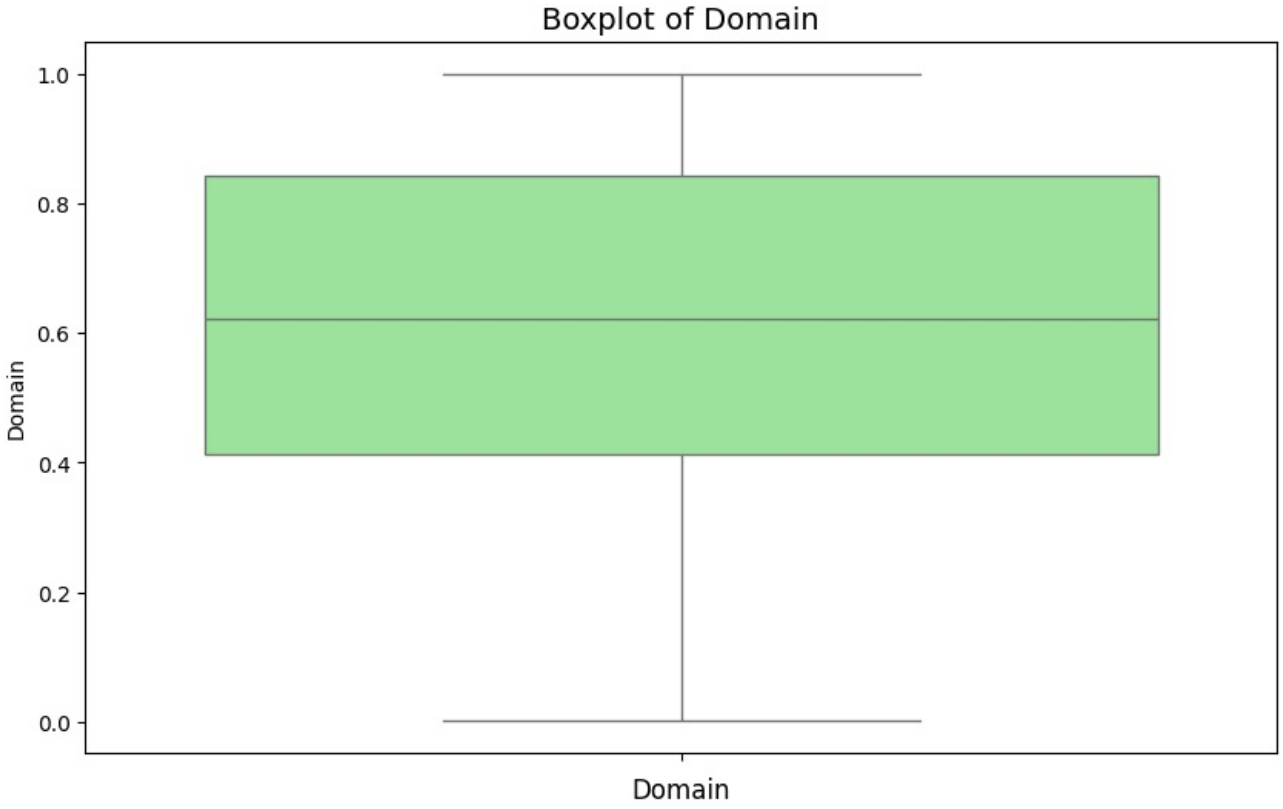

Boxplot of CollegeCityTier
1.0
0.8
0.6
0.4
0.2
0.0
CollegeCityTier
CollegeCityTier

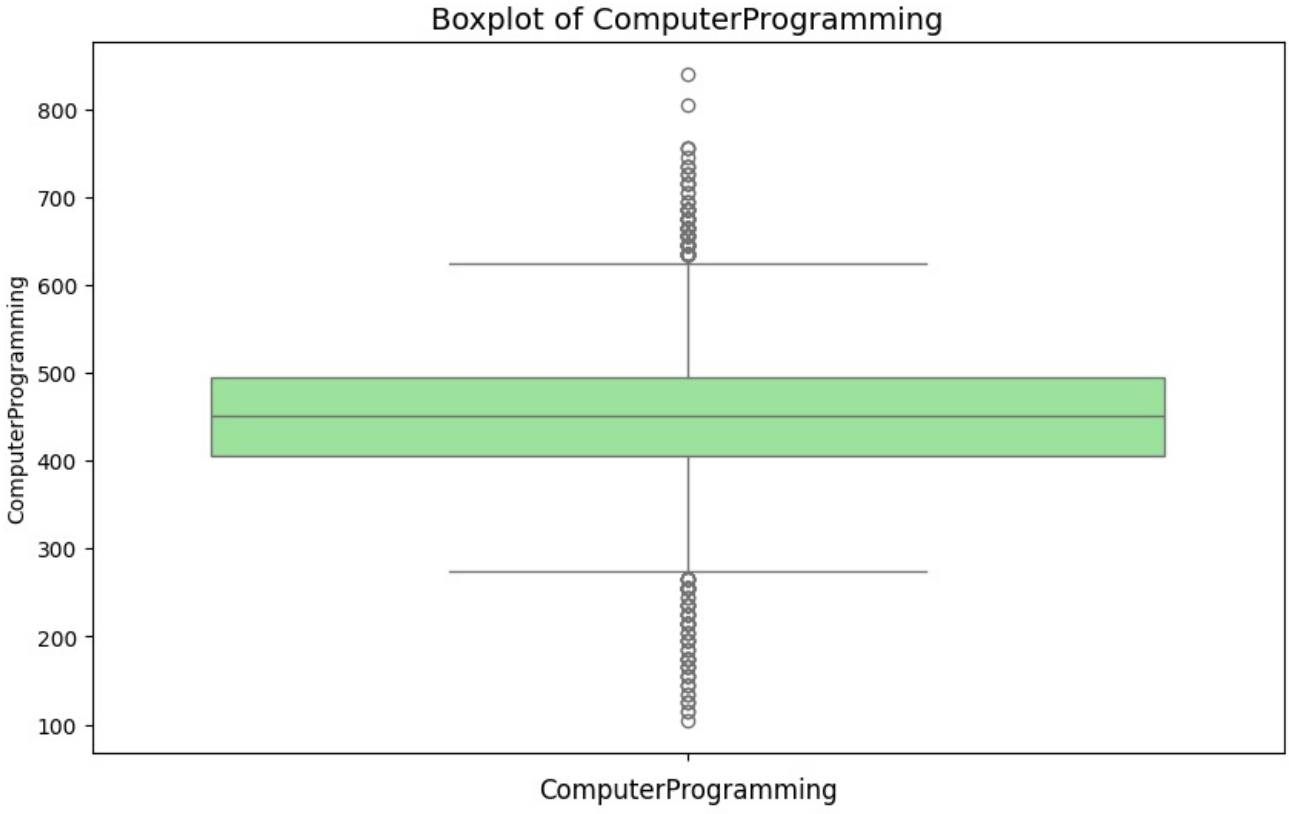

Boxplot of GraduationYear
2016
2014
2012
2010
2008
GraduationYear
GraduationYear

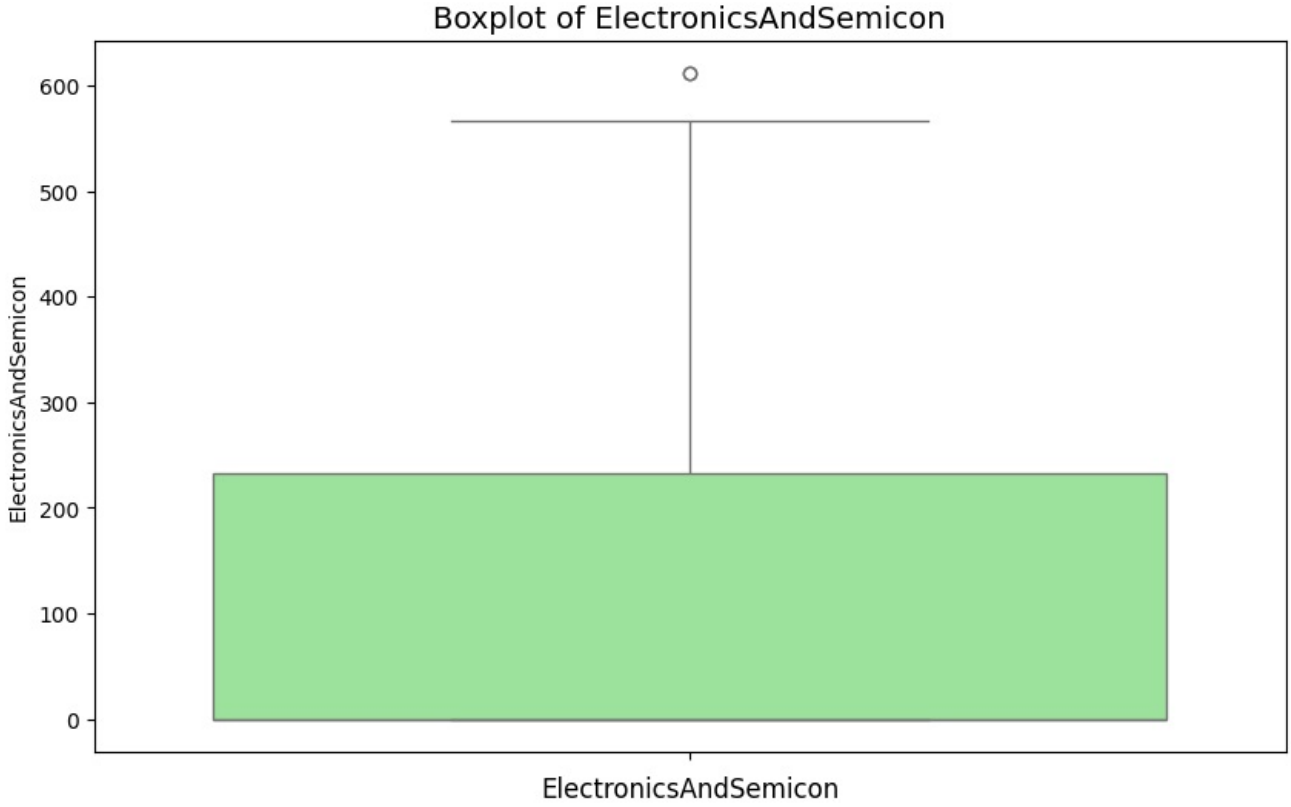

Boxplot of English
900
800
700
600
500
400
300
200
English
English

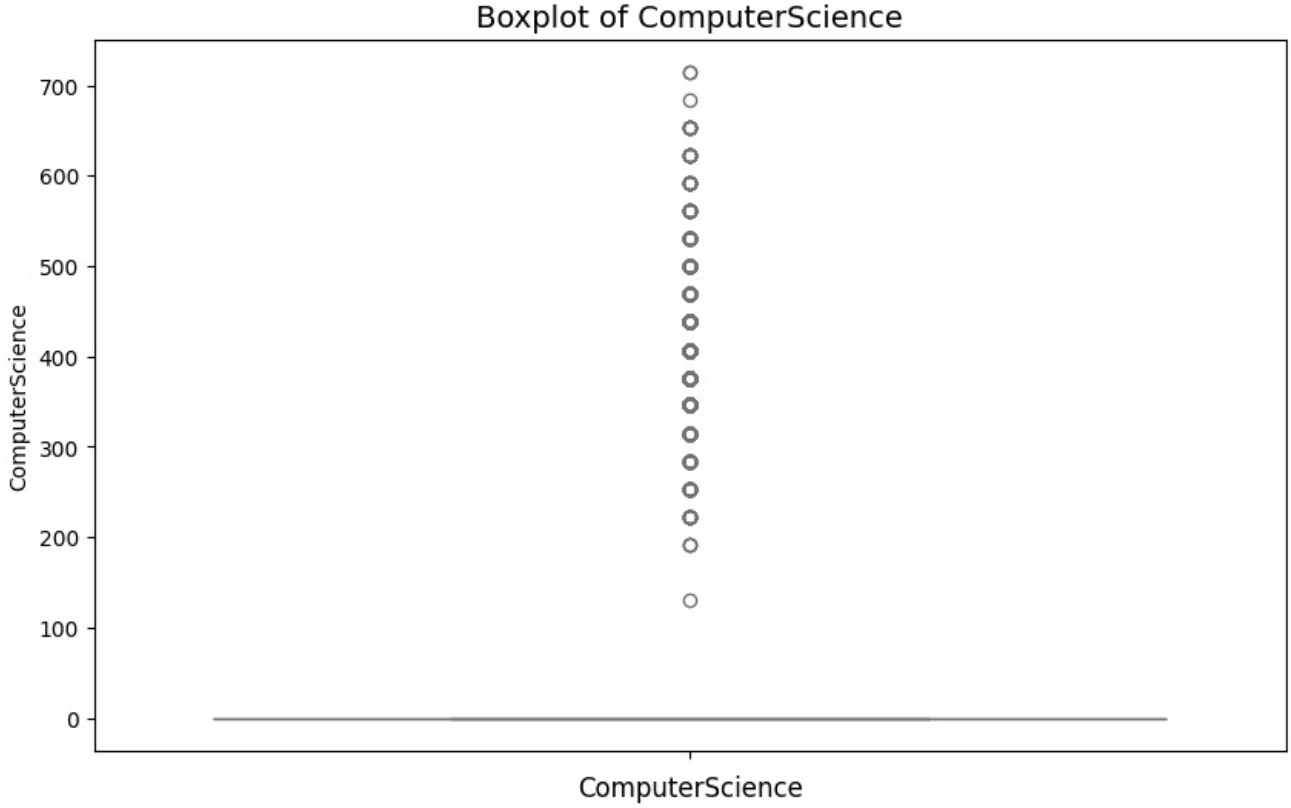

Boxplot of Logical
800
700
600
500
400
300
200
Logical
Logical

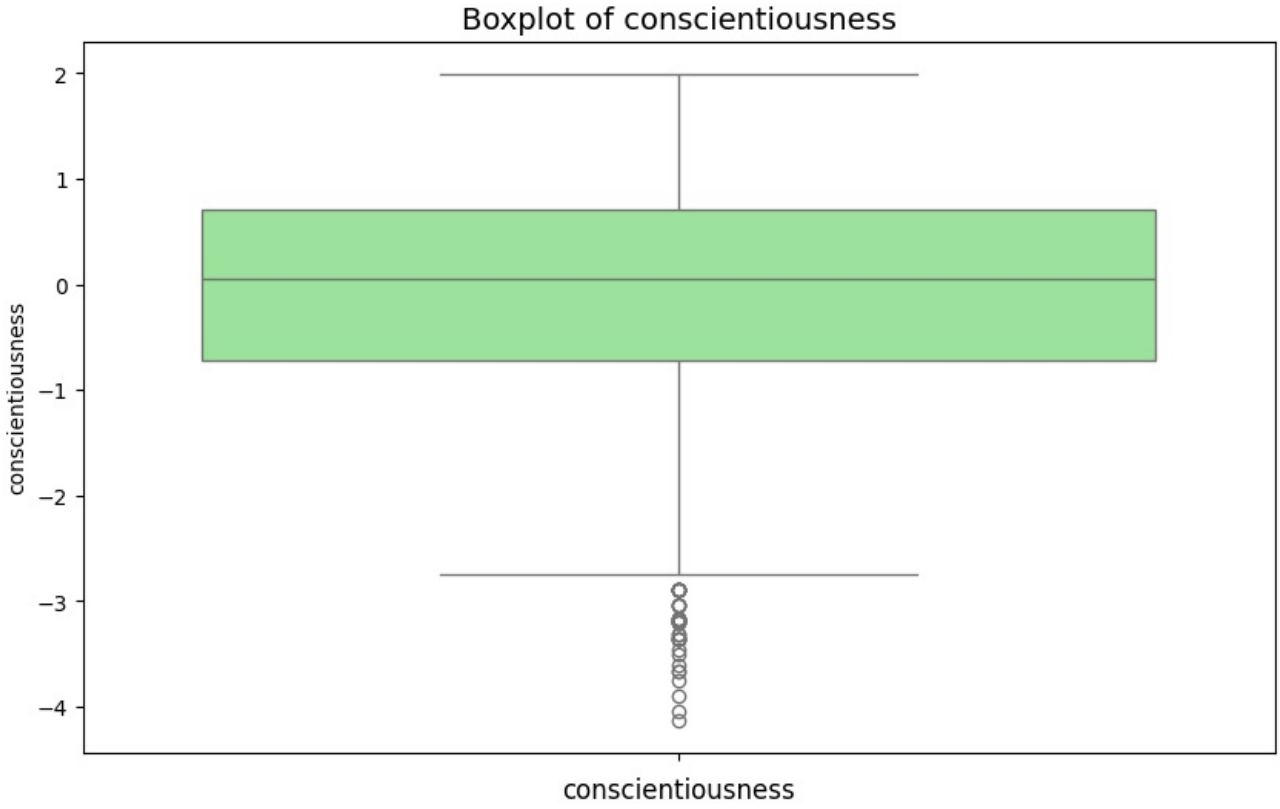

Boxplot of Quant
900
800
700
600
500
400
300
200
100
Quant
Quant

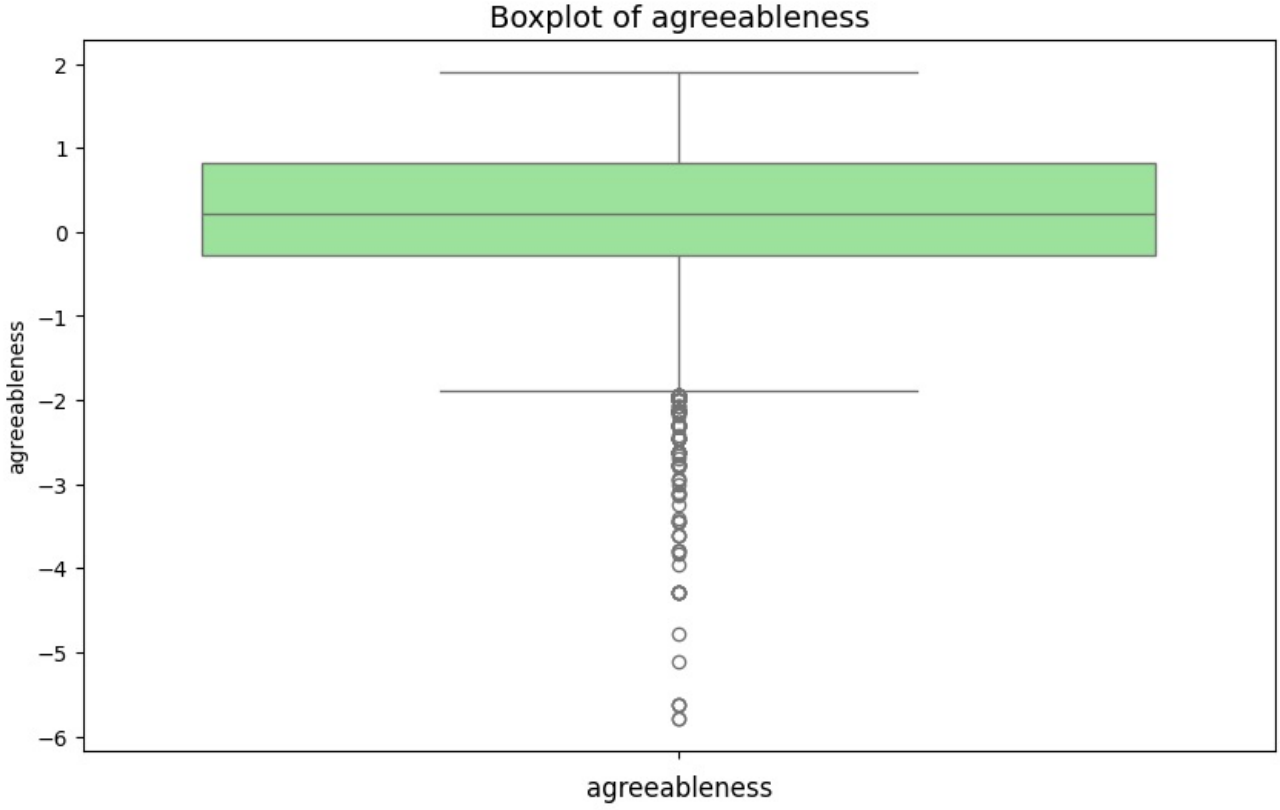

Boxplot of Domain
1.0
0.8
0.6
0.4
0.2
0.0
Domain
Domain

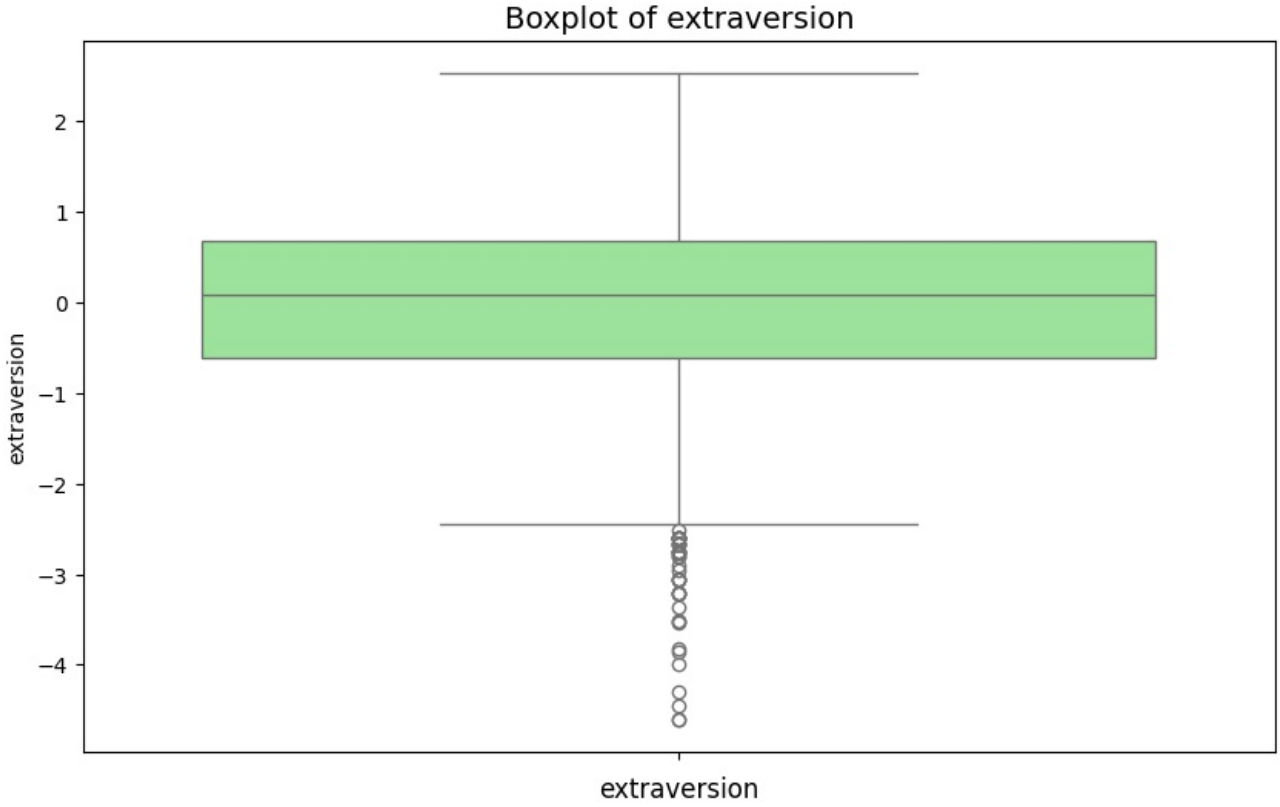

Boxplot of ComputerProgramming
800
700
600
500
400
300
200
100
ComputerProgramming
ComputerProgramming

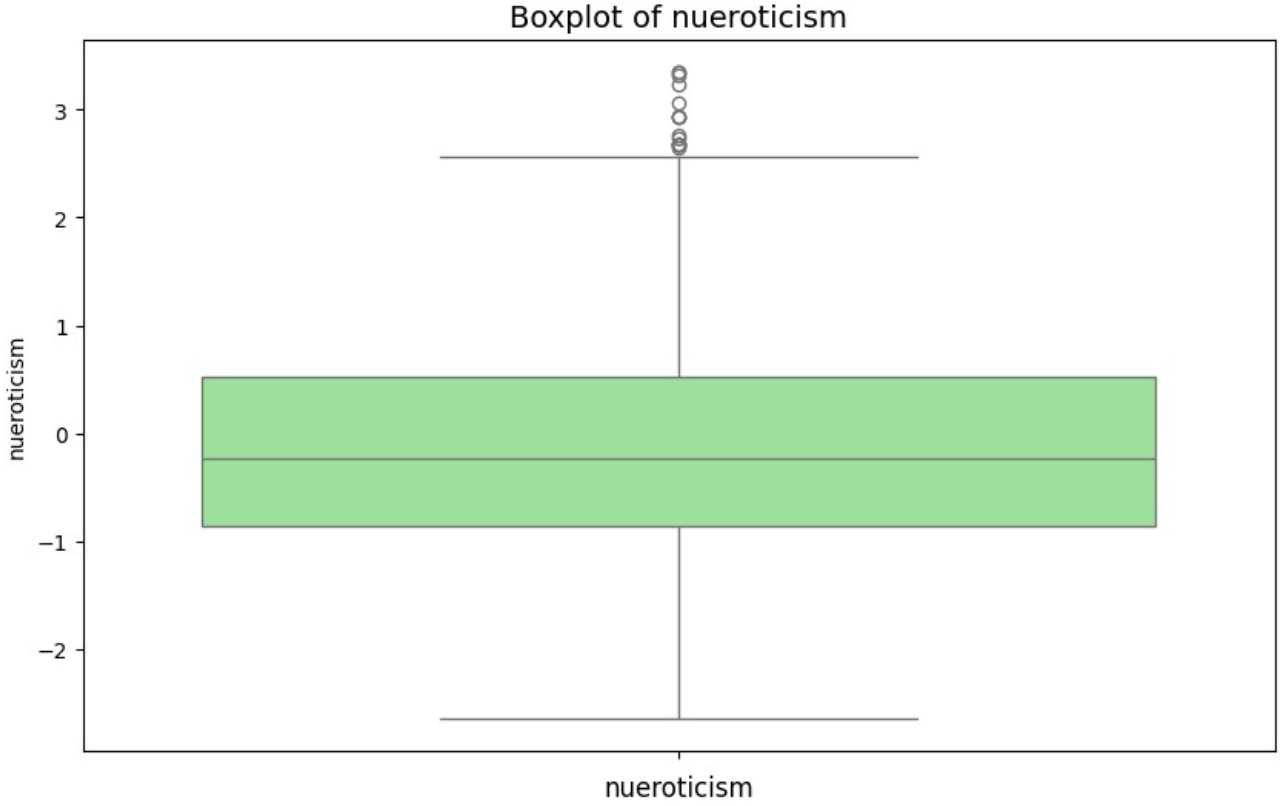

Boxplot of ElectronicsAndSemicon
ElectronicsAndSemicon
0
100
200
300
400
500
600
ElectronicsAndSemicon

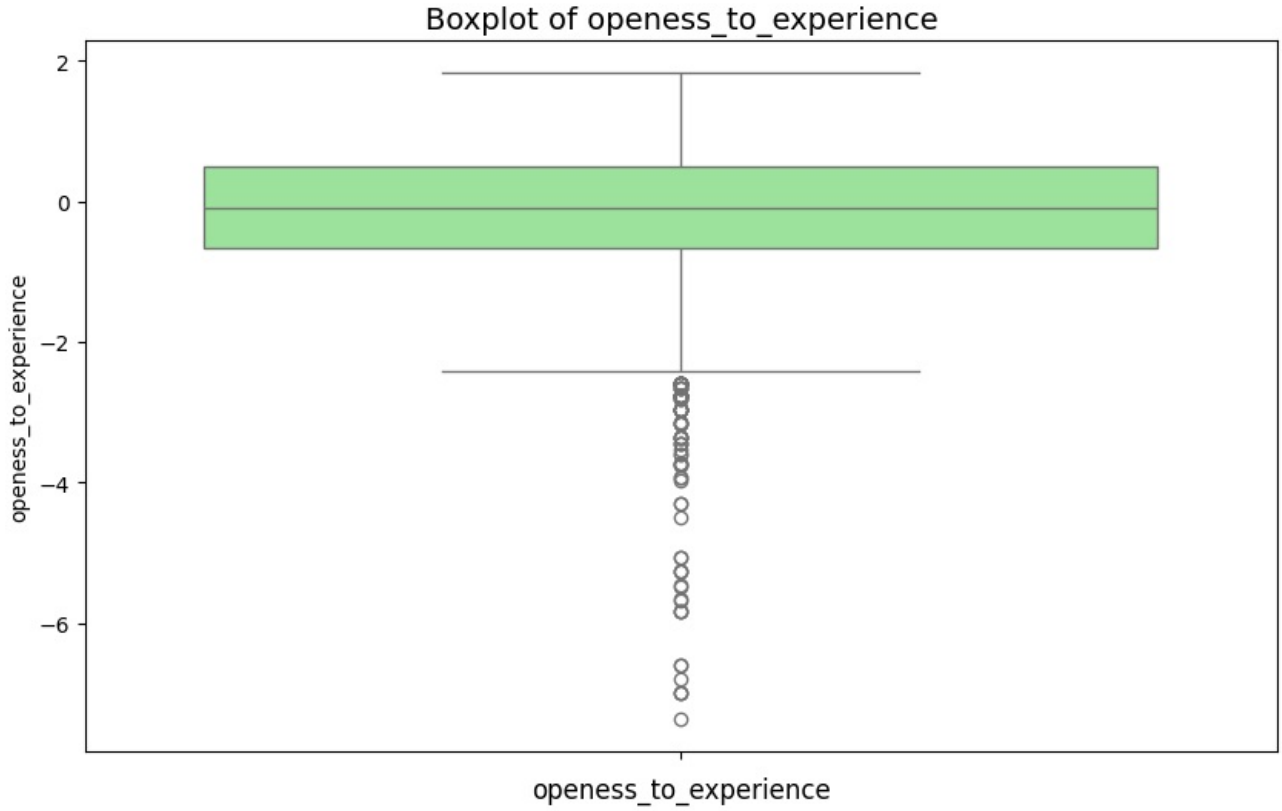

Boxplot of ComputerScience
ComputerScience
0
100
200
300
400
500
600
700
ComputerScience


Boxplot of conscientiousness
conscientiousness
2
1
0
−1
−2
−3
−4
conscientiousness


Boxplot of agreeableness
agreeableness
2
1
0
−1
−2
−3
−4
−5
−6
agreeableness


Boxplot of extraversion
extraversion
2
1
0
−1
−2
−3
−4
extraversion


Boxplot of nueroticism
nueroticism
3
2
1
0
−1
−2
nueroticism



In [ ]:
```
#Removing outliers (IQR)

def outlier_treatment(df, col):
    sorted_df = df[col].sort_values()
    Q1, Q3 = np.percentile(sorted_df, [25, 75])
    IQR = Q3 - Q1
```

```
    lower_range = Q1 - (1.5 * IQR)
    upper_range = Q3 + (1.5 * IQR)
    return lower_range, upper_range

columns = ['Salary', '10percentage', '12percentage', 'English',
           'Logical', 'Quant', 'Domain', 'ComputerProgramming',
           'ElectronicsAndSemicon', 'ComputerScience', 'conscientiousness',
           'agreeableness', 'extraversion', 'nueroticism', 'openess_to_experience',
           'YearsExperience']

df_copy = df.copy()

for col in columns:
    # Extracting upper fence and lower fence
    lowerbound, upperbound = outlier_treatment(df_copy, col)

    # Dropping the outliers
    df_copy = df_copy.drop(df_copy[(df_copy[col] < lowerbound) | (df_copy[col] > upperbound)].index)

print(f'Number of observations with outliers : {df.shape[0]}')
print(f'Number of observations without outliers : {df_copy.shape[0]}')
```

```
Number of observations with outliers : 3998
Number of observations without outliers : 2578
```

In [ ]:
```
df_copy.head()
```

Out[ ]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	Domain	ComputerProgramming	E
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	...	0.635979	445.000000	
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	...	0.960603	451.301278	
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	...	0.450877	395.000000	
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	...	0.124502	451.301278	
9	train	1203363	230000	2014-07-01	present	project engineer	Kolkata	m	1993-06-13	77.0	...	0.493596	385.000000	

5 rows × 36 columns

In [ ]:
```
df_copy.shape
```

Out[ ]:
```
(2578, 36)
```

In [ ]:
```
df_copy.describe()
```

Out[ ]:

	ID	Salary	10percentage	12graduation	12percentage	CollegeID	CollegeTier	collegeGPA	CollegeCityID	Colleg
count	2.578000e+03	2578.000000	2578.000000	2578.000000	2578.000000	2578.000000	2578.000000	2578.000000	2578.000000	25
mean	5.783818e+05	298126.066718	78.515206	2007.906905	75.080287	4878.375873	1.935997	71.651490	4878.375873	
std	3.370632e+05	127951.166392	9.403211	1.605392	10.967787	4649.937321	0.244806	8.070226	4649.937321	
min	1.124400e+04	35000.000000	50.600000	1998.000000	43.120000	11.000000	1.000000	6.450000	11.000000	
25%	3.146578e+05	200000.000000	72.152500	2007.000000	67.000000	462.000000	2.000000	66.530000	462.000000	
50%	5.375660e+05	300000.000000	79.800000	2008.000000	75.000000	3649.000000	2.000000	72.000000	3649.000000	
75%	8.141958e+05	385000.000000	86.000000	2009.000000	83.000000	8346.000000	2.000000	76.600000	8346.000000	
max	1.297877e+06	655000.000000	97.120000	2012.000000	98.200000	18409.000000	2.000000	99.930000	18409.000000	

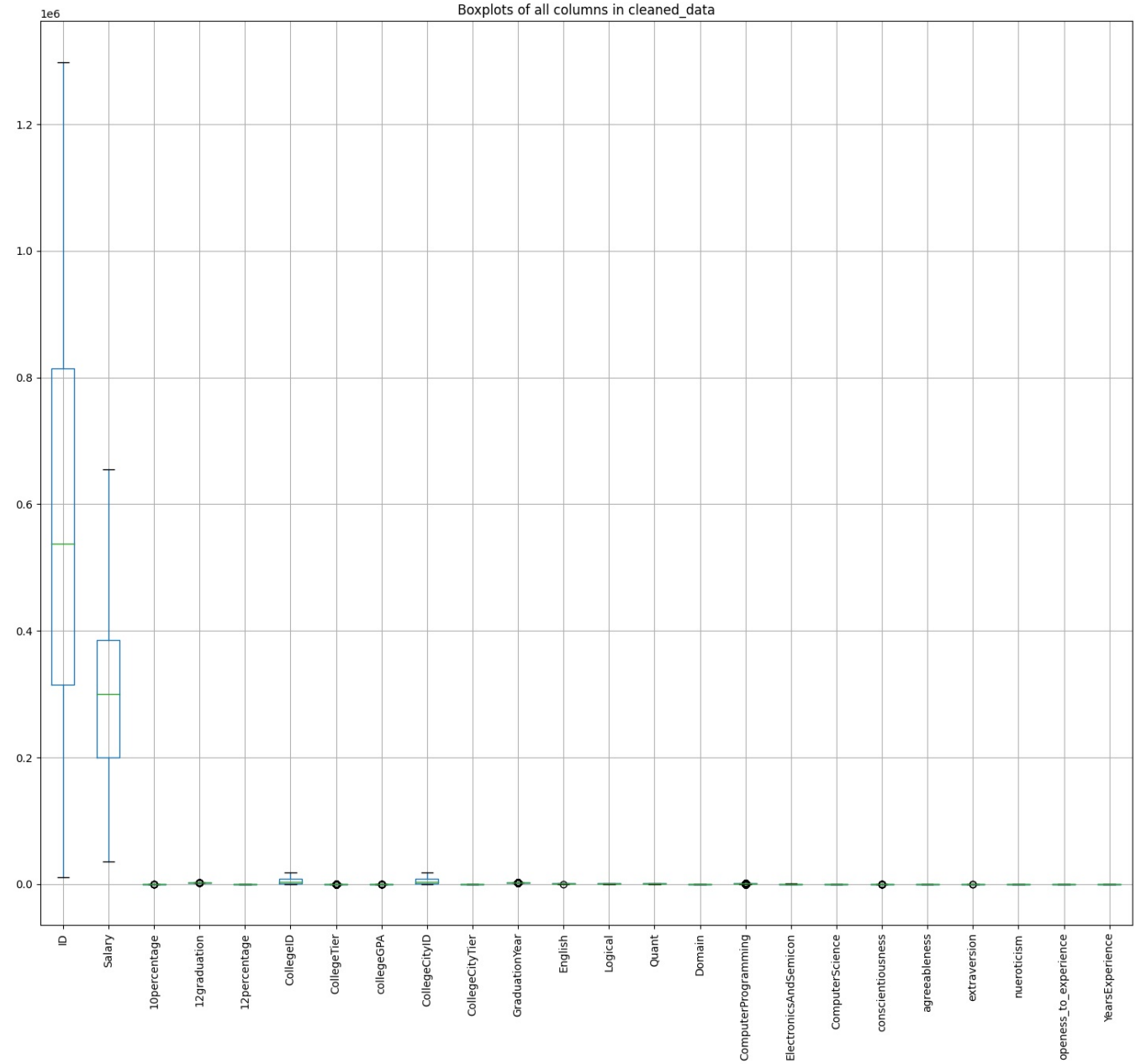
8 rows × 24 columns

In [ ]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2578 entries, 0 to 3997
Data columns (total 36 columns):
 #   Column                 Non-Null Count  Dtype         
---  ------                 --------------  -----         
 0   Unnamed: 0             2578 non-null   object        
 1   ID                     2578 non-null   int64         
 2   Salary                 2578 non-null   int64         
 3   DOJ                    2578 non-null   datetime64[ns]
 4   DOL                    2578 non-null   object        
 5   Designation            2578 non-null   object        
 6   JobCity                2578 non-null   object        
 7   Gender                 2578 non-null   object        
 8   DOB                    2578 non-null   datetime64[ns]
 9   10percentage           2578 non-null   float64       
 10  10board                2578 non-null   object        
 11  12graduation           2578 non-null   int64         
 12  12percentage           2578 non-null   float64       
 13  12board                2578 non-null   object        
 14  CollegeID              2578 non-null   int64         
 15  CollegeTier            2578 non-null   int64         
 16  Degree                 2578 non-null   object        
 17  Specialization         2578 non-null   object        
 18  collegeGPA             2578 non-null   float64       
 19  CollegeCityID          2578 non-null   int64         
 20  CollegeCityTier        2578 non-null   int64         
 21  CollegeState           2578 non-null   object        
 22  GraduationYear         2578 non-null   float64       
 23  English                2578 non-null   int64         
 24  Logical                2578 non-null   int64         
 25  Quant                  2578 non-null   int64         
 26  Domain                 2578 non-null   float64       
 27  ComputerProgramming    2578 non-null   float64       
 28  ElectronicsAndSemicon  2578 non-null   int64         
 29  ComputerScience        2578 non-null   int64         
 30  conscientiousness      2578 non-null   float64       
 31  agreeableness          2578 non-null   float64       
 32  extraversion           2578 non-null   float64       
 33  nueroticism            2578 non-null   float64       
 34  openess_to_experience  2578 non-null   float64       
 35  YearsExperience        2578 non-null   int64         
dtypes: datetime64[ns](2), float64(11), int64(13), object(10)
memory usage: 745.2+ KB
```

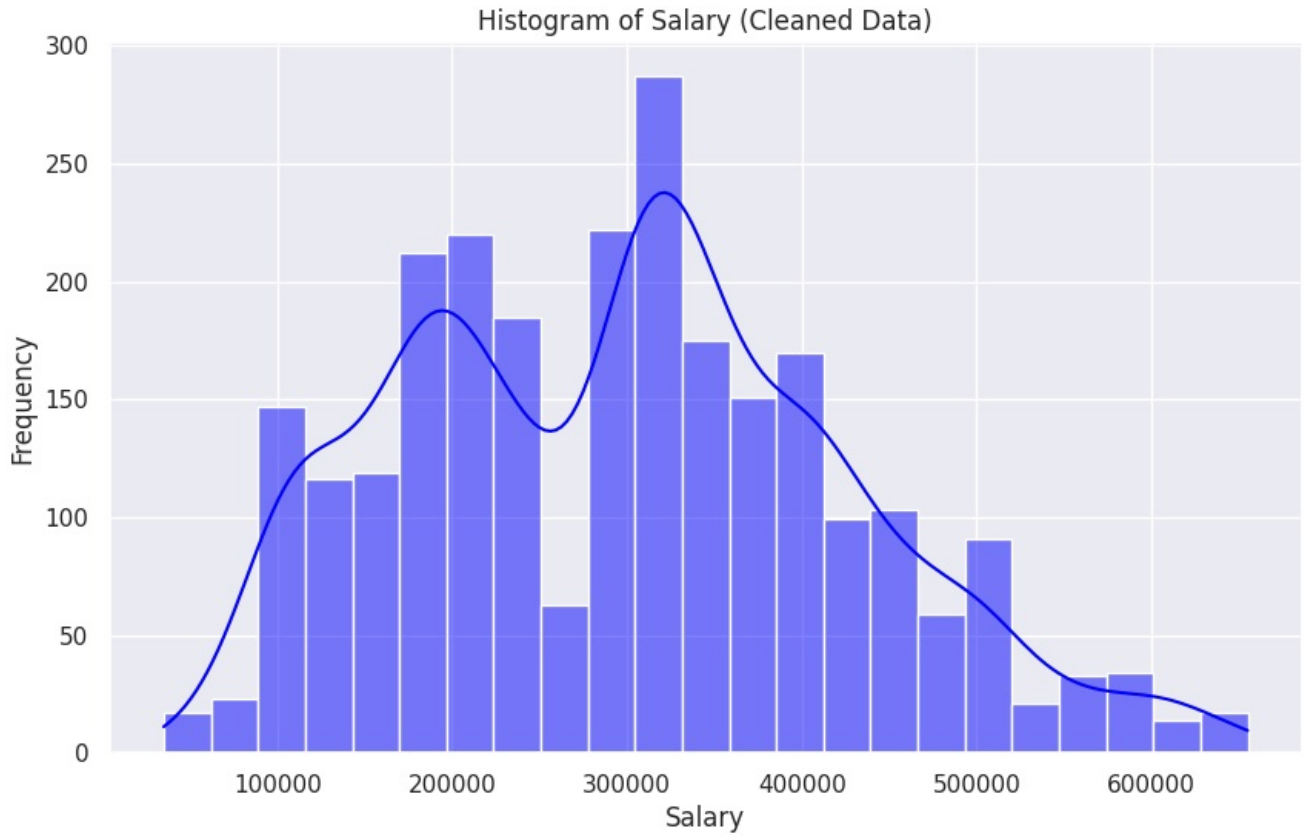
In [ ]:

```
#Boxplots of all columns in cleaned_data
plt.figure(figsize=(18, 15))
df_copy.boxplot(rot=90)
plt.title("Boxplots of all columns in cleaned_data")
plt.show()
```



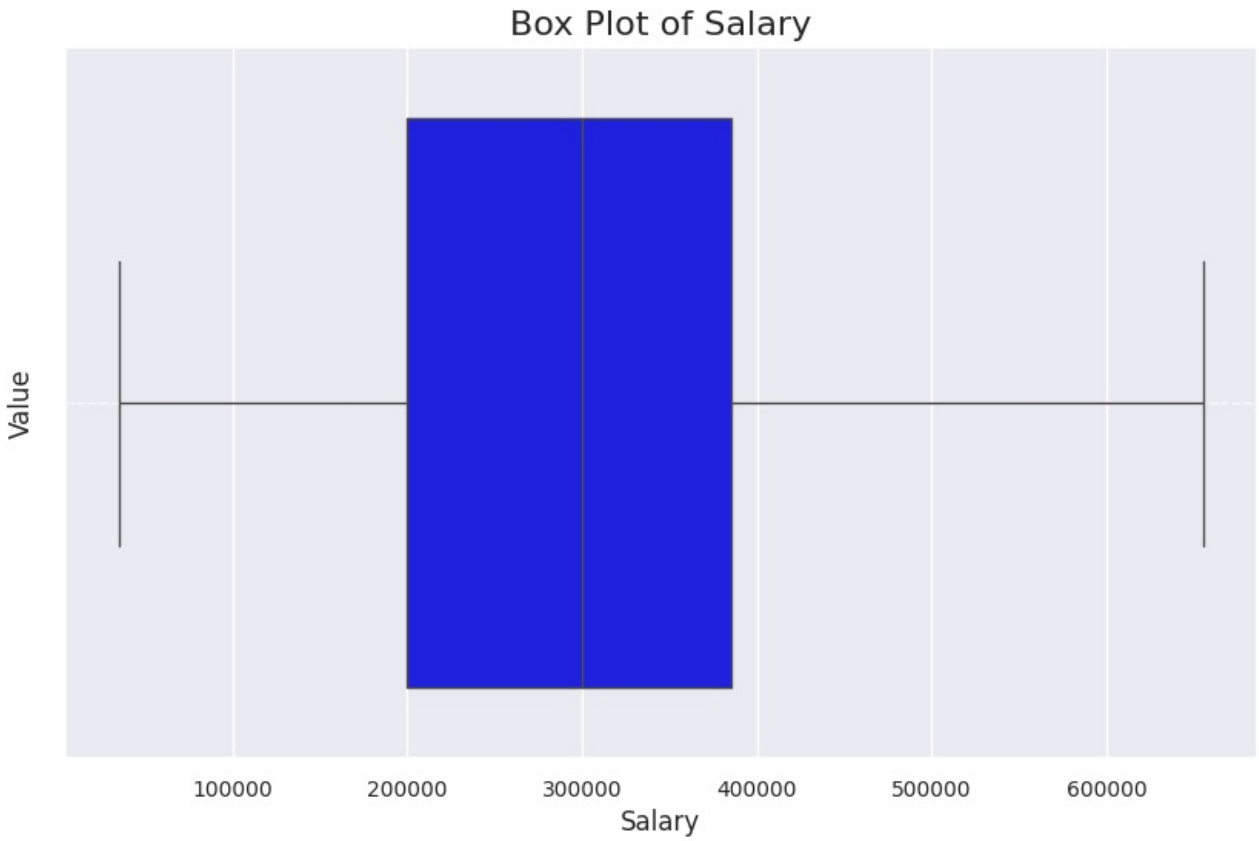
Salary:

In [ ]:
```
# Histogram of Salary
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="Salary", kde=True, color='Blue')
plt.title("Histogram of Salary (Cleaned Data)")
plt.xlabel("Salary")
plt.ylabel("Frequency")
plt.show()
```



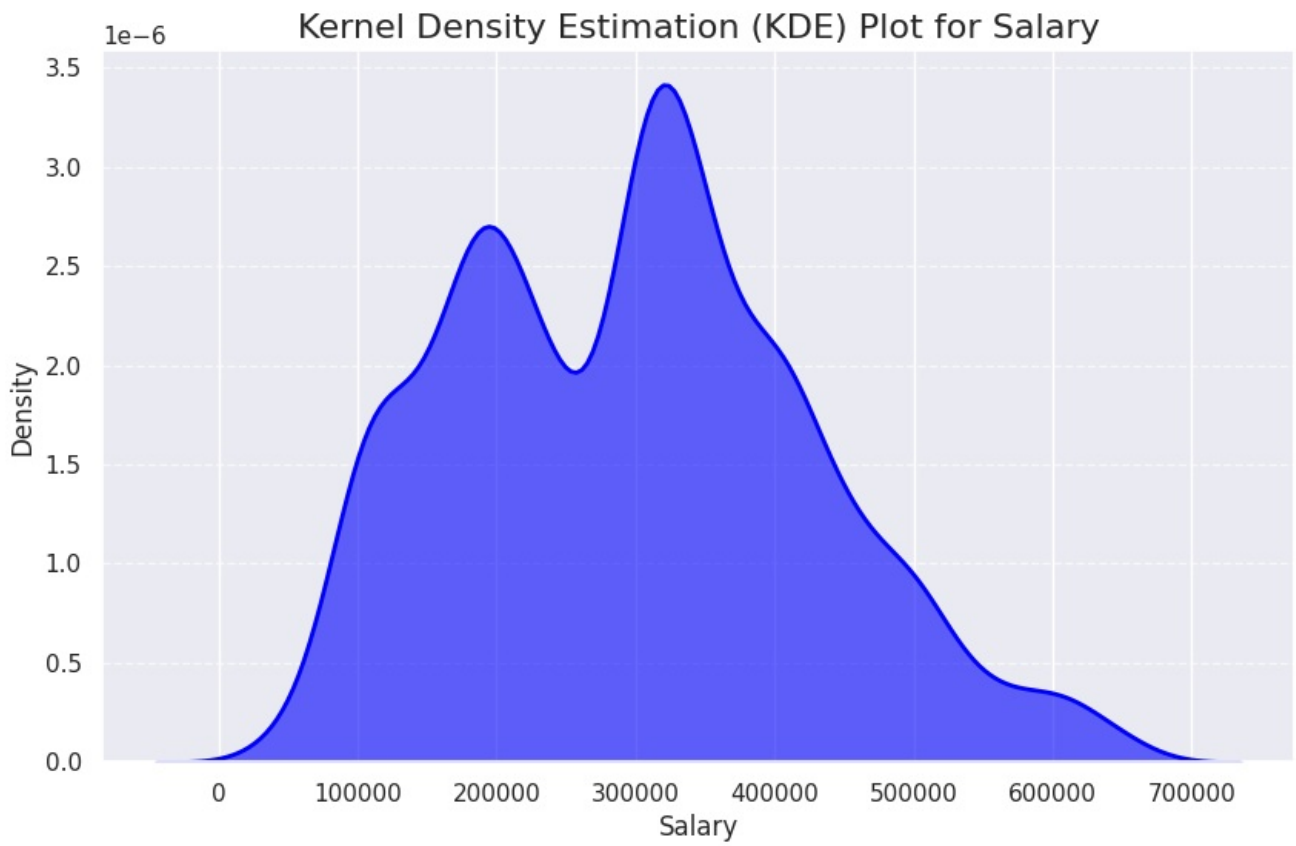
observations: The majority of salaries fall within the range of 200,000 to 400,000. This indicates a concentration of individuals earning salaries in this bracket.

In [ ]:
```
#Box Plot of Salary

sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["Salary"], color='Blue', flierprops=dict(marker='o', markersize=8, linestyle='none'))
plt.title("Box Plot of Salary", fontsize=16)
plt.xlabel("Salary", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observation: The box plot shows the middle 50% of the data, with the median salary in the middle of the box

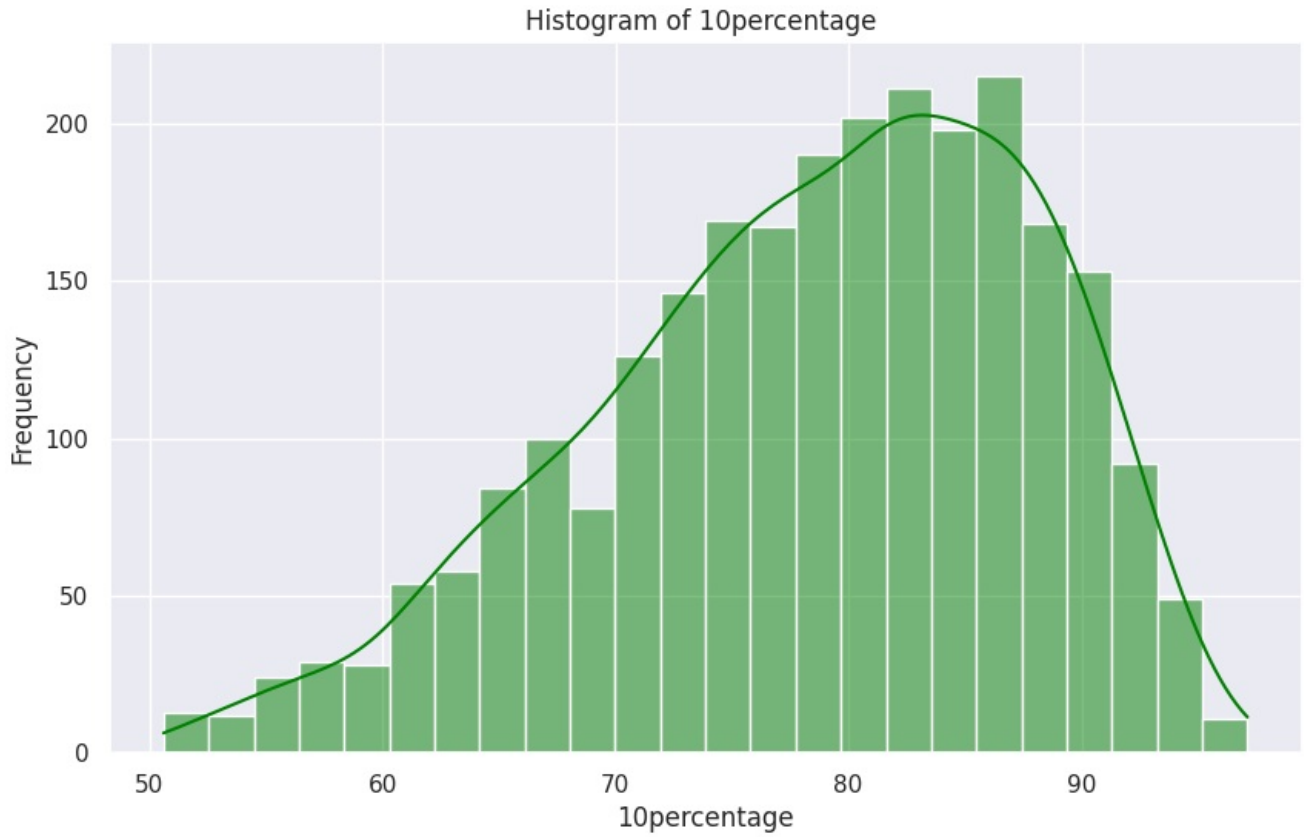
In [ ]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.kdeplot(df_copy['Salary'].dropna(), fill=True, color='blue', alpha=0.6, linewidth=2)
plt.title('Kernel Density Estimation (KDE) Plot for Salary', fontsize=16)
plt.xlabel('Salary', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations: The bimodal distribution observed in the KDE plot suggests two common salary ranges, with peaks around 100,000 and 400,000. This finding can guide compensation strategies and talent management decisions

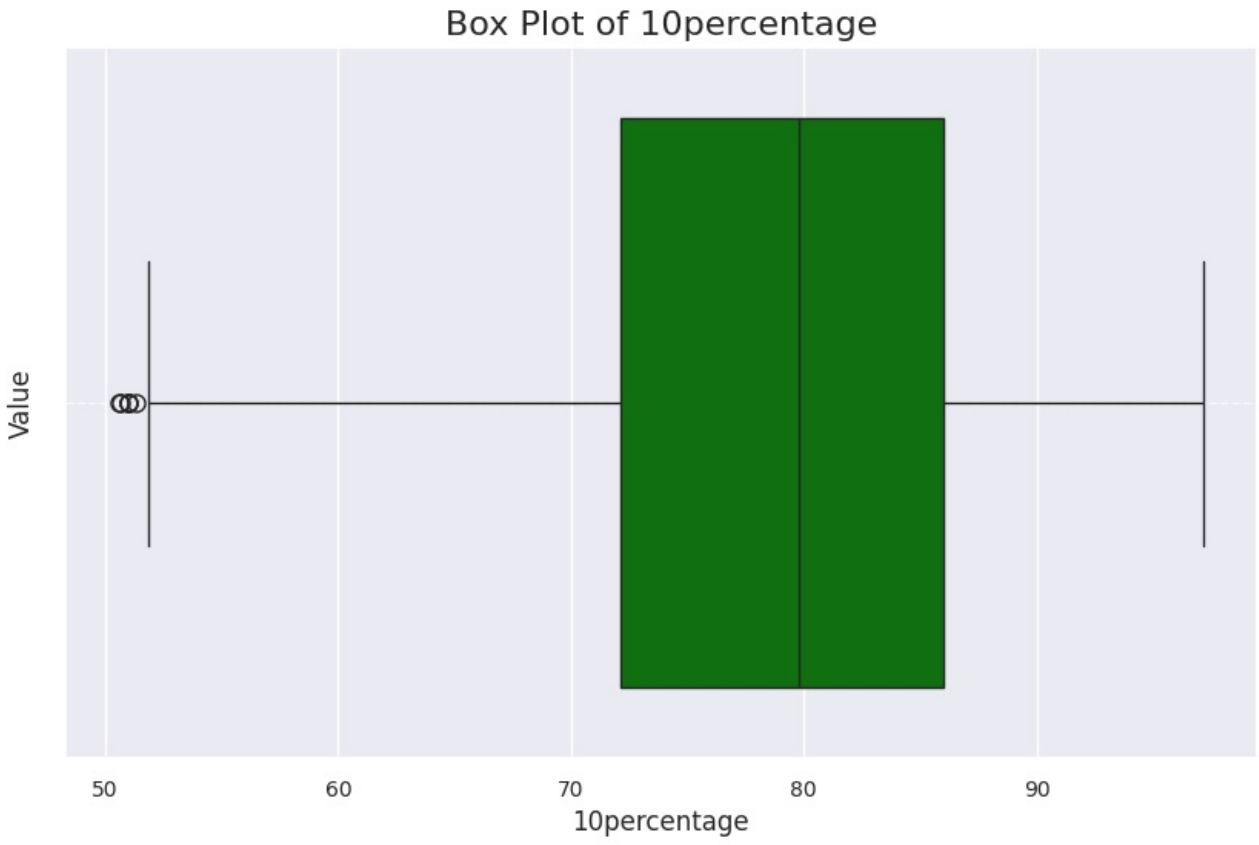
10 percentage

In [ ]:
```
# Histogram of Salary
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="10percentage", kde=True, color='green')
plt.title("Histogram of 10percentage ")
plt.xlabel("10percentage")
plt.ylabel("Frequency")
plt.show()
```



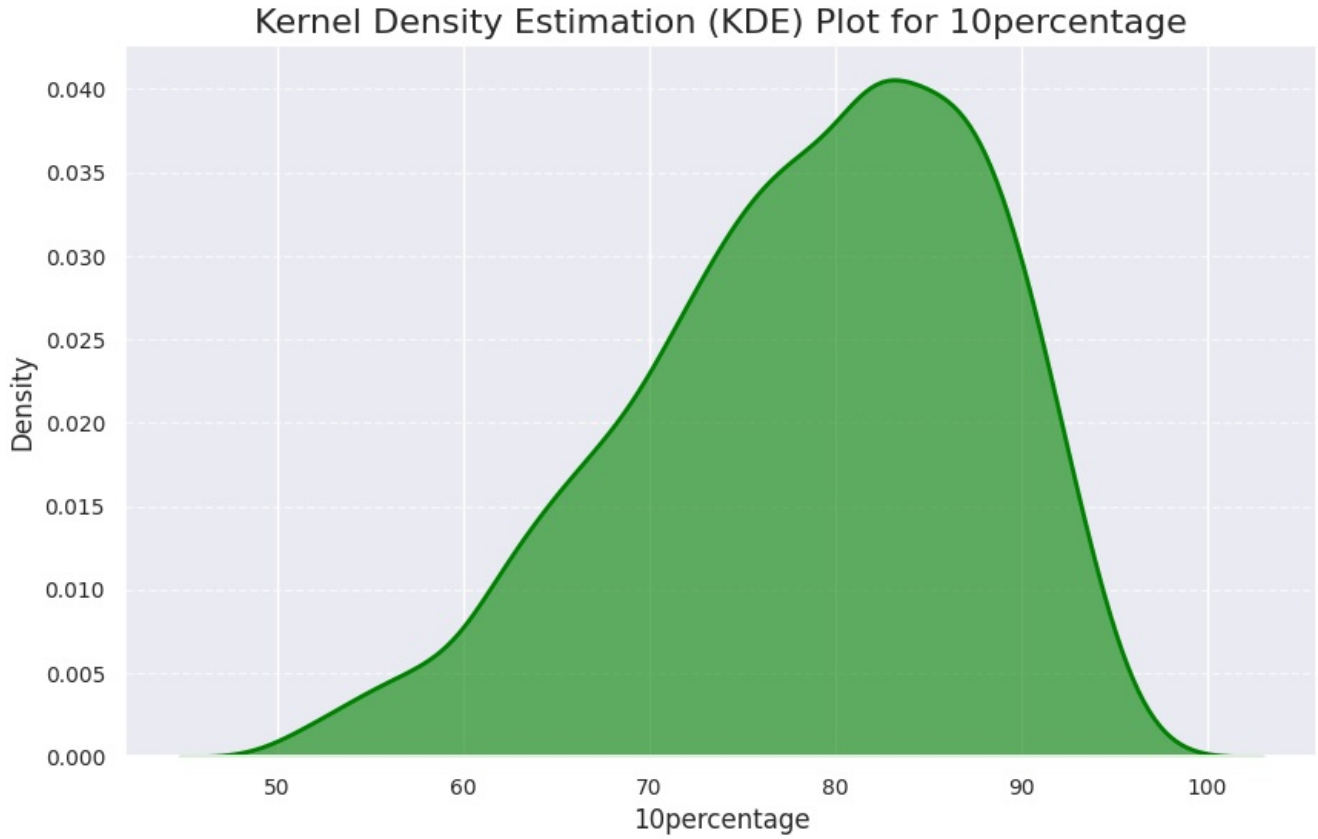
observations:The histogram reveals a high concentration of data points in the 80 to 90 range for the labeled dataset “10percentage.

In [ ]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["10percentage"], color='green', flierprops=dict(marker='o', markersize=8, linestyle='none
plt.title("Box Plot of 10percentage", fontsize=16)
plt.xlabel("10percentage", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations: The box plot for the “10percentage” reveals that most data points are concentrated between approximately 80 and 90, with some outliers present below 60

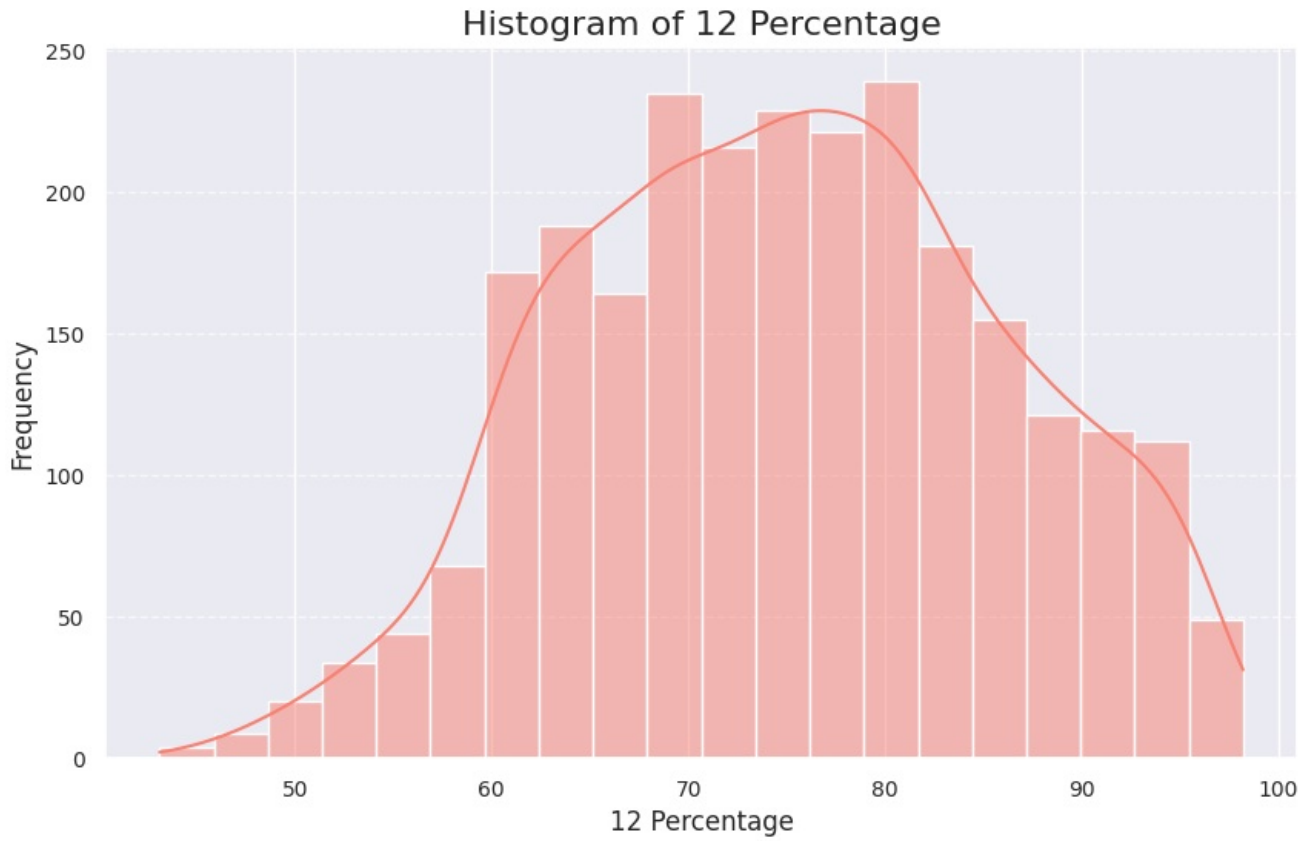
In [ ]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.kdeplot(df_copy['10percentage'].dropna(), fill=True, color='green', alpha=0.6, linewidth=2)
plt.title('Kernel Density Estimation (KDE) Plot for 10percentage', fontsize=16)
plt.xlabel('10percentage', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations:The (KDE) plot for the dataset labeled “10percentage” reveals a significant concentration of data points around the value of 90

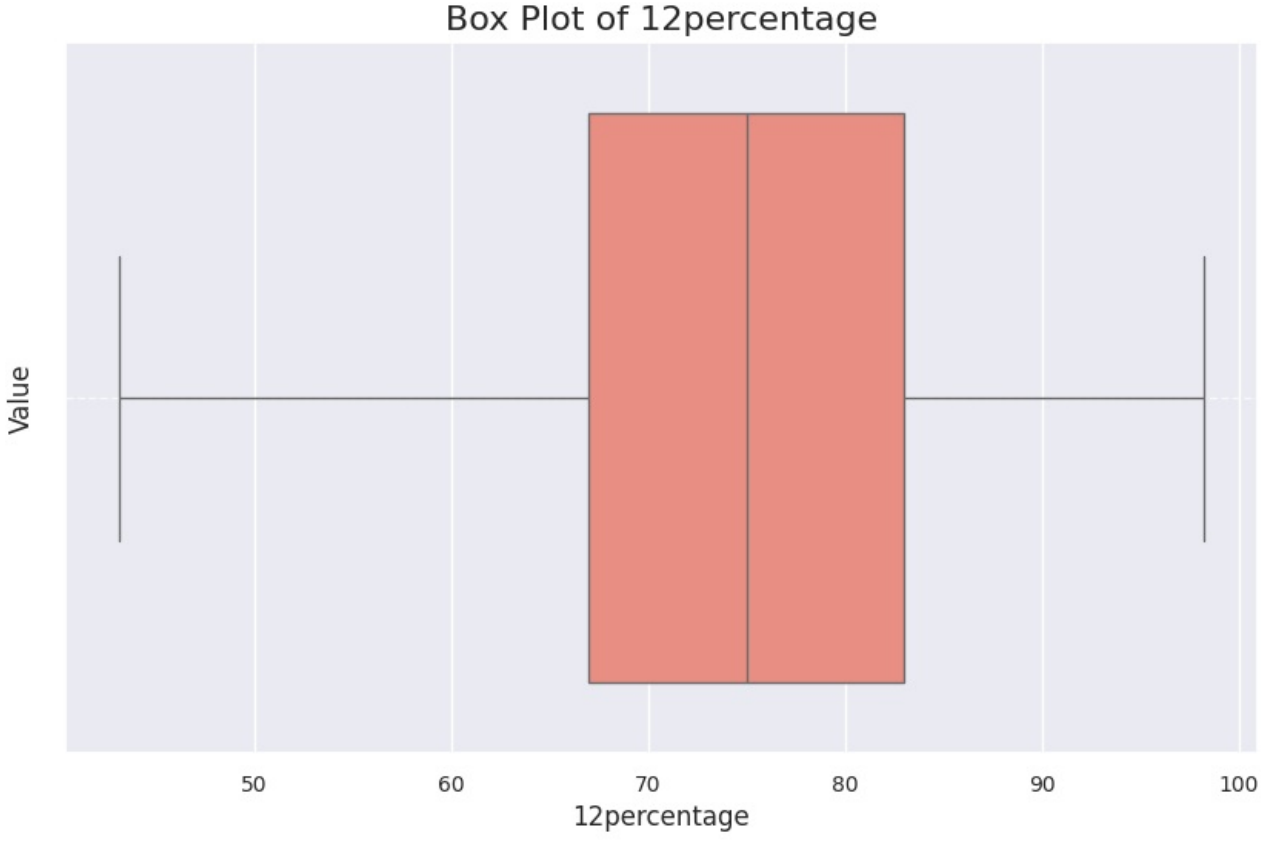
12 Percentage

In [ ]:
```
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="12percentage", kde=True, color='salmon', bins=20)
plt.title("Histogram of 12 Percentage", fontsize=16)
plt.xlabel("12 Percentage", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations: The histogram of 12 Percentage scores reveals a concentration of data around the 75% mark, indicating the common performance level of the students

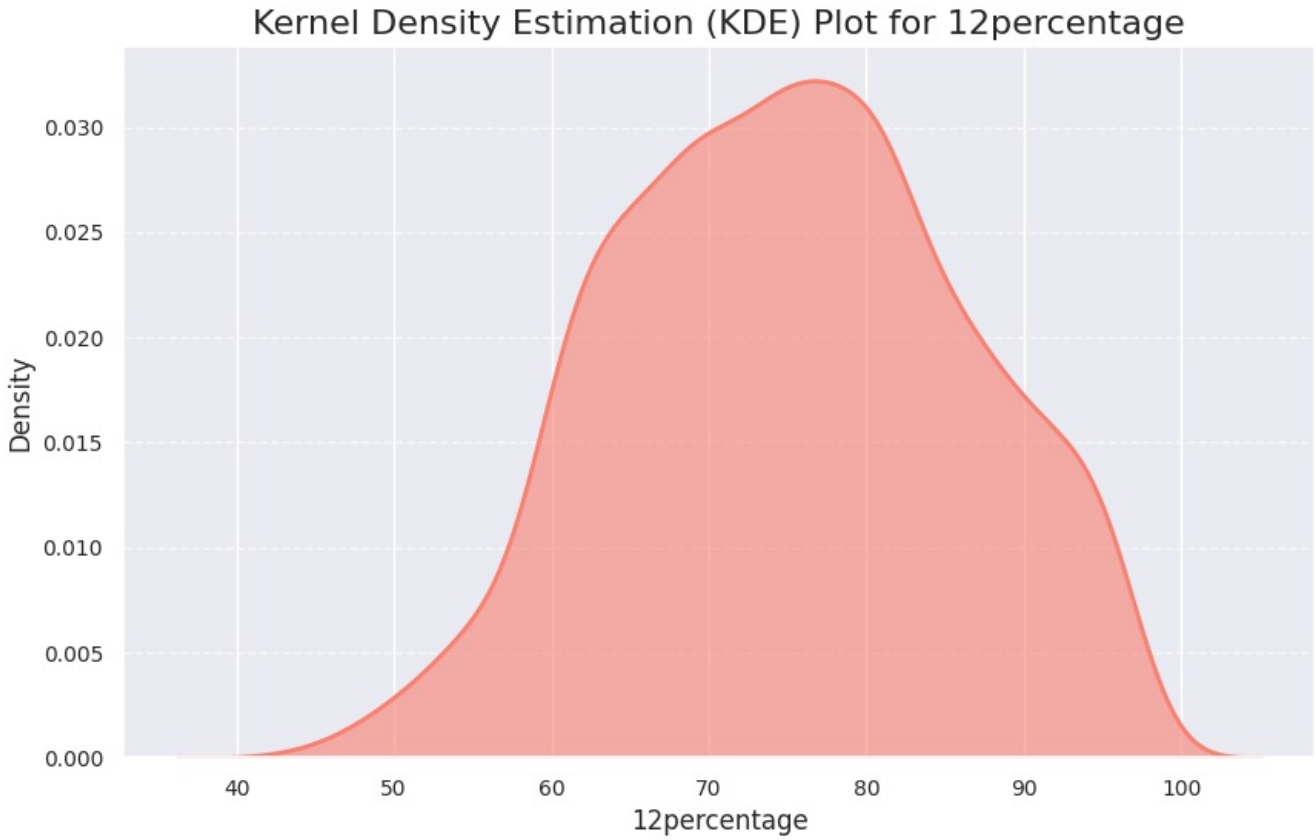
In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["12percentage"], color='salmon', flierprops=dict(marker='o', markersize=8, linestyle='non
plt.title("Box Plot of 12percentage", fontsize=16)
plt.xlabel("12percentage", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



obsevations: the 12 percentages are spread out between 50% and 90%, with a median of around 70%. There are no outliers.

In [ ]:
```
plt.figure(figsize=(10, 6))
```

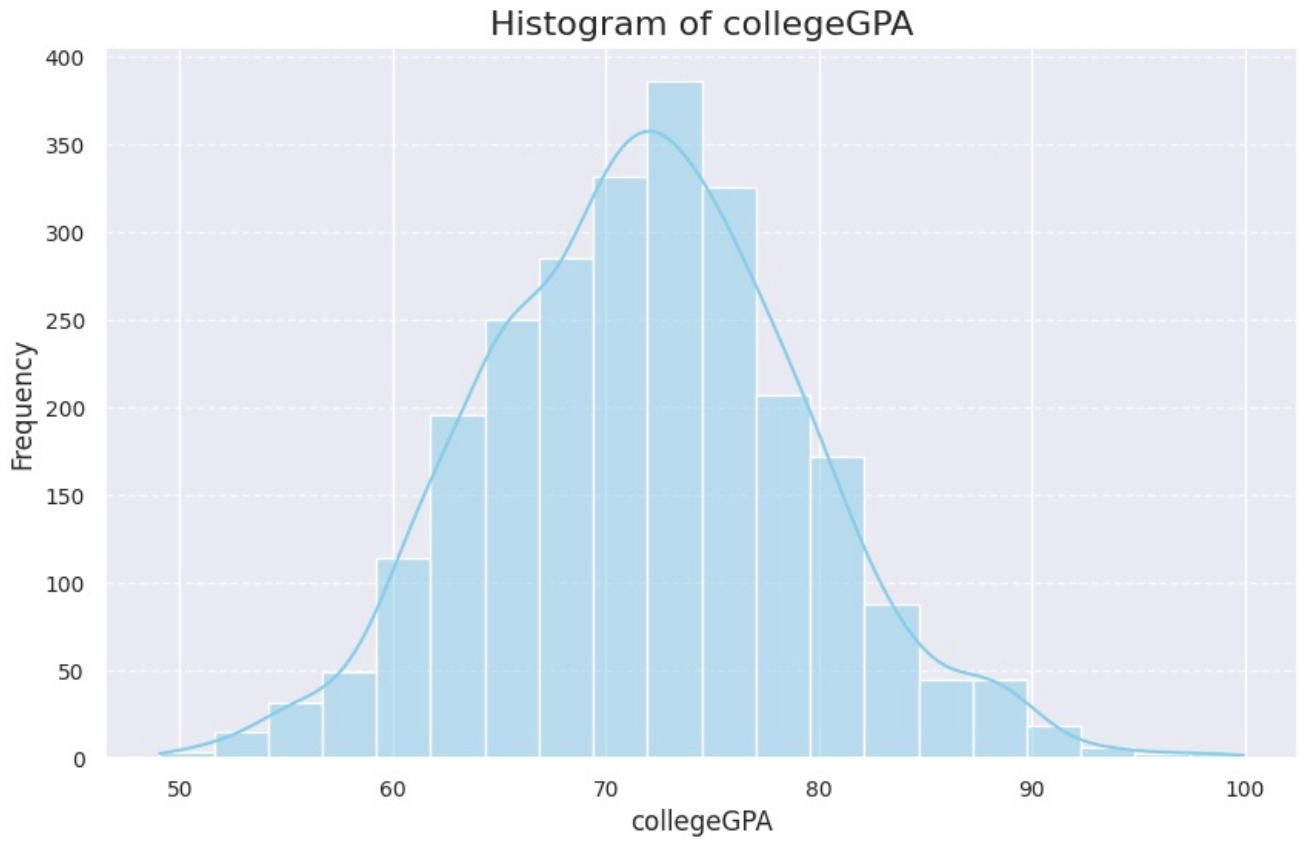
```
sns.kdeplot(df_copy['12percentage'].dropna(), fill=True, color='salmon', alpha=0.6, linewidth=2)
plt.title('Kernel Density Estimation (KDE) Plot for 12percentage', fontsize=16)
plt.xlabel('12percentage', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observation:The KDE plot suggests that a significant proportion of the 12th-grade percentage scores fall within the range of 70% to 90%, indicating a concentration of student performance in this band

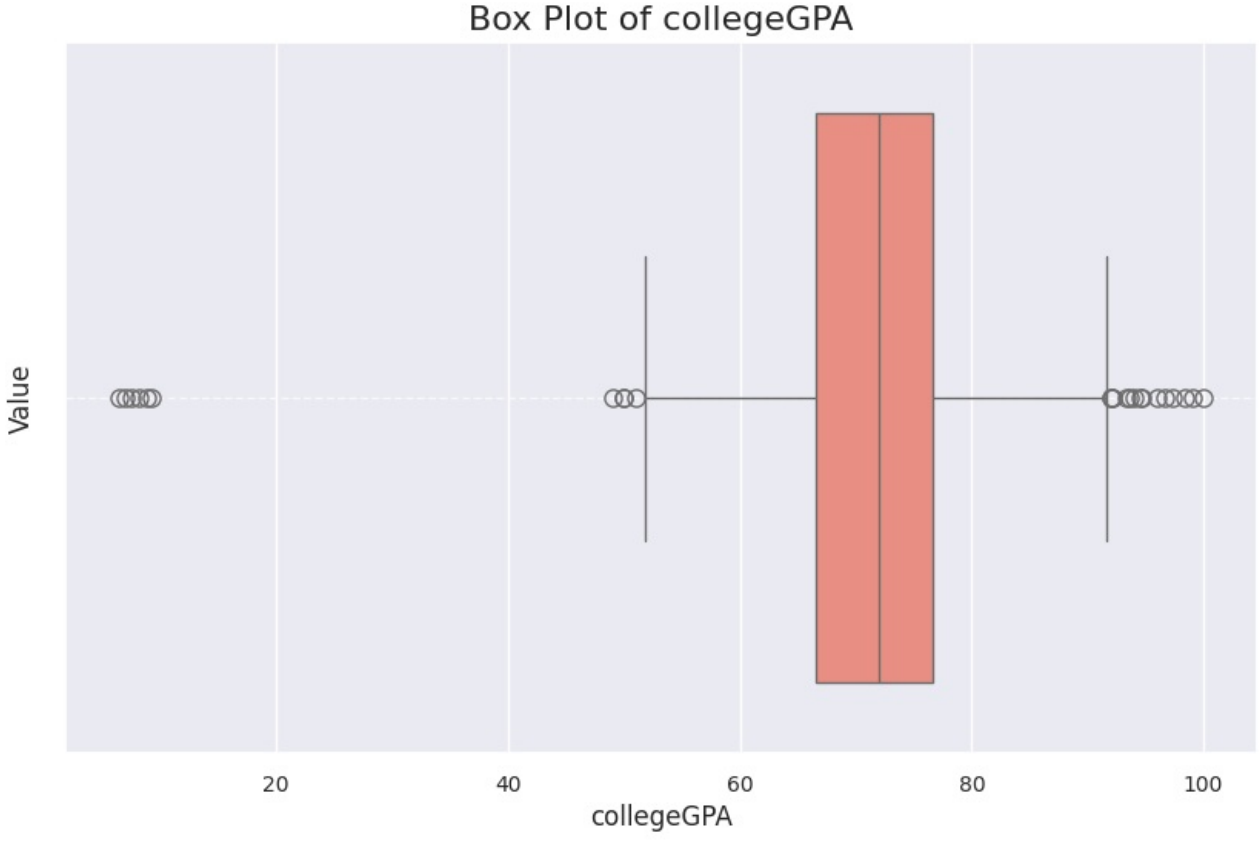
collegeGPA

In [ ]:
```
# Histogram of collegeGPA
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="collegeGPA", kde=True, color='skyblue', bins=20)
plt.title("Histogram of collegeGPA", fontsize=16)
plt.xlabel("collegeGPA", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations: The histogram reveals that the majority of students exhibit a normal distribution of college GPAs, with a peak around 70. This suggests that a significant number of students fall within the 60-80 GPA range.

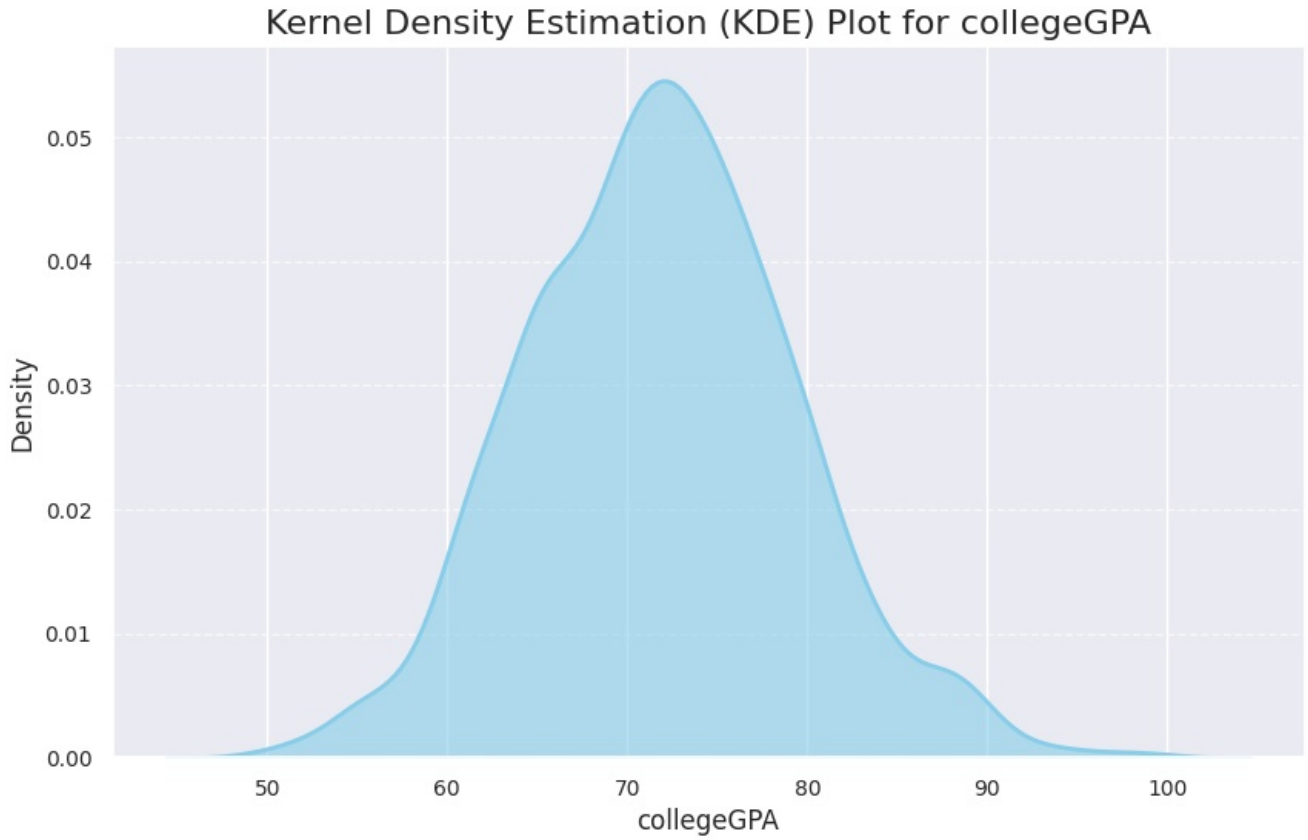
In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["collegeGPA"], color='salmon', flierprops=dict(marker='o', markersize=8, linestyle='none'
plt.title("Box Plot of collegeGPA", fontsize=16)
plt.xlabel("collegeGPA", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations: The box plot reveals a concentration of college GPA scores in the 60-80 range, with few outliers

In [ ]:
```
plt.figure(figsize=(10, 6))
```

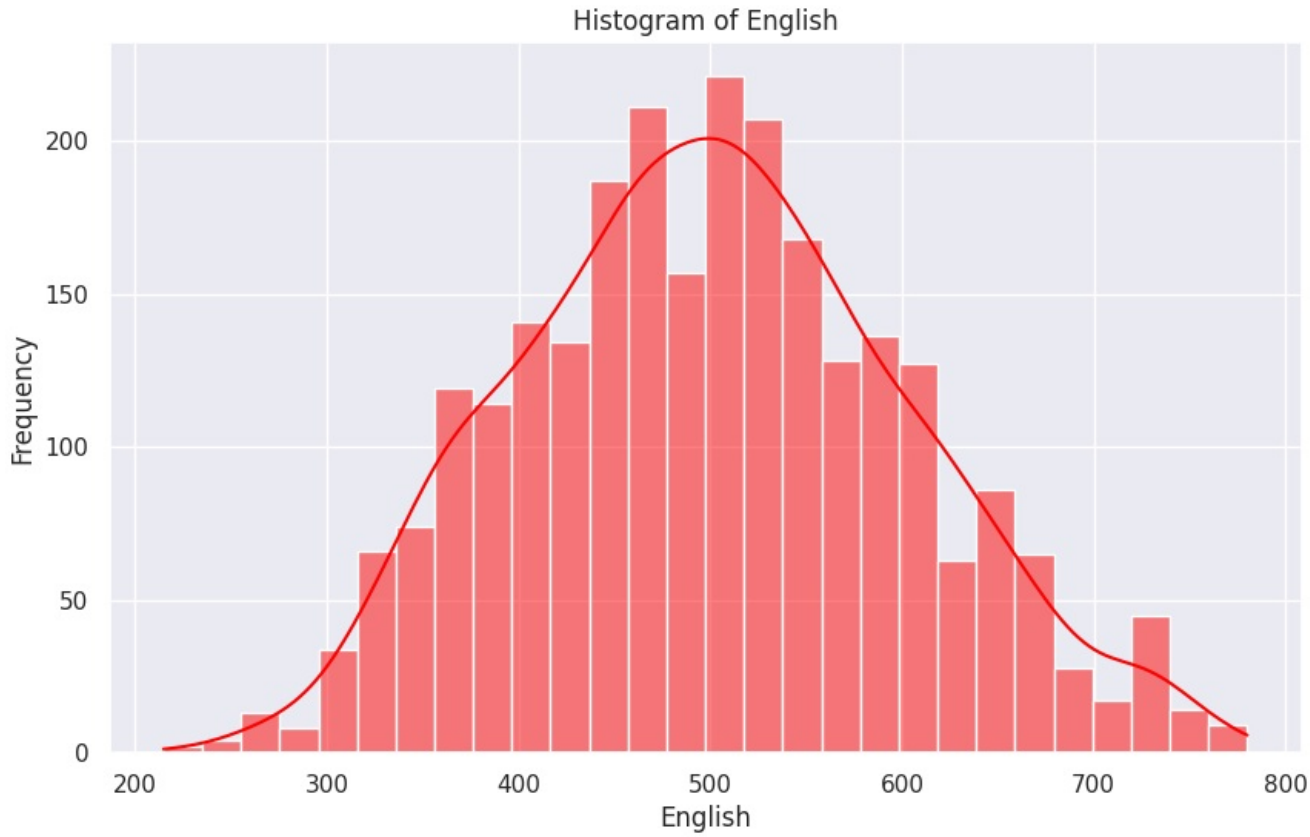
```
sns.kdeplot(df_copy['collegeGPA'].dropna(), fill=True, color='skyblue', alpha=0.6, linewidth=2)
plt.title('Kernel Density Estimation (KDE) Plot for collegeGPA', fontsize=16)
plt.xlabel('collegeGPA', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Obsevations: College GPAs appear normally distributed with a peak around 3.2.

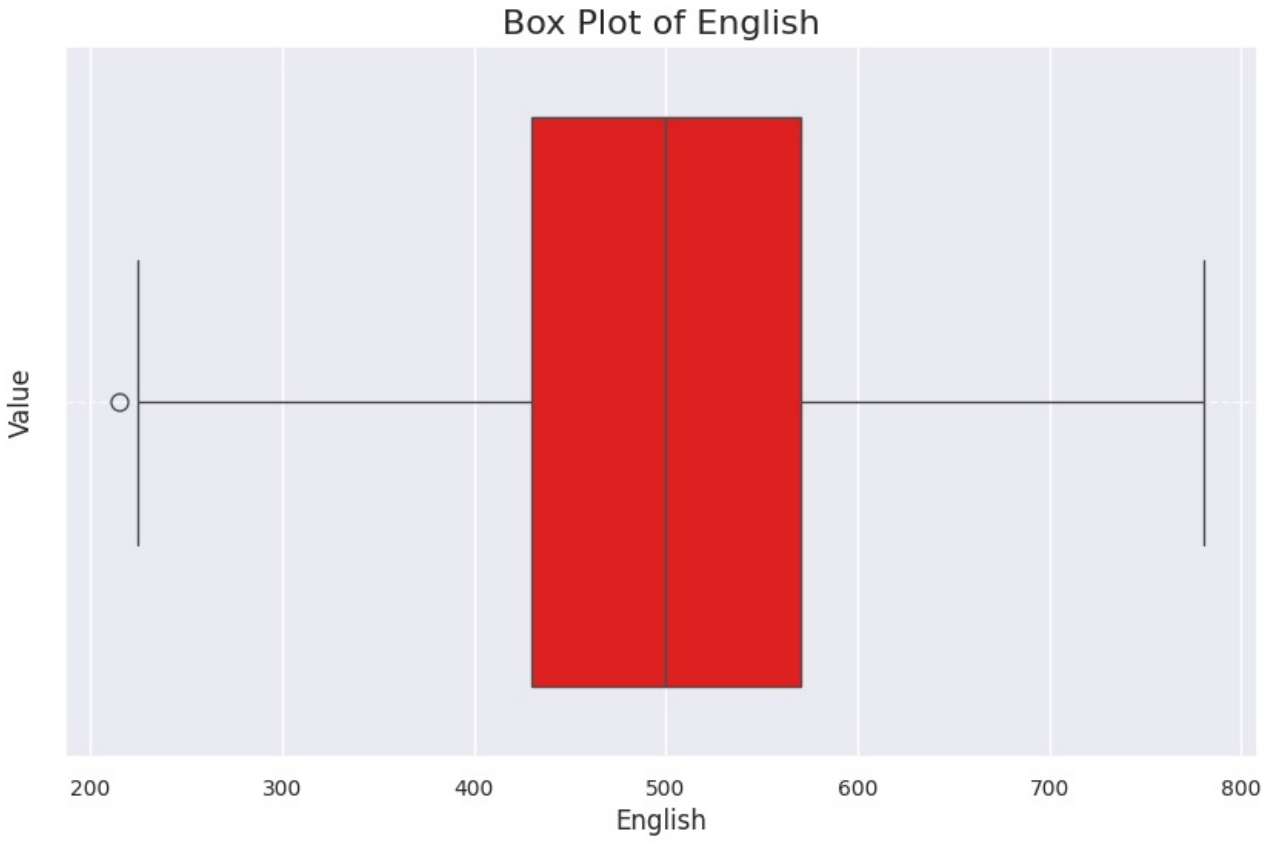
English

In [ ]:
```
# Histogram of English
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="English", kde=True, color='red')
plt.title("Histogram of English")
plt.xlabel("English")
plt.ylabel("Frequency")
plt.show()
```



Observations: The histogram indicates a normal distribution of English scores, with the majority of students scoring around 500, suggesting an average proficiency in the subject

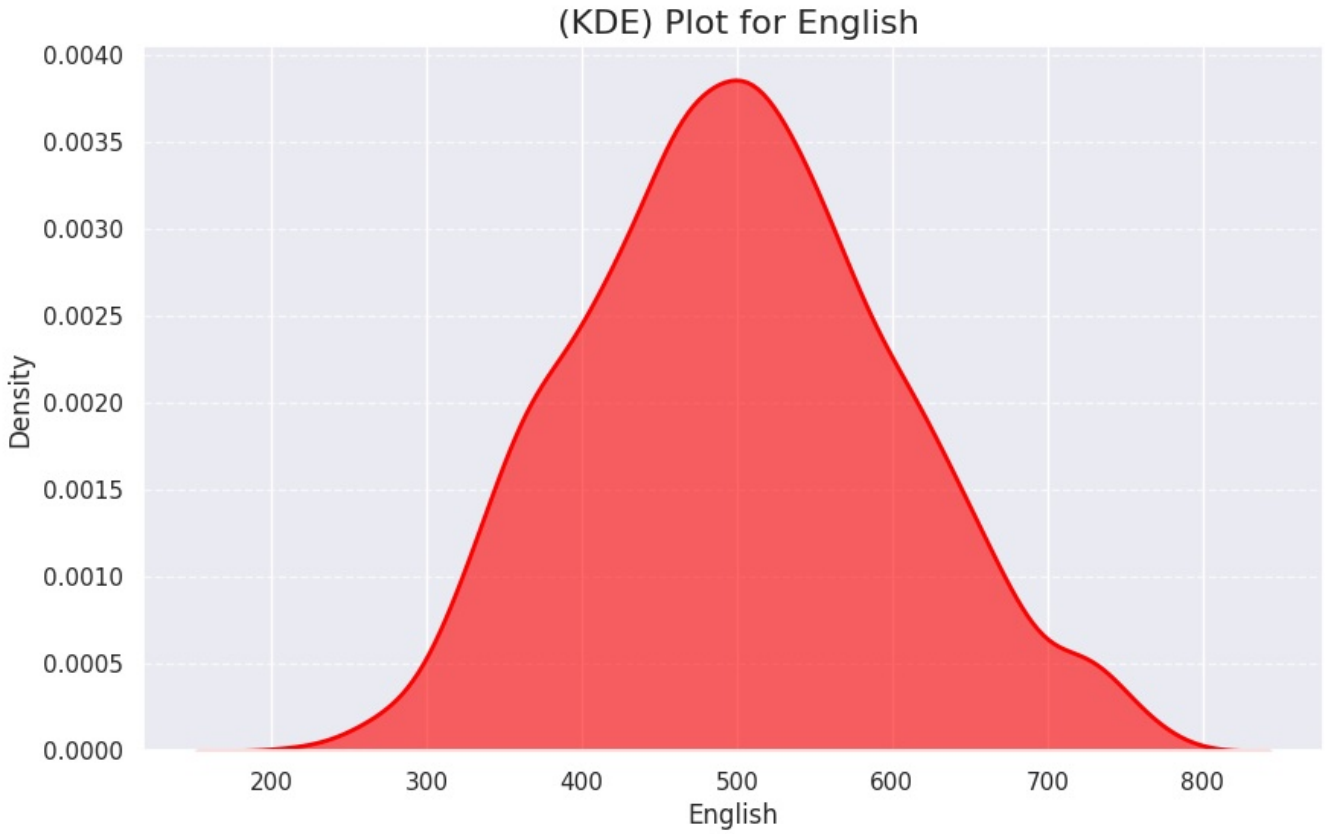
In [ ]:
```
#Box Plot of English

plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["English"], color='red', flierprops=dict(marker='o', markersize=8, linestyle='none'))
plt.title("Box Plot of English", fontsize=16)
plt.xlabel("English", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations: The box plot indicates a concentration of English scores around the 500 mark, with outliers present on both ends of the

score spectrum

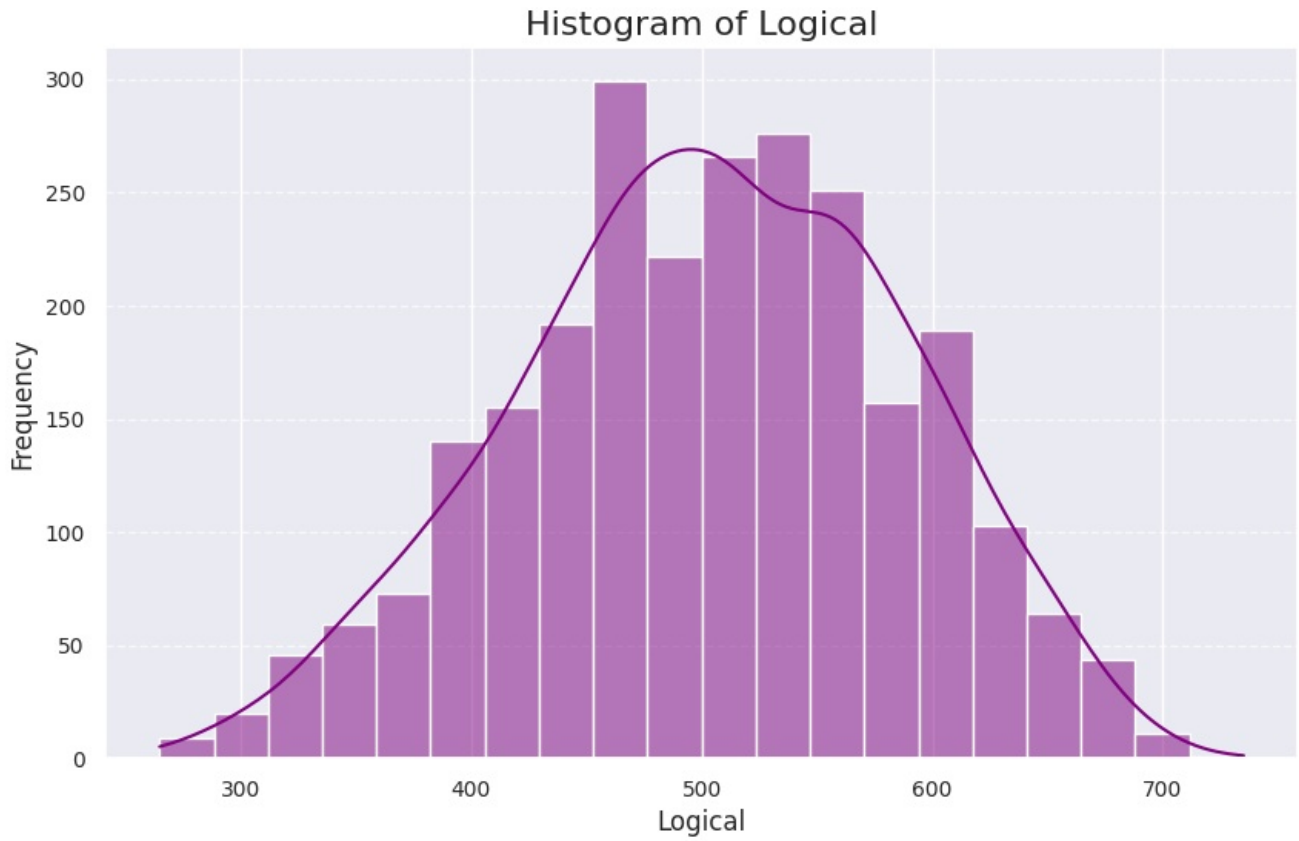
In [ ]:
```
#KDE Plot for English

plt.figure(figsize=(10, 6))
sns.kdeplot(df_copy['English'].dropna(), fill=True, color='red', alpha=0.6, linewidth=2)
plt.title(' (KDE) Plot for English', fontsize=16)
plt.xlabel('English', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations: The KDE plot indicates a normal distribution of English scores, with the majority of students scoring around 500, providing insights into the concentration and spread of students' performance in English.

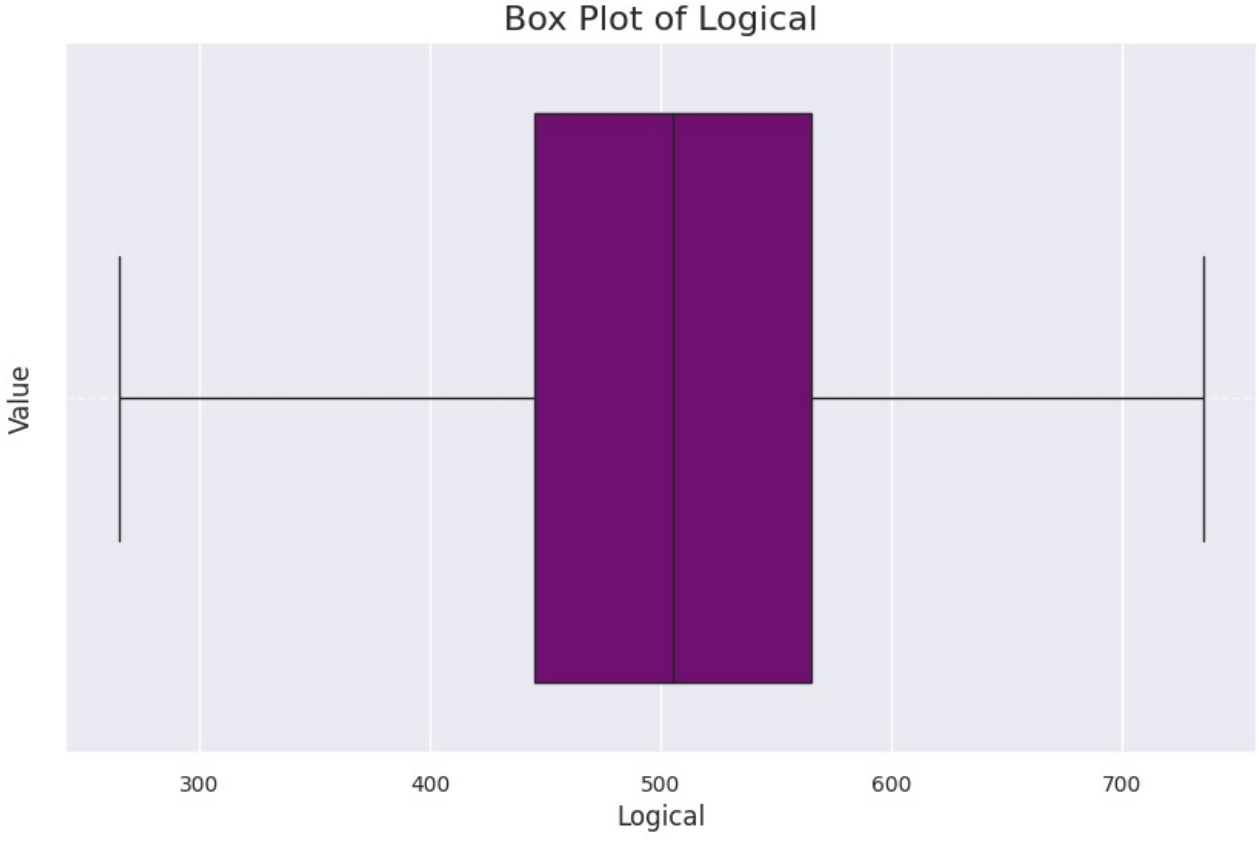
Logical

In [ ]:
```
#Histogram of Logical
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="Logical", kde=True, color='purple', bins=20)
plt.title("Histogram of Logical", fontsize=16)
plt.xlabel("Logical", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



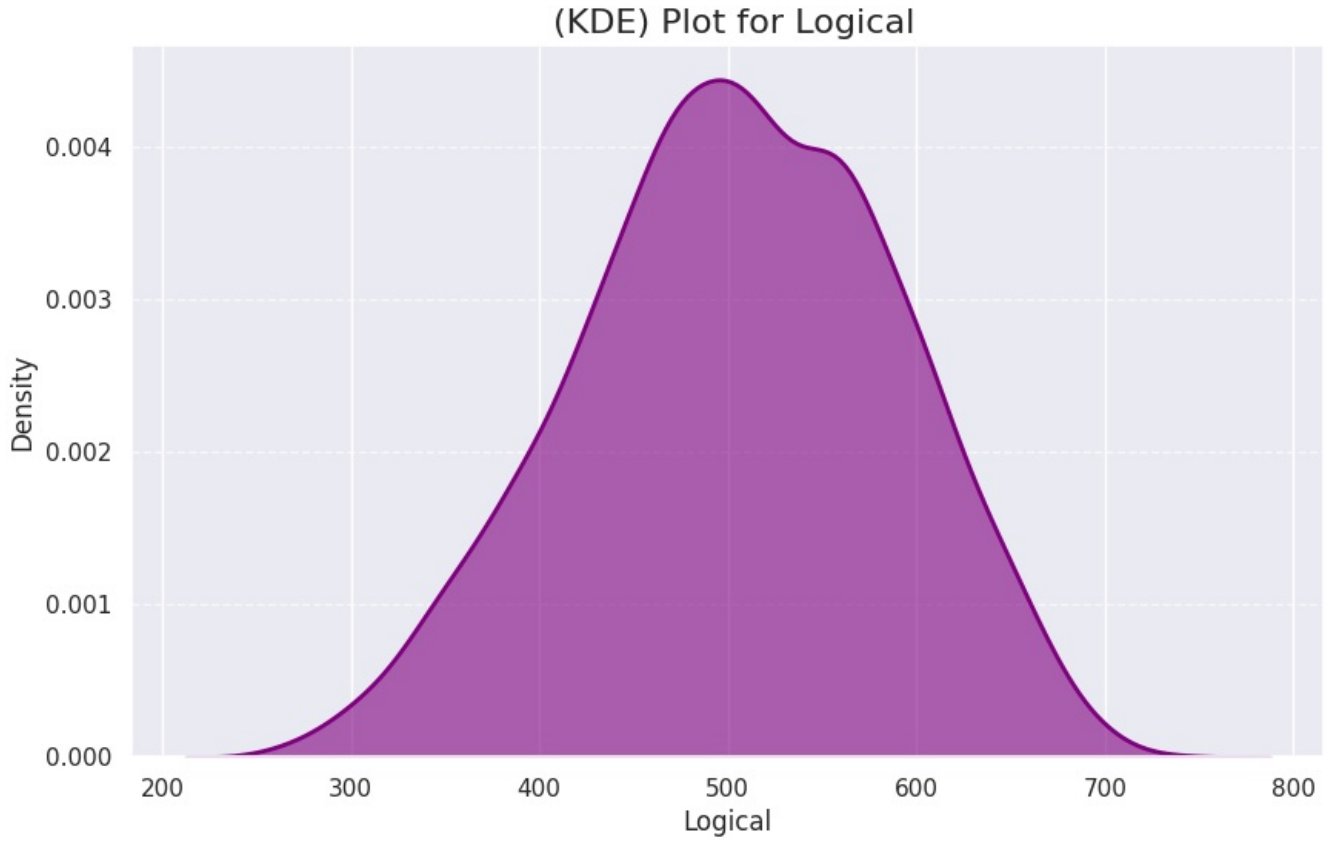
Observations:The histogram with the overlaid normal distribution curve indicates that most logical values cluster around the mean, providing insights into the commonality and variance of logical data points.

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["Logical"], color='purple', flierprops=dict(marker='o', markersize=8, linestyle='none'))
plt.title("Box Plot of Logical", fontsize=16)
plt.xlabel("Logical", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations: The box plot indicates that the majority of the “Logical” data values are concentrated around 500, with minimal spread, suggesting low variability and high consistency in the data

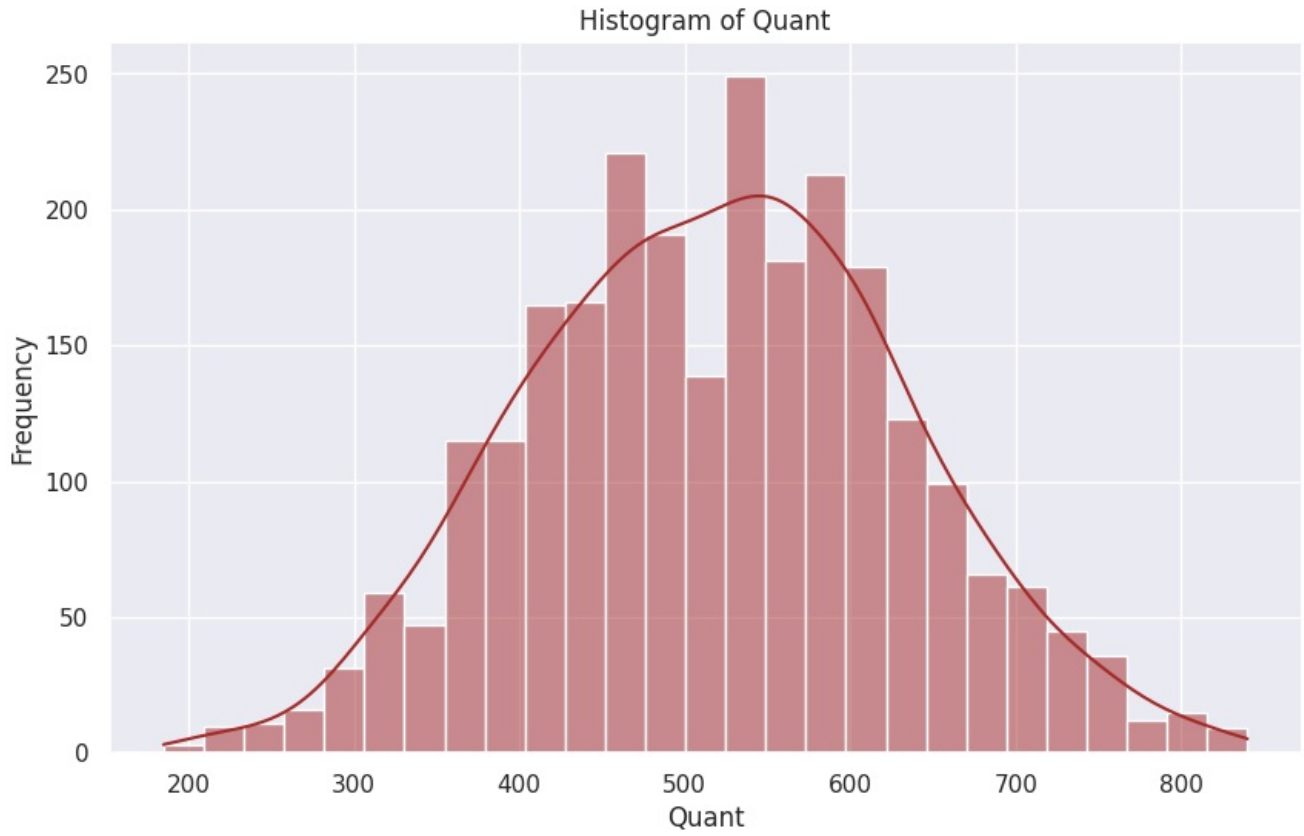
In [ ]:
```
#KDE Plot for Logical

plt.figure(figsize=(10, 6))
sns.kdeplot(df_copy['Logical'].dropna(), fill=True, color='purple', alpha=0.6, linewidth=2)
plt.title(' (KDE) Plot for Logical', fontsize=16)
plt.xlabel('Logical', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations:The KDE plot indicates a normal distribution of logical scores, with the majority of data points clustered around 500

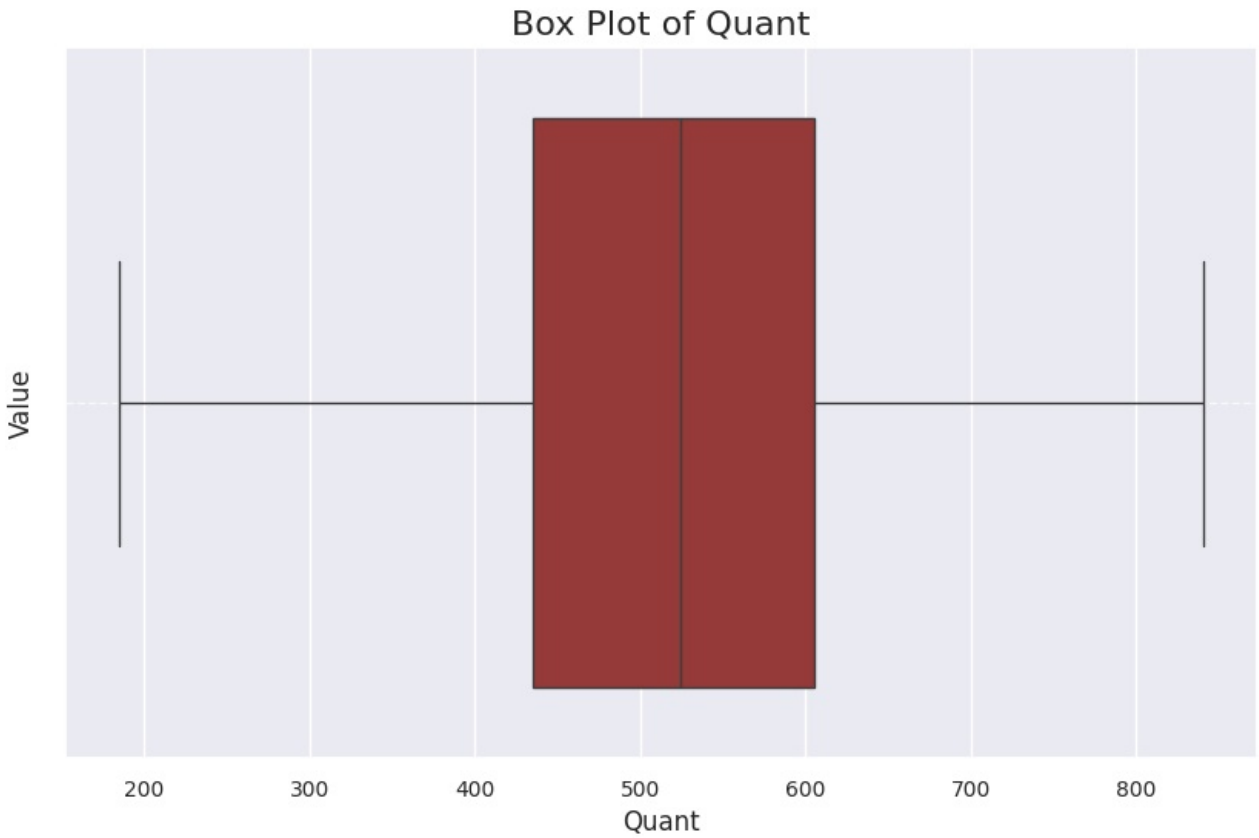
Quant

In [ ]:
```
# Histogram of Quant
plt.figure(figsize=(10, 6))
sns.histplot(data=df_copy, x="Quant", kde=True, color='brown')
plt.title("Histogram of Quant")
plt.xlabel("Quant")
plt.ylabel("Frequency")
plt.show()
```



Observations: The histogram with the overlaid normal distribution curve indicates that most “Quant” data points are clustered around the value of 500, providing insights into the commonality and variance of these Quant values

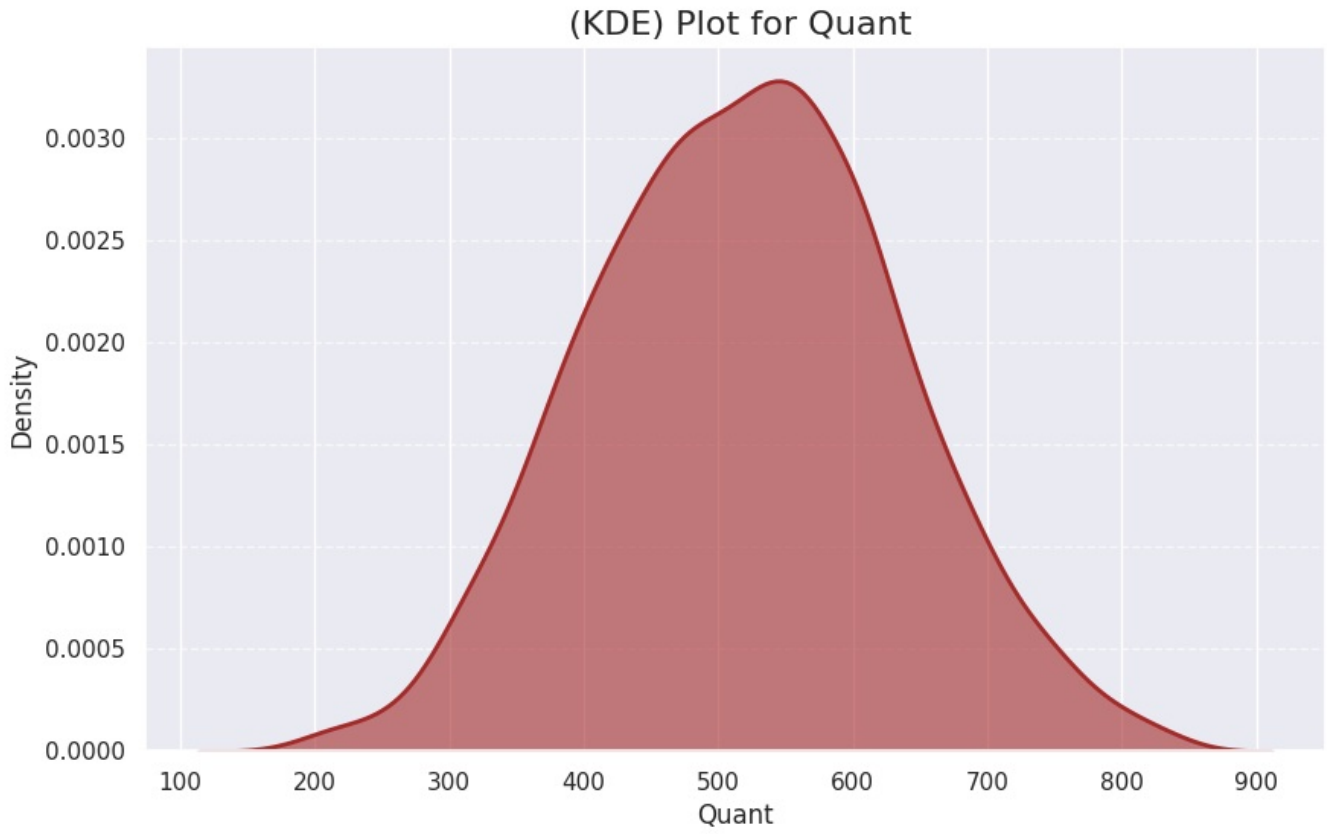
In [ ]:
```
#Box Plot of Quant

sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.boxplot(x=df_copy["Quant"], color='brown', flierprops=dict(marker='o', markersize=8, linestyle='none'))
plt.title("Box Plot of Quant", fontsize=16)
plt.xlabel("Quant", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Observations: The box plot indicates a concentration of data values between approximately 500 and 600, with outliers present below

around 250 and above near 750

In [ ]:
```
#KDE Plot for Quant
sns.set_theme(style="darkgrid")
plt.figure(figsize=(10, 6))
sns.kdeplot(df_copy['Quant'].dropna(), fill=True, color='brown', alpha=0.6, linewidth=2)
plt.title(' (KDE) Plot for Quant', fontsize=16)
plt.xlabel('Quant', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



observations:The KDE plot indicates a distribution of a quantitative variable, peaking around 500, suggesting that the majority of the data points are centered around this value.

Categorical analysis

In [ ]:
```
cat_df.head()
```

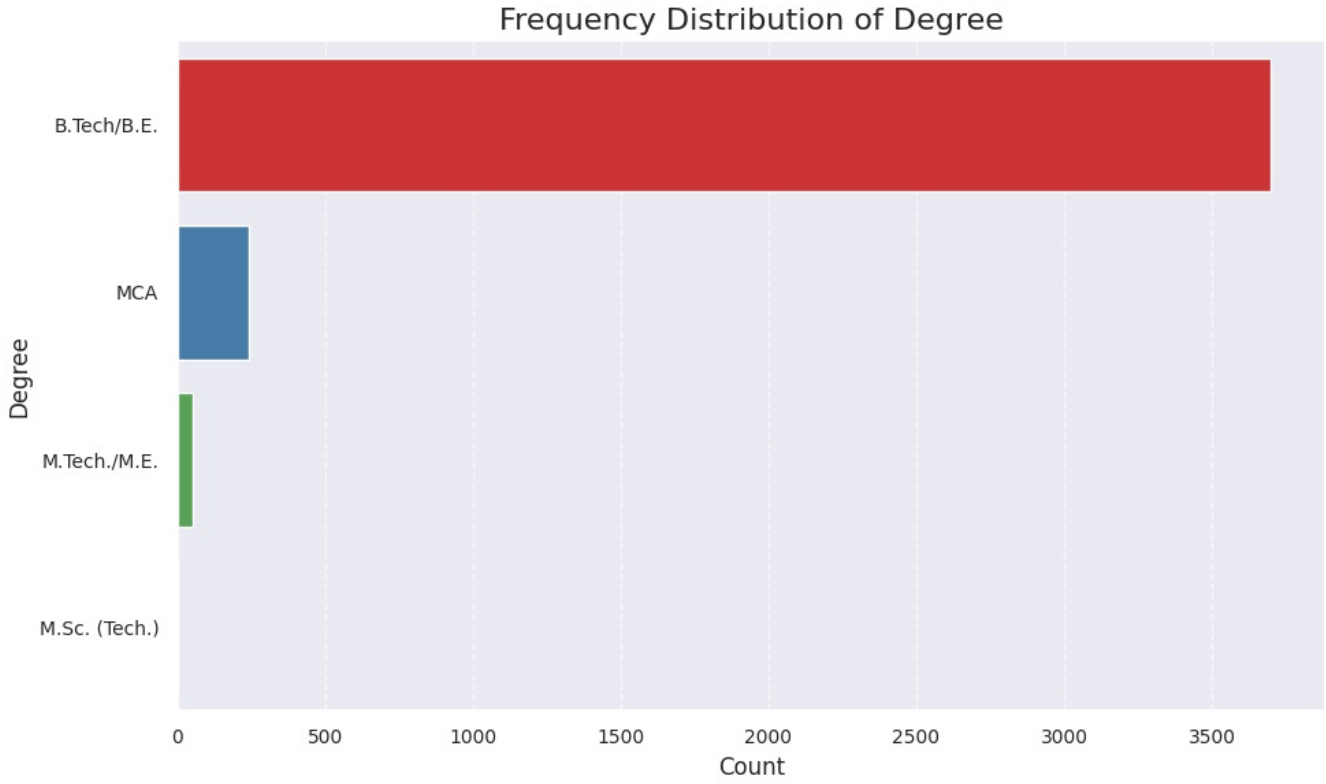
Out[ ]:

	Designation	JobCity	Gender	10board	12board	Degree	Specialization	CollegeState
0	senior quality engineer	Bangalore	f	board ofsecondary education,ap	board of intermediate education,ap	B.Tech/B.E.	computer engineering	Andhra Pradesh
1	assistant manager	Indore	m	cbse	cbse	B.Tech/B.E.	electronics and communication engineering	Madhya Pradesh
2	systems engineer	Chennai	f	cbse	cbse	B.Tech/B.E.	information technology	Uttar Pradesh
3	senior software engineer	Gurgaon	m	cbse	cbse	B.Tech/B.E.	computer engineering	Delhi
4	get	Manesar	m	cbse	cbse	B.Tech/B.E.	electronics and communication engineering	Uttar Pradesh

In [ ]:
```
cat_df.info()
```

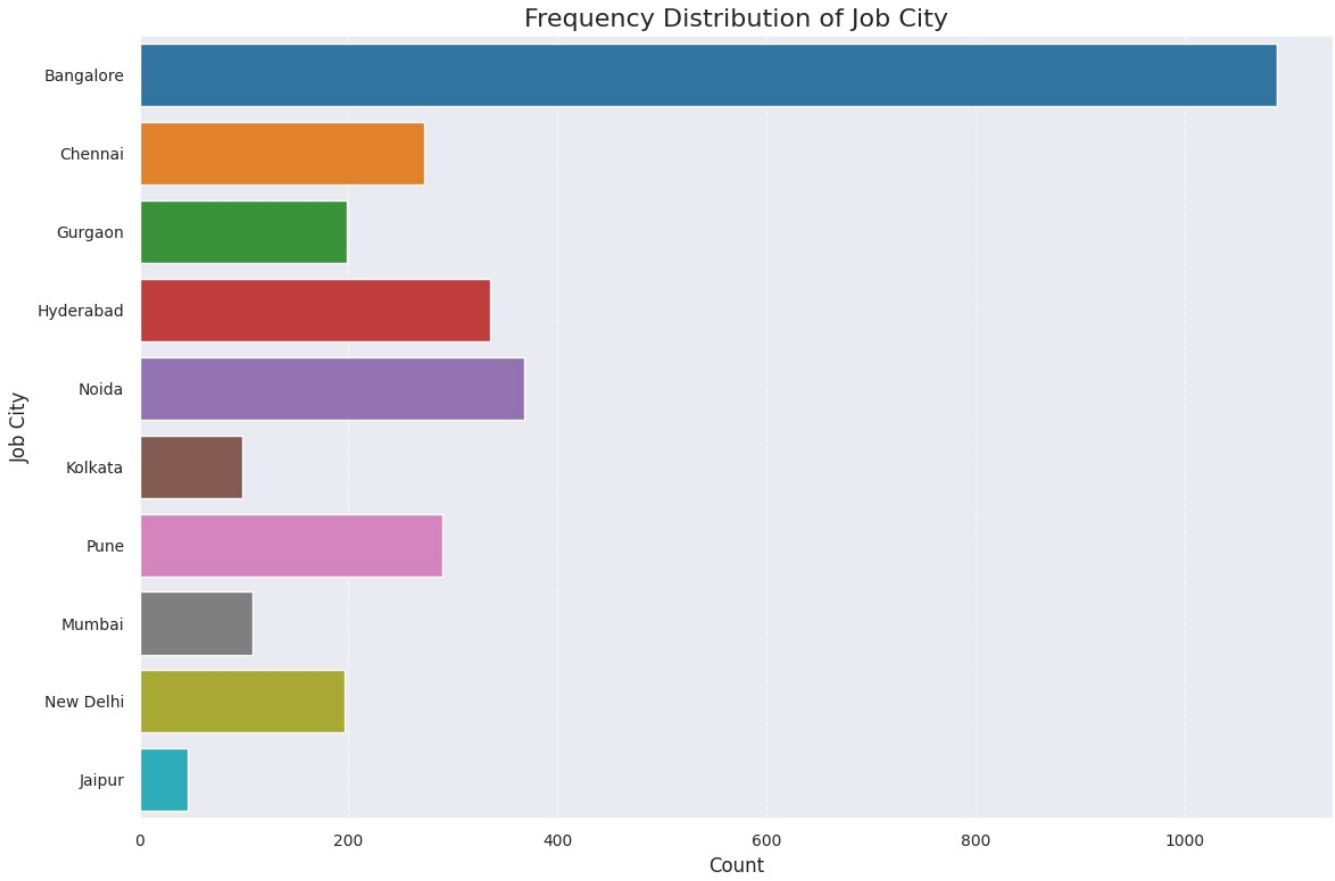
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3997 entries, 0 to 3997
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Designation     3997 non-null   object
 1   JobCity         3997 non-null   object
 2   Gender          3997 non-null   object
 3   10board         3997 non-null   object
 4   12board         3997 non-null   object
 5   Degree          3997 non-null   object
 6   Specialization  3997 non-null   object
 7   CollegeState    3997 non-null   object
dtypes: object(8)
memory usage: 410.1+ KB
```

In [ ]:
```python
#Frequency Distribution of Degree
plt.figure(figsize=(10, 6))
sns.countplot(data=cat_df, y='Degree', hue='Degree', palette='Set1', dodge=False, legend=False)
plt.title('Frequency Distribution of Degree', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Degree', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



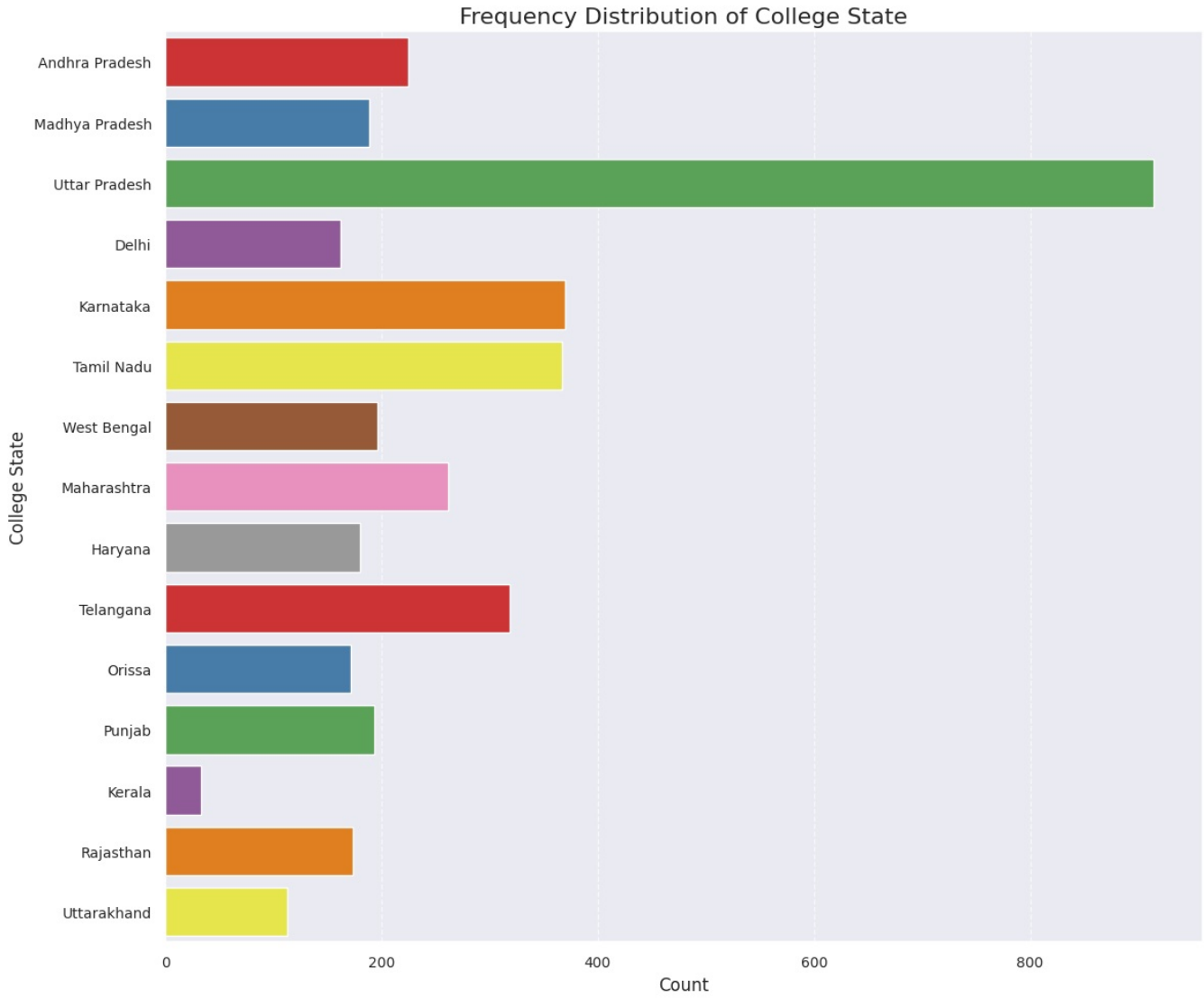
observations= Most students have a 'B.Tech' degree, followed by 'M.Tech' and 'MCA

In [ ]:
```python
#Frequency Distribution of Job City

top_categories = df['JobCity'].value_counts().head(10).index
df_top_categories = df[df['JobCity'].isin(top_categories)]

sns.set_theme(style="dark")
plt.figure(figsize=(12, 8))
sns.countplot(data=df_top_categories, y='JobCity', hue='JobCity', palette='tab10', dodge=False, legend=False)
plt.title('Frequency Distribution of Job City', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Job City', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



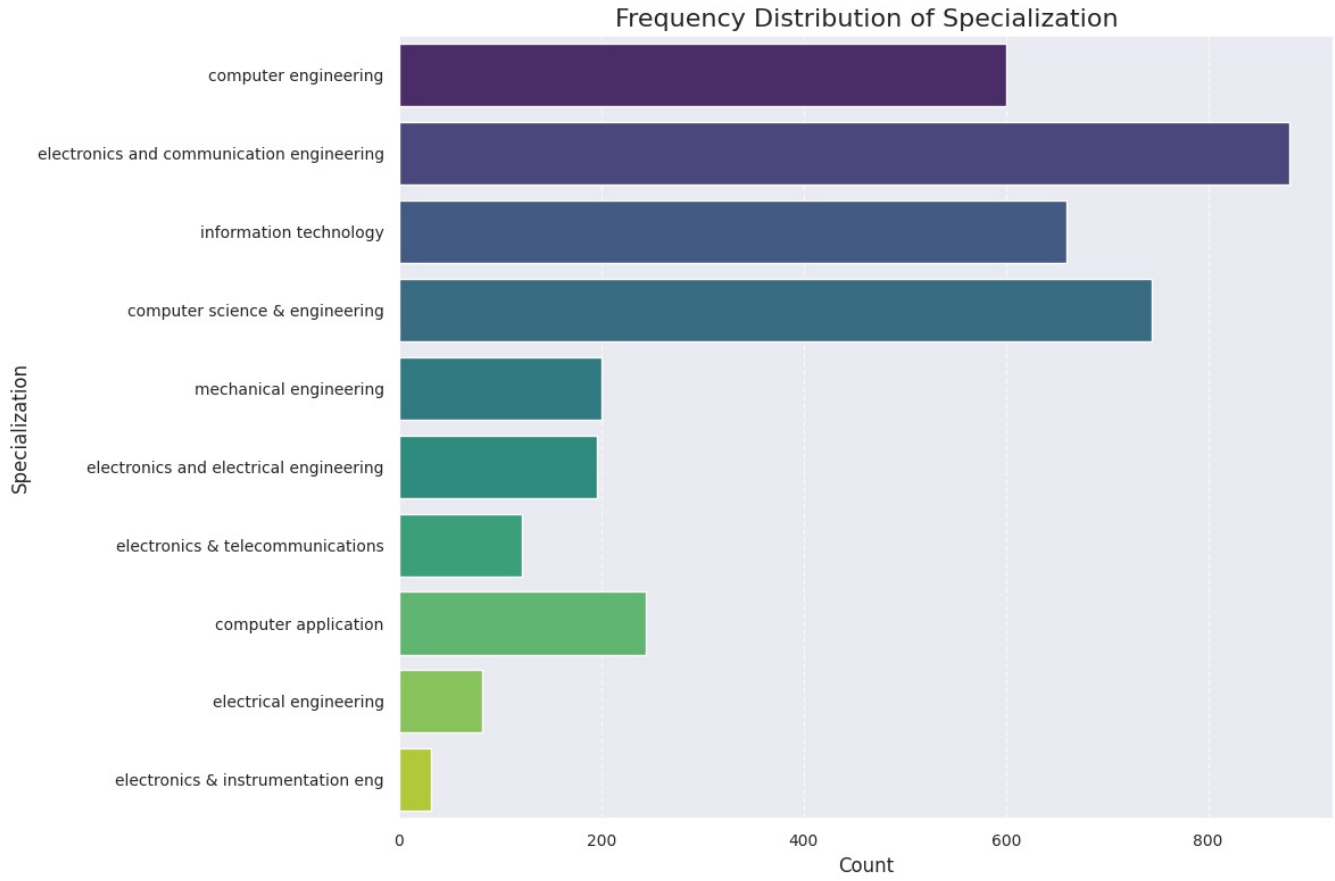
observations= Most students get jobs in 'Bangalore', followed by 'Noida' and 'Pune'.

In [ ]:
```
#Frequency Distribution of College State
top_CollegeState = cat_df['CollegeState'].value_counts().head(15).index
df_CollegeState = cat_df[cat_df['CollegeState'].isin(top_CollegeState)]

plt.figure(figsize=(12, 10))
sns.countplot(data=df_CollegeState, y='CollegeState', hue='CollegeState', palette='Set1', dodge=False, legend=F
plt.title('Frequency Distribution of College State', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('College State', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



observations:'Uttar Pradesh' has the highest number of students, followed by 'Karnataka' and 'Tamil Nadu'.

In [ ]:
```
#Frequency Distribution of Specialization

top_specializations = df['Specialization'].value_counts().head(10).index
df_top_specializations = df[df['Specialization'].isin(top_specializations)]

sns.set_theme(style="darkgrid")

plt.figure(figsize=(12, 8))
sns.countplot(data=df_top_specializations, y='Specialization', hue='Specialization', palette='viridis', dodge=F
plt.title('Frequency Distribution of Specialization', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Specialization', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



observations: Electronics and Communication Engineering' is the most common specialization.

Bi Variate Analysis

In [ ]:
```
df_copy.shape
```

Out[ ]:
```
(2577, 33)
```

In [ ]:
```
# Replace negative values with NaN
df_copy.loc[df['YearsExperience'] < 0, 'YearsExperience'] = np.nan
df.loc[df['YearsExperience'] < 0, 'YearsExperience'] = np.nan
```

In [ ]:
```
df_copy.head()
```

Out[ ]:

	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	10percentage	10board	12graduation	...	Domain	ComputerProgramn
0	203097	420000	2012-06-01	2024-02-26	senior quality engineer	Bangalore	f	84.3	board ofsecondary education,ap	2007	...	0.635979	445.000
1	579905	500000	2013-09-01	2024-02-26	assistant manager	Indore	m	85.4	cbse	2007	...	0.960603	451.301
2	810601	325000	2014-06-01	2024-02-26	systems engineer	Chennai	f	85.0	cbse	2010	...	0.450877	395.000
4	343523	200000	2014-03-01	2015-03-01	get	Manesar	m	78.0	cbse	2008	...	0.124502	451.301
9	1203363	230000	2014-07-01	2024-02-26	project engineer	Kolkata	m	77.0	cbse	2010	...	0.493596	385.000

5 rows × 33 columns

In [ ]:
```
df_copy.describe()
```

Out[ ]:

	ID	Salary	10percentage	12graduation	12percentage	CollegeTier	collegeGPA	CollegeCityID	CollegeCityTier	Gra
count	2.577000e+03	2577.000000	2577.000000	2577.000000	2577.000000	2577.000000	2577.000000	2577.000000	2577.000000	
mean	5.782980e+05	298115.638339	78.511137	2007.906092	75.076360	1.935972	71.817699	4877.030268	0.303066	
std	3.371018e+05	127974.903431	9.402766	1.605173	10.968103	0.244850	7.483492	4650.337731	0.459673	
min	1.124400e+04	35000.000000	50.600000	1998.000000	43.120000	1.000000	49.070000	11.000000	0.000000	
25%	3.137650e+05	200000.000000	72.150000	2007.000000	67.000000	2.000000	66.600000	462.000000	0.000000	
50%	5.369470e+05	300000.000000	79.800000	2008.000000	75.000000	2.000000	72.000000	3649.000000	0.000000	
75%	8.142300e+05	385000.000000	86.000000	2009.000000	83.000000	2.000000	76.600000	8346.000000	1.000000	
max	1.297877e+06	655000.000000	97.120000	2012.000000	98.200000	2.000000	99.930000	18409.000000	1.000000	

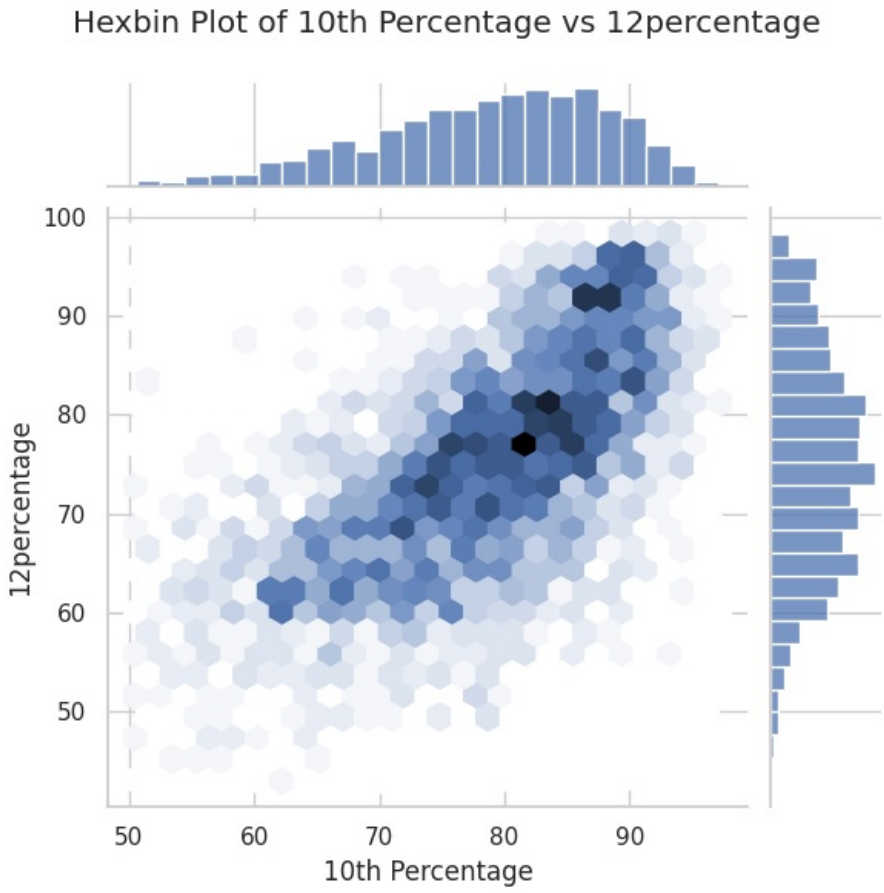
8 rows × 23 columns

In [ ]:
```
df_copy.dtypes
```

Out[ ]:
```
ID                          int64
Salary                      int64
DOJ                datetime64[ns]
DOL                datetime64[ns]
Designation                object
JobCity                    object
Gender                     object
10percentage              float64
10board                    object
12graduation                int64
12percentage              float64
12board                    object
CollegeTier                 int64
Degree                     object
Specialization             object
collegeGPA                float64
CollegeCityID               int64
CollegeCityTier             int64
CollegeState               object
GraduationYear              int64
English                     int64
Logical                     int64
Quant                       int64
Domain                    float64
ComputerProgramming       float64
ElectronicsAndSemicon       int64
ComputerScience             int64
conscientiousness         float64
agreeableness             float64
extraversion              float64
nueroticism               float64
openess_to_experience     float64
YearsExperience           float64
dtype: object
```

In [ ]:
```
# Plot of 10th Percentage vs 12percentage

plt.figure(figsize=(10, 8))
sns.jointplot(x='10percentage', y='12percentage', data=df_copy, kind='hex')
plt.suptitle('Hexbin Plot of 10th Percentage vs 12percentage')
plt.xlabel('10th Percentage')
plt.ylabel('12percentage')
plt.subplots_adjust(top=0.9)
plt.show()
```

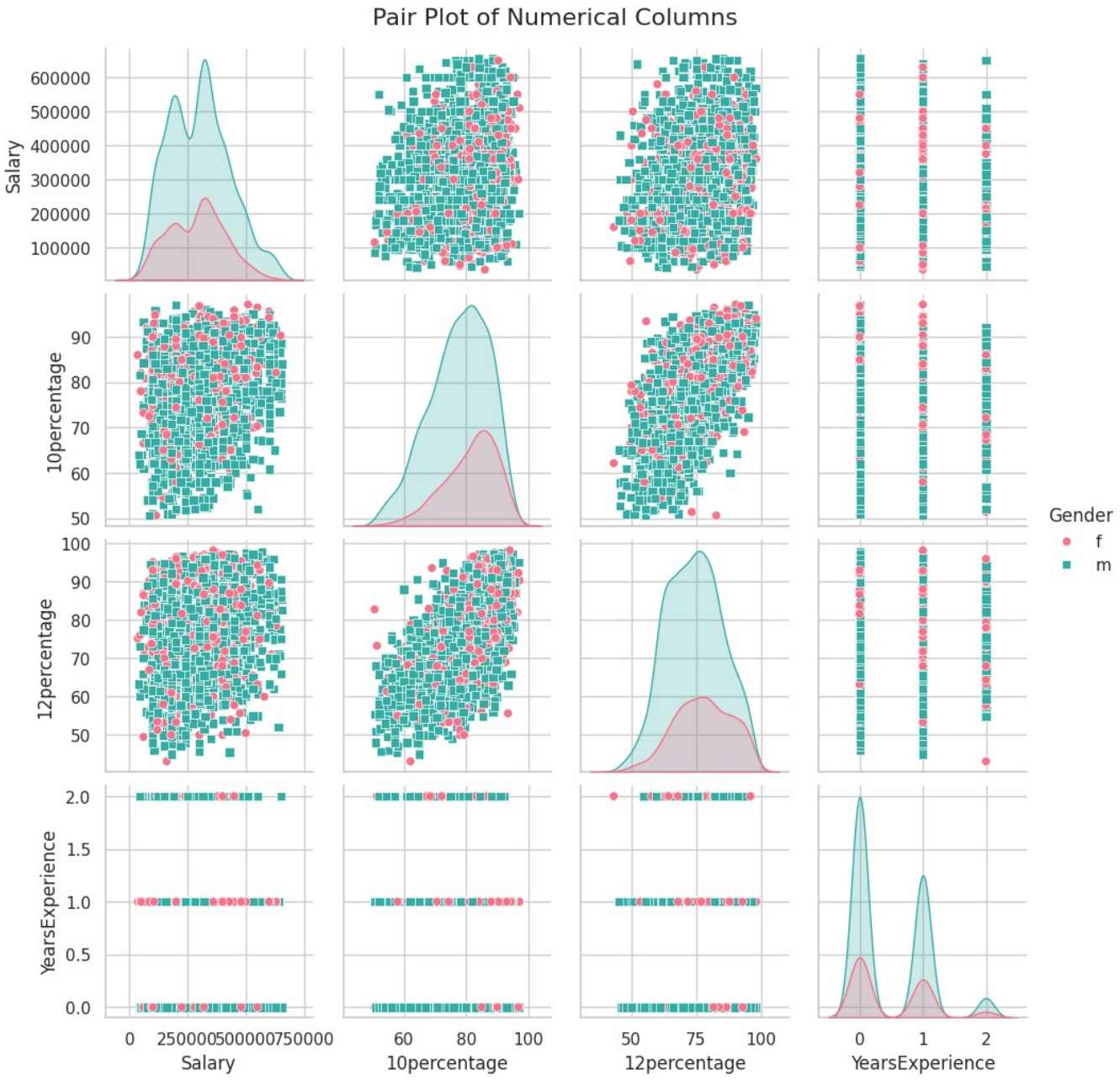
```
<Figure size 1000x800 with 0 Axes>
```



The hexbin plot indicates a positive correlation between 10th and 12th grade percentages, suggesting that students who perform well in 10th grade are likely to perform similarly in 12th grade.

In [ ]:
```
#Pair Plot of Numerical Column
custom_palette = sns.color_palette('husl', 2)
custom_markers = ['o', 's']
plt.figure(figsize=(12, 8))
sns.pairplot(df_copy, vars=['Salary', '10percentage', '12percentage', 'YearsExperience'],
             hue='Gender', palette=custom_palette, markers=custom_markers)
plt.suptitle('Pair Plot of Numerical Columns', fontsize=16, y=1.02)
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```

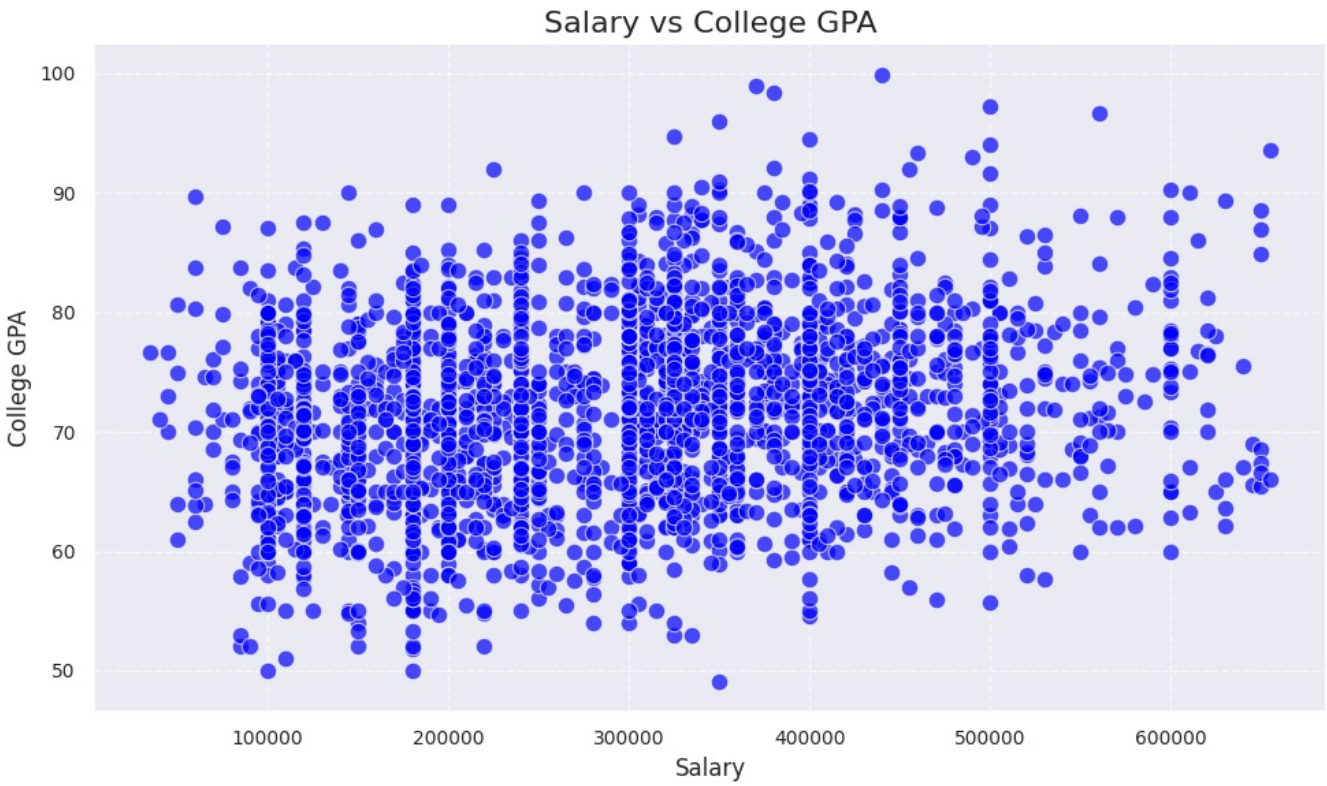


In [ ]:
```
# Plot between Gender and CollegeTier
plt.figure(figsize=(10, 8))
gender_college = df_copy.groupby(['Gender', 'CollegeTier']).size().unstack()
gender_college.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Stacked Bar Plot: Gender vs College Tier')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```

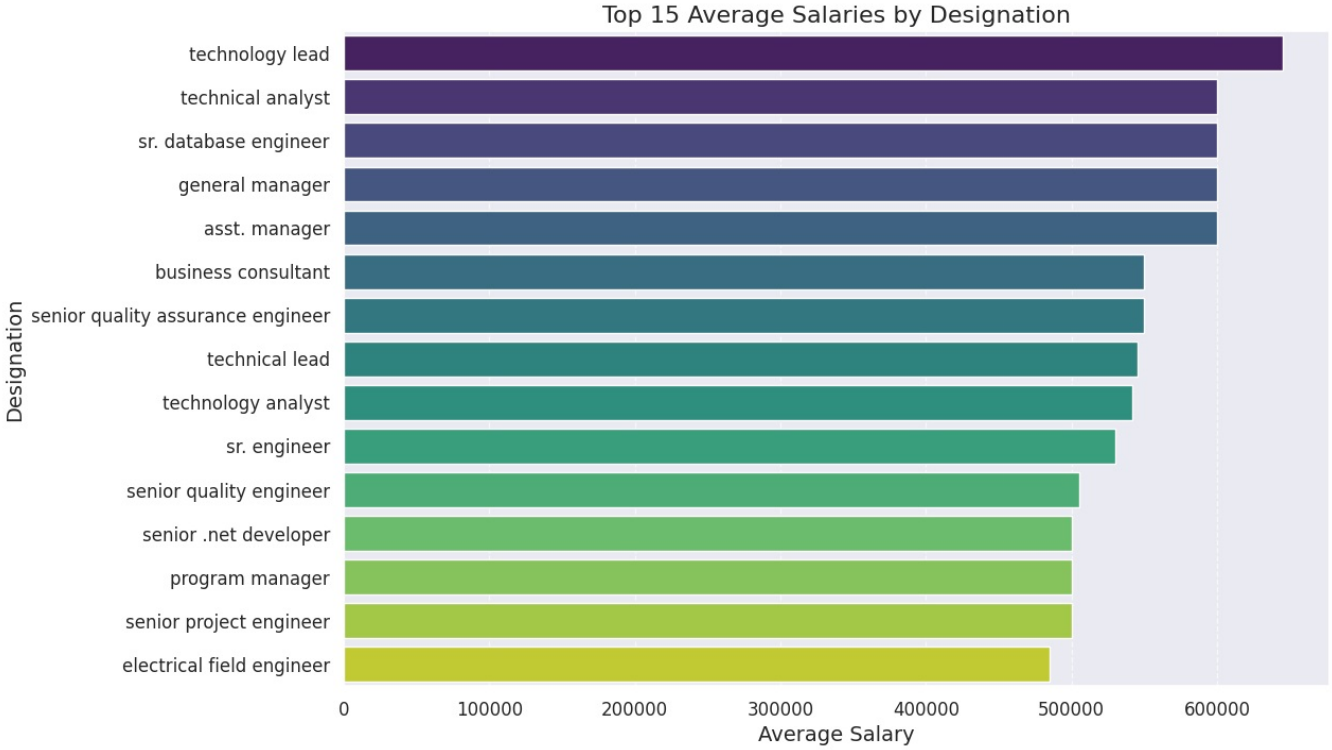
The stacked bar plot shows representation of students in college tiers by gender,indicating that there are more male students in tire 1 as compared to female students in tire 1

In [ ]:
```
#Plot of Salary vs College GPA

sns.set_theme(style="dark")

plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_copy, x='Salary', y='collegeGPA', color='blue', alpha=0.7, s=80)
plt.title('Salary vs College GPA', fontsize=16)
plt.xlabel('Salary', fontsize=12)
plt.ylabel('College GPA', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



observations:The scatter plot indicates a weak correlation between salary and college GPA, suggesting that a higher GPA does not necessarily equate to a higher salary.

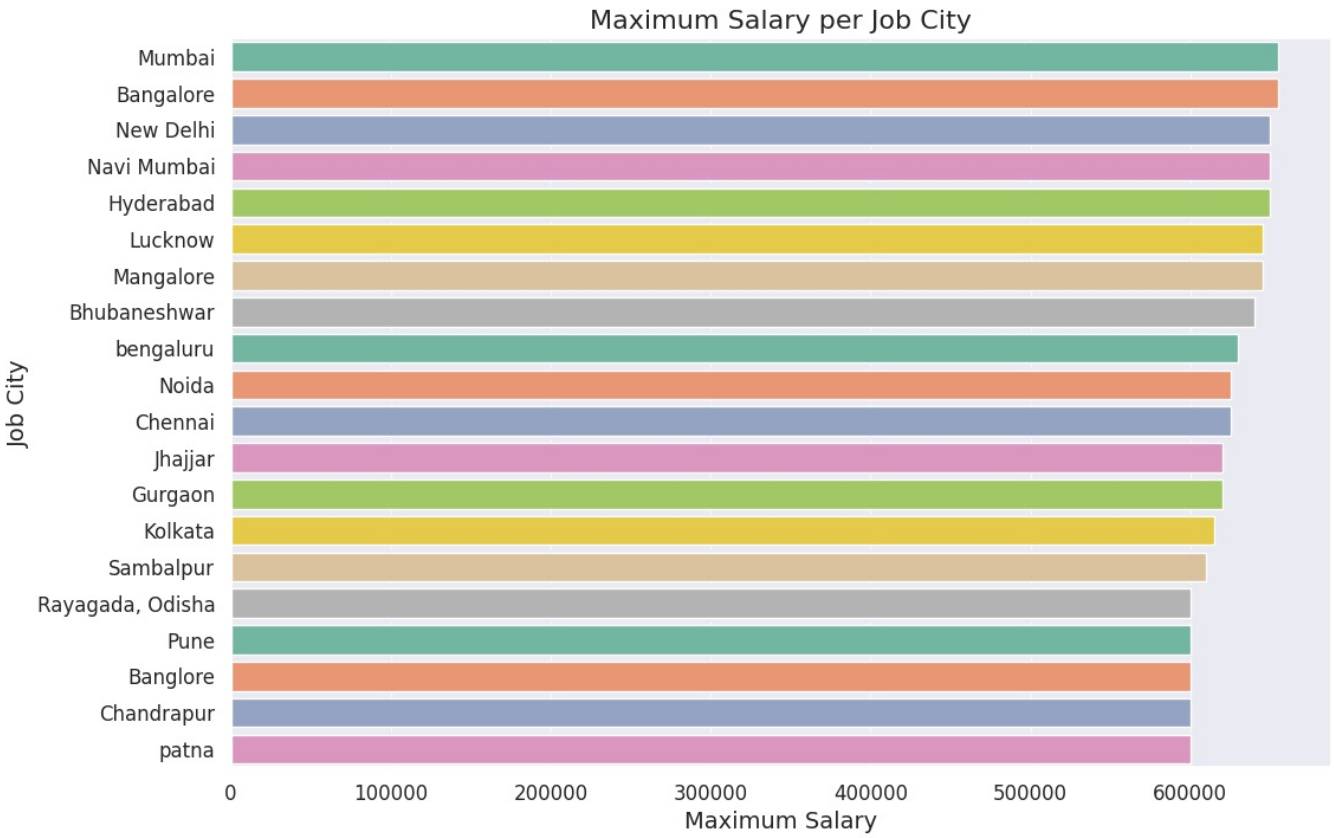
In [ ]:
```
# Average Salary for each Designation

salary_by_designation = df_copy.groupby('Designation')['Salary'].mean().reset_index()
salary_by_designation = salary_by_designation.sort_values(by='Salary', ascending=False)
top_15_designations = salary_by_designation.head(15)
colors = sns.color_palette('viridis', len(top_15_designations))

plt.figure(figsize=(12, 8))
sns.barplot(x='Salary', y='Designation', data=top_15_designations, palette=colors, hue='Designation', dodge=Fal
plt.title('Top 15 Average Salaries by Designation', fontsize=16)
plt.xlabel('Average Salary', fontsize=14)
plt.ylabel('Designation', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



The bar chart indicates that among the top 15 designations listed, the 'Technology Lead' has the highest average salary

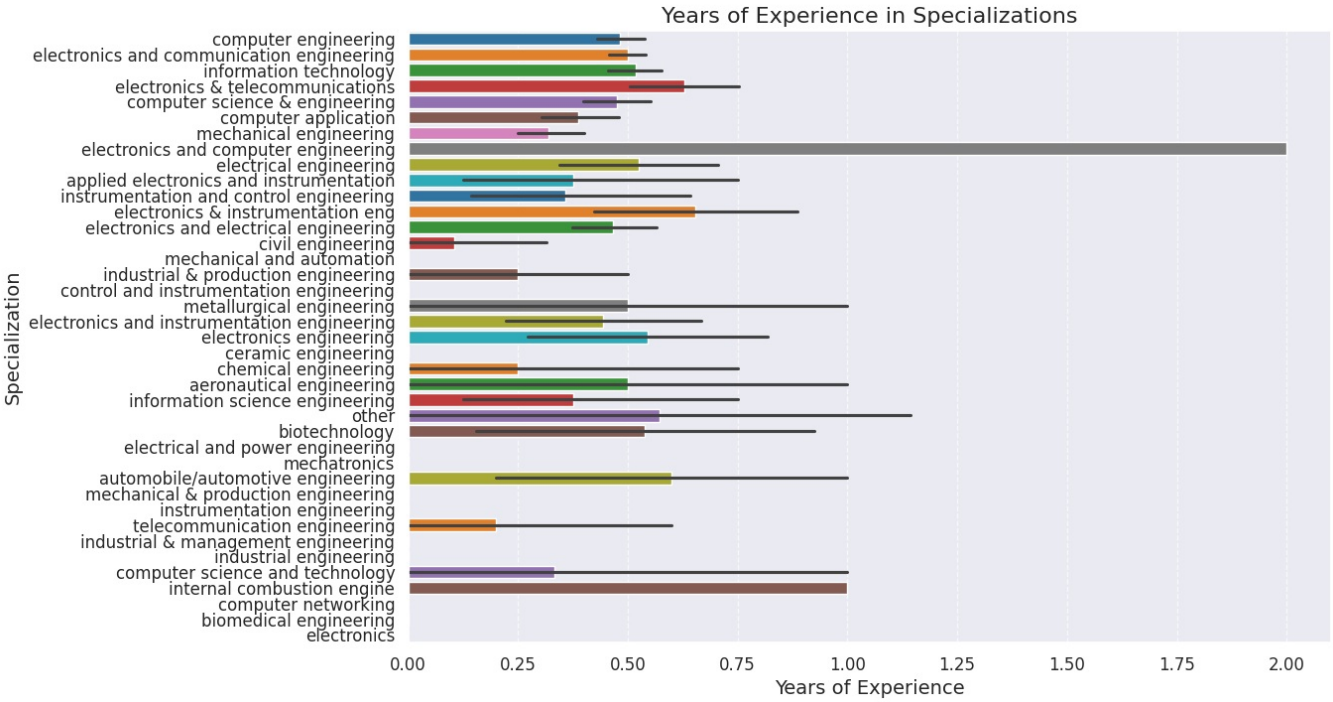
In [ ]:
```
#plot of Maximum Salary per Job City
```

In [ ]:
```
#plot of Maximum Salary per Job City

# Calculate maximum Salary for each Job City
max_salary_per_city = df_copy.groupby('JobCity')['Salary'].max().reset_index()
top_15_cities = max_salary_per_city.sort_values(by='Salary', ascending=False).head(20)

plt.figure(figsize=(12, 8))
sns.barplot(x='Salary', y='JobCity', data=top_15_cities, hue='JobCity', dodge=False, palette='Set2', legend=Fal
plt.title('', fontsize=16)
plt.xlabel('Maximum Salary', fontsize=14)
plt.ylabel('Job City', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



The bar chart indicates a comparison of the maximum salaries offered in various cities, with Mumbai offering the highest maximum salary. This could be useful in understanding the salary distribution across different cities in India.

In [ ]:
```
sns.set_theme(style="dark")
plt.figure(figsize=(12, 8))
sns.barplot(x='YearsExperience', y='Specialization', data=df_copy, palette='tab10', hue='Specialization', dodge
plt.title('Years of Experience in Specializations', fontsize=16)
plt.xlabel('Years of Experience', fontsize=14)
plt.ylabel('Specialization', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



The plot illustrates the years of experience in various engineering specializations, showing a diverse range of expertise across different fields. This could be useful in understanding the distribution of experience in these specializations.

In [ ]:
```
# Plot of salary vs years of experience( with outliers)
sns.set_theme(style="darkgrid")

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='YearsExperience', y='Salary', color='red')
plt.title('Salary vs Years of Experience', fontsize=16)
plt.xlabel('Years of Experience', fontsize=14)
plt.ylabel('Salary', fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

observations: The salary vs years of experience plot you sent me shows a positive correlation between salary and experience, with a few outliers on the high end.

In [ ]:
```
# The scatter plot of salary vs years of experience (Without Outliers)
sns.set_theme(style="darkgrid")
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_copy, x='YearsExperience', y='Salary', color='red')
plt.title('Salary vs Years of Experience', fontsize=16)
plt.xlabel('Years of Experience', fontsize=14)
plt.ylabel('Salary', fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='both', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

observations : The data points tend to move upwards from left to right, indicating a positive correlation. This means that as people gain more experience (years on the x-axis), their salaries tend to increase (salary on the y-axis).

Conclusion: When outliers are present, the plot suggests a positive correlation with a few high-earning individuals. When outliers are removed, the plot confirms a stronger positive correlation between salary and experience.

Research Question: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 given to you

In [ ]:
```
df_S =df[(df['Specialization']== 'computer science & engineering')]
df_S
```

Out[ ]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	Domain	ComputerProgra
6	train	947847	300000	2014-08-01	2015-05-01 00:00:00	java software engineer	Banglore	m	1993-02-01	86.08	...	0.356536	405
18	train	711342	120000	2014-01-01	2014-06-01 00:00:00	data entry operator	Gurgaon	m	1992-12-07	65.00	...	0.563268	425
24	train	963123	335000	2014-06-01	2015-06-01 00:00:00	programmer analyst	Hyderabad	m	1993-06-28	88.00	...	0.356536	475
25	train	350211	435000	2012-09-01	present	systems analyst	Gurgaon	f	1991-03-02	86.80	...	0.744758	565
31	train	1094324	340000	2014-08-01	2015-04-01 00:00:00	software engineer	Bangalore	m	1992-10-23	77.20	...	0.622643	485
...	...	...	...	...	...	...	...	...	...	...	...	...	
3969	train	1233826	330000	2015-06-01	present	technical engineer	pune	m	1993-01-24	76.00	...	0.609525	451
3975	train	1240207	300000	2014-07-01	2015-04-01 00:00:00	game developer	Noida	m	1991-06-03	86.00	...	0.968237	605
3981	train	1077872	220000	2014-09-01	present	software engineer	Gurgaon	m	1991-12-17	53.40	...	0.953900	575
3989	train	1204604	300000	2014-09-01	present	software engineer	Bangalore	m	1991-11-23	74.88	...	0.356536	465
3996	train	947111	200000	2014-07-01	2015-01-01 00:00:00	software developer	Asifabadbanglore	f	1992-03-20	78.72	...	0.744758	445

744 rows × 36 columns

In [ ]:
```
df_R=df_S[(df_S["Designation"]=="programmer analyst")|(df_S["Designation"]=="software engineer")|(df_S["Designa
       |(df_S["Designation"]=="associate engineer")]
df_R
```

Out[ ]:

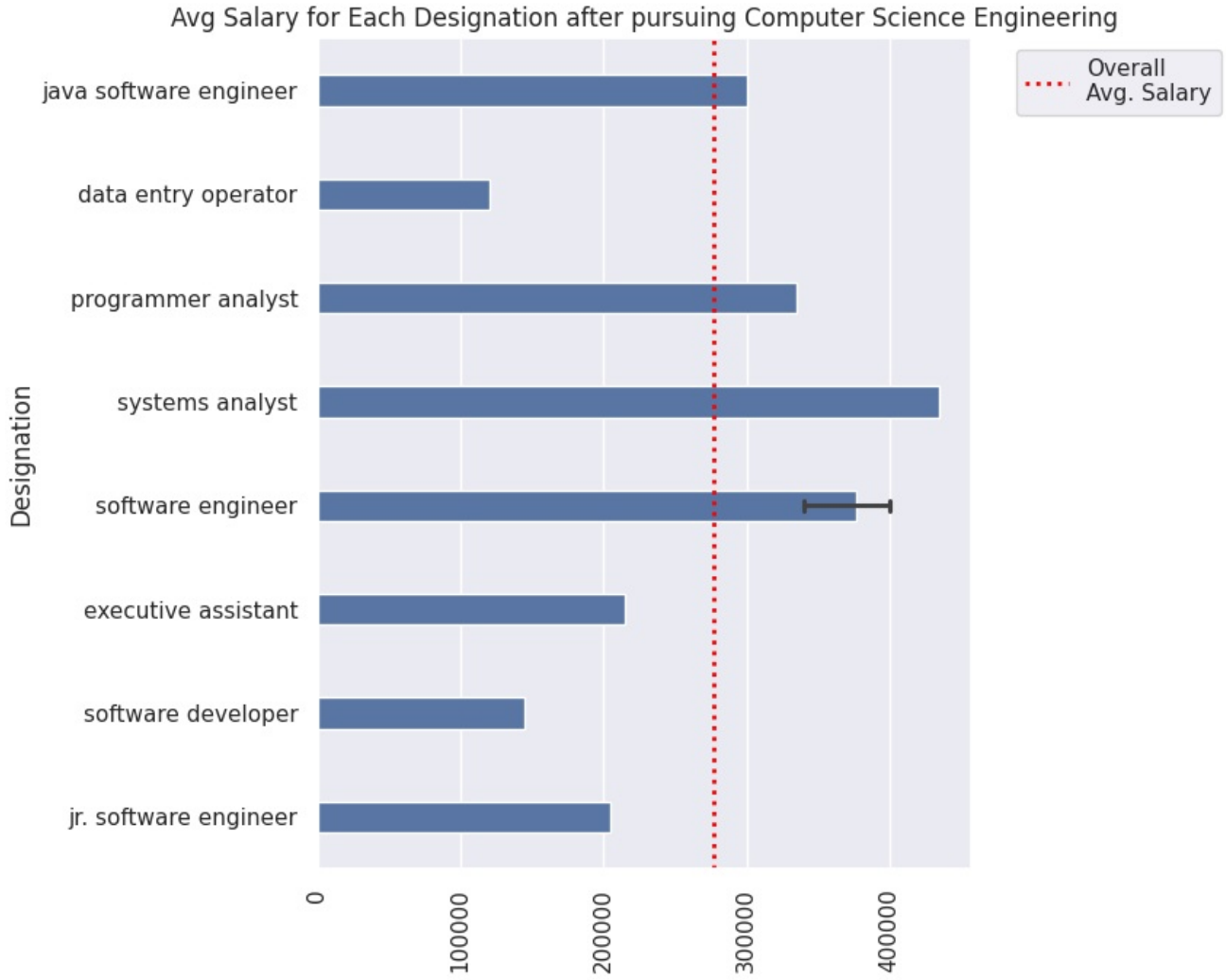
	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	Domain	ComputerProgramm
24	train	963123	335000	2014-06-01	2015-06-01 00:00:00	programmer analyst	Hyderabad	m	1993-06-28	88.00	...	0.356536	47
31	train	1094324	340000	2014-08-01	2015-04-01 00:00:00	software engineer	Bangalore	m	1992-10-23	77.20	...	0.622643	48
48	train	338428	390000	2013-09-01	present	software engineer	Bangalore	m	1991-02-28	86.60	...	0.356536	47
52	train	794209	400000	2015-04-01	present	software engineer	Navi Mumbai	m	1992-03-09	85.20	...	0.600057	43
55	train	989860	250000	2014-08-01	present	software engineer	Mangalore	m	1992-02-13	90.80	...	0.486747	48
...	...	...	...	...	...	...	...	...	...	...	...	...	
3917	train	638674	105000	2014-10-01	2015-04-01 00:00:00	software engineer	Kochi/Cochin	f	1991-12-14	93.00	...	0.670743	45
3939	train	716325	100000	2013-07-01	2014-12-01 00:00:00	software engineer	Hyderabad	m	1992-07-05	65.00	...	0.377551	37
3959	train	775902	390000	2014-01-01	2015-04-01 00:00:00	software engineer	Gurgaon	m	1991-09-30	89.60	...	0.842248	54
3981	train	1077872	220000	2014-09-01	present	software engineer	Gurgaon	m	1991-12-17	53.40	...	0.953900	57
3989	train	1204604	300000	2014-09-01	present	software engineer	Bangalore	m	1991-11-23	74.88	...	0.356536	46

167 rows × 36 columns

In [ ]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.barplot(x='Salary', y='Designation',data=df_S.head(10),capsize=0.1,width=0.3, ax=ax)
ax.axvline(df_S['Salary'].mean(), color='red',linestyle=':',linewidth=2, label='Overall\nAvg. Salary')

ax.set_title('Avg Salary for Each Designation after pursuing Computer Science Engineering')
ax.legend(loc='upper right', bbox_to_anchor=(1.4, 1))
```

```
ax.set_xlabel('')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)

plt.tight_layout()
plt.show()
```

```
<ipython-input-81-6d8a657cdf29>:8: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```



The bar chart indicates that among the listed designations for Computer Science Engineering graduates, Java Software Engineers receive the highest average salary, while Data Entry Operators are paid the least

In [ ]:
```
roles = ["Programming Analyst", "Software Engineer", "Hardware Engineer", "Associate Engineer"]
df_copy= df_copy[df_copy['Designation'].isin(roles) & (df_copy['YearsExperience'] == 0)]

# Calculate the average salary
average_salary =df_copy['Salary'].mean()

print(f"The average salary for fresh graduates in the specified roles is: {average_salary}")
```

```
The average salary for fresh graduates in the specified roles is: nan
```

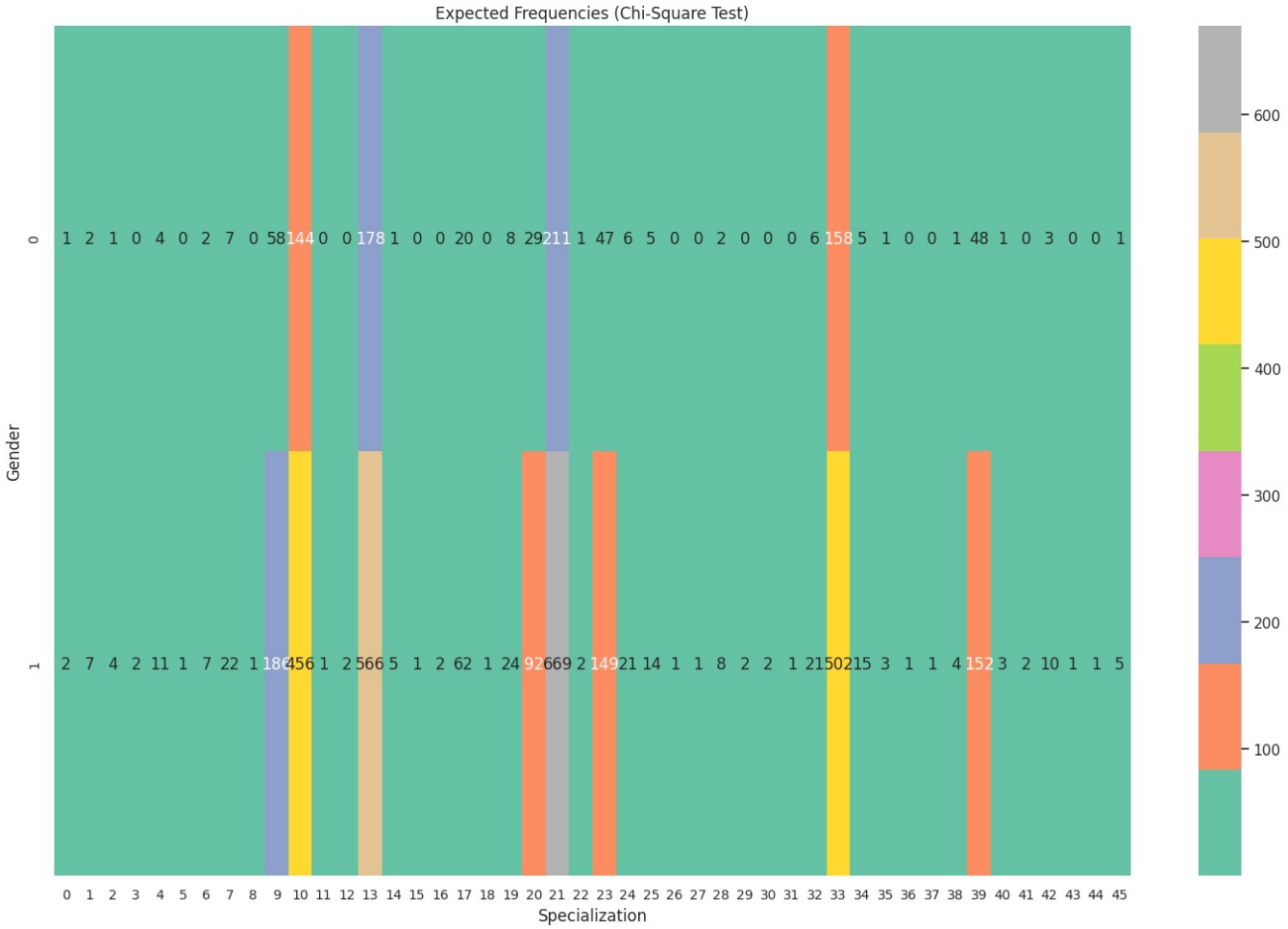
conclusion: The average salary for fresh graduates in the specified roles is not available. Therefore, with the current data, we cannot conclusively verify or refute the claim made in the Times of India article dated Jan 18, 2019. It would be beneficial to have more complete data to draw a more accurate conclusiondated 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.

Q2. Is there a relationship between gender and specialization? (i.e. Does the preference of Specialisation depend on the Gender?)

In [ ]:
```
from scipy.stats import chi2_contingency

#  create a cross-tabulation table
table = pd.crosstab(df['Gender'], df['Specialization'])

# Perform the Chi-Square Test of Independence
chi2, p, dof, expected = chi2_contingency(table)

# Print the results
print(f"Chi-Square Statistic: {chi2}")
print(f"P-value: {p}")
```

```
Chi-Square Statistic: 104.14994896714552
P-value: 1.3704397838344918e-06
```

In [ ]:
```python
# Heatmap of the expected frequencies
plt.figure(figsize=(15, 10))
sns.heatmap(expected, annot=True, fmt='.0f', cmap='Set2')
plt.title('Expected Frequencies (Chi-Square Test)')
plt.xlabel('Specialization')
plt.ylabel('Gender')
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```



Conclusion: Based on the Chi-Square Test, this suggests that there is a statistically significant relationship between gender and specialization.In other words, the preference for specialization appears to depend on gender in our dataset