Exploratory Data Analysis

A detailed data description and objective

objective -> is to understand the relationships among features and the spread of the data

The dataset was released by Aspiring Minds from the Aspiring Mind Employment Outcome 2015 (AMEO). The study is primarily limited only to students with engineering disciplines. 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. The dataset also contains demographic features. The dataset contains around 40 independent variables and 4000 data points. 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.

VARIABLES	TYPE	Description
ID	UID	A unique ID to identify a candidate
Salary	Continuous	Annual CTC offered to the candidate (in INR)
DOJ	Date	Date of joining the company
DOL	Date	Date of leaving the company
Designation	Categorical	Designation offered in the job
JobCity	Categorical	Location of the job (city)
Gender	Categorical	Candidate's gender
DOB	Date	Date of birth of candidate
10percentage	Continuous	Overall marks obtained in grade 10 examinations

10board	Continuous	The school board whose curriculum the candidate followed in grade 10
12graduation	Date	Year of graduation - senior year high school
12percentage	Continuous	Overall marks obtained in grade 12 examinations
12board	Date	The school board whose curriculum the candidate followed in grade 12
CollegeID	NA/ID	Unique ID identifying the college which the candidate attended
CollegeTier	Categorical	Tier of college
Degree	Categorical	Degree obtained/pursued by the candidate
Specialization	Categorical	Specialization pursued by the candidate
CollegeGPA	Continuous	Aggregate GPA at graduation
CollegeCityID	NA/ID	A unique ID to identify the city in which the college is located in
CollegeCityTier	Categorical	The tier of the city in which the college is located
CollegeState	Categorical	Name of States

GraduationYear	Date	Year of graduation (Bachelor’s degree)
English	Continuous	Scores in AMCAT English section
Logical	Continuous	Scores in AMCAT Logical section
Quant	Continuous	Scores in AMCAT Quantitative section
Domain	Continuous/ Standardized	Scores in AMCAT’s domain module
ComputerProgramming	Continuous	Score in AMCAT’s Computer programming section
ElectronicsAndSemicon	Continuous	Score in AMCAT’s Electronics & Semiconductor Engineering section
ComputerScience	Continuous	Score in AMCAT’s Computer Science section
MechanicalEngg	Continuous	Score in AMCAT’s Mechanical Engineering section
ElectricalEngg	Continuous	Score in AMCAT’s Electrical Engineering section

TelecomEngg	Continuous	Score in AMCAT's Telecommunication Engineering section
CivilEngg	Continuous	Score in AMCAT's Civil Engineering section
conscientiousness	Continuous/ Standardized	Scores in one of the sections of AMCAT's personality test
agreeableness	Continuous/ Standardized	Scores in one of the sections of AMCAT's personality test
extraversion	Continuous/ Standardized	Scores in one of the sections of AMCAT's personality test
neuroticism	Continuous/ Standardized	Scores in one of the sections of AMCAT's personality test
openess_to_experience	Continuous/ Standardized	Scores in one of the sections of AMCAT's personality test

Import the data and display the head, shape and description of the data.

```
import pandas as pd
import numpy as np
df= pd.read_csv("/content/aspiring_minds_employability_outcomes_2015.csv")
df.head()
```

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	
0	train	203097	420000.0	6/1/2012 0:00	present	senior quality engineer	Bangalore	f	2/19
1	train	579905	500000.0	9/1/2013 0:00	present	assistant manager	Indore	m	10/4
2	train	810601	325000.0	6/1/2014 0:00	present	systems engineer	Chennai	f	8/3
3	train	267447	1100000.0	7/1/2011 0:00	present	senior software	Gurgaon	m	12/5

```
df.shape
```

```
(3998, 39)
```

5 rows × 39 columns

```
df.describe()
```

	ID	Salary	10percentage	12graduation	12percentage	Colle
count	3.998000e+03	3.998000e+03	3998.000000	3998.000000	3998.000000	3998.00
mean	6.637945e+05	3.076998e+05	77.925443	2008.087544	74.466366	5156.85
std	3.632182e+05	2.127375e+05	9.850162	1.653599	10.999933	4802.26
min	1.124400e+04	3.500000e+04	43.000000	1995.000000	40.000000	2.00
25%	3.342842e+05	1.800000e+05	71.680000	2007.000000	66.000000	494.00
50%	6.396000e+05	3.000000e+05	79.150000	2008.000000	74.400000	3879.00
75%	9.904800e+05	3.700000e+05	85.670000	2009.000000	82.600000	8818.00
max	1.298275e+06	4.000000e+06	97.760000	2013.000000	98.700000	18409.00

8 rows × 27 columns

```
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   float64
 3   DOJ                    3998 non-null   object
 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   object 
 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: float64(10), int64(17), object(12)
memory usage: 1.2+ MB
```

Univariate Analysis -> PDF, Histograms, Boxplots, Countplots, etc..

- Find the outliers in each numerical column
- Understand the probability and frequency distribution of each numerical column
- Understand the frequency distribution of each categorical Variable/Column
- Mention observations after each plot.

```
# outliers in each numerical column
num_df= df.select_dtypes(include=['int64','float64'])
num_df.columns
```

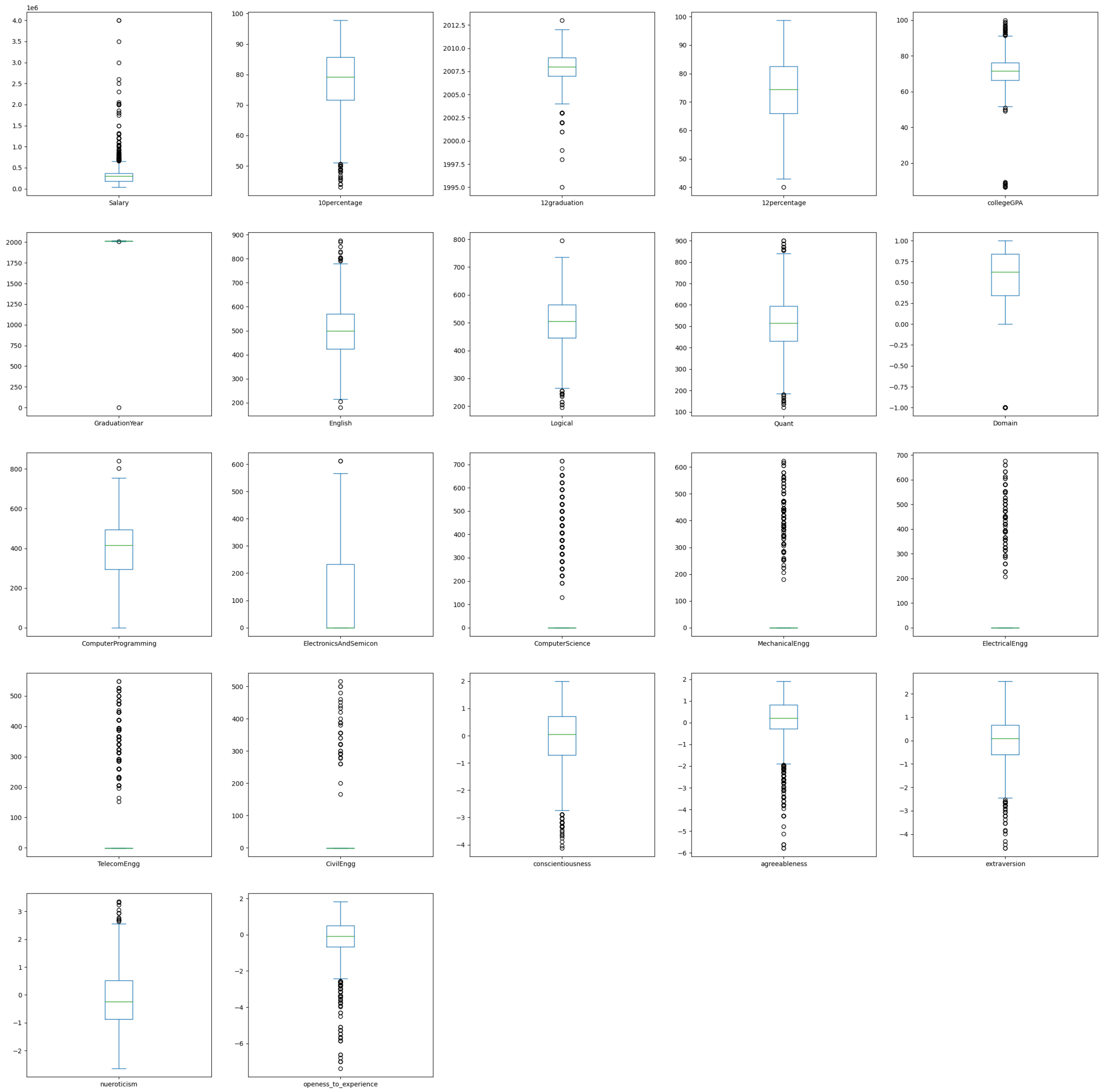
```
Index(['ID', 'Salary', '10percentage', '12graduation', '12percentage',
       'CollegeID', 'CollegeTier', 'collegeGPA', 'CollegeCityID',
       'CollegeCityTier', 'GraduationYear', 'English', 'Logical', 'Quant',
       'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon',
       'ComputerScience',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

```
#to remove ID,CollegeID,CollegeTier,CollegeCityID,CollegeCityTier columns as we wont check
num_df= num_df.drop(columns=['ID','CollegeID','CollegeTier','CollegeCityID','CollegeCityTi
num_df.columns
```

```
Index(['Salary', '10percentage', '12graduation', '12percentage', 'collegeGPA',
       'GraduationYear', 'English', 'Logical', 'Quant', 'Domain',
       'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience',
       'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg',
       'conscientiousness', 'agreeableness', 'extraversion', 'nueroticism',
       'openess_to_experience'],
      dtype='object')
```

```
num_df.plot(kind='box',subplots=True,layout=(5,5),figsize=(30,30))
```

```
Salary                           Axes(0.125,0.747241;0.133621x0.132759)
10percentage                  Axes(0.285345,0.747241;0.133621x0.132759)
12graduation                   Axes(0.44569,0.747241;0.133621x0.132759)
12percentage                  Axes(0.606034,0.747241;0.133621x0.132759)
collegeGPA                    Axes(0.766379,0.747241;0.133621x0.132759)
GraduationYear                   Axes(0.125,0.587931;0.133621x0.132759)
English                       Axes(0.285345,0.587931;0.133621x0.132759)
Logical                        Axes(0.44569,0.587931;0.133621x0.132759)
Quant                         Axes(0.606034,0.587931;0.133621x0.132759)
Domain                        Axes(0.766379,0.587931;0.133621x0.132759)
ComputerProgramming              Axes(0.125,0.428621;0.133621x0.132759)
ElectronicsAndSemicon         Axes(0.285345,0.428621;0.133621x0.132759)
ComputerScience                Axes(0.44569,0.428621;0.133621x0.132759)
MechanicalEngg                Axes(0.606034,0.428621;0.133621x0.132759)
ElectricalEngg                Axes(0.766379,0.428621;0.133621x0.132759)
TelecomEngg                       Axes(0.125,0.26931;0.133621x0.132759)
CivilEngg                      Axes(0.285345,0.26931;0.133621x0.132759)
conscientiousness               Axes(0.44569,0.26931;0.133621x0.132759)
agreeableness                  Axes(0.606034,0.26931;0.133621x0.132759)
extraversion                   Axes(0.766379,0.26931;0.133621x0.132759)
nueroticism                          Axes(0.125,0.11;0.133621x0.132759)
openess_to_experience             Axes(0.285345,0.11;0.133621x0.132759)
dtype: object
```



```
#lets check if we got some null values in these column
num_df.isnull().sum().sort_values(ascending=True)
```

```
Salary                    0
extraversion              0
agreeableness             0
conscientiousness         0
CivilEngg                 0
TelecomEngg               0
ElectricalEngg            0
MechanicalEngg            0
ComputerScience           0
ElectronicsAndSemicon     0
ComputerProgramming       0
Domain                    0
Quant                     0
Logical                   0
English                   0
GraduationYear            0
collegeGPA                0
12percentage              0
12graduation              0
10percentage              0
nueroticism               0
openess_to_experience     0
dtype: int64
```

Salary column

```
df['Salary'].mean()
```

```
307699.8499249625
```

```
df['Salary'].median()
```

```
300000.0
```

mean and median are that close there is a gap of 7,699 which means we may have some outliers

```
print(df['Salary'].quantile(0.25))
```

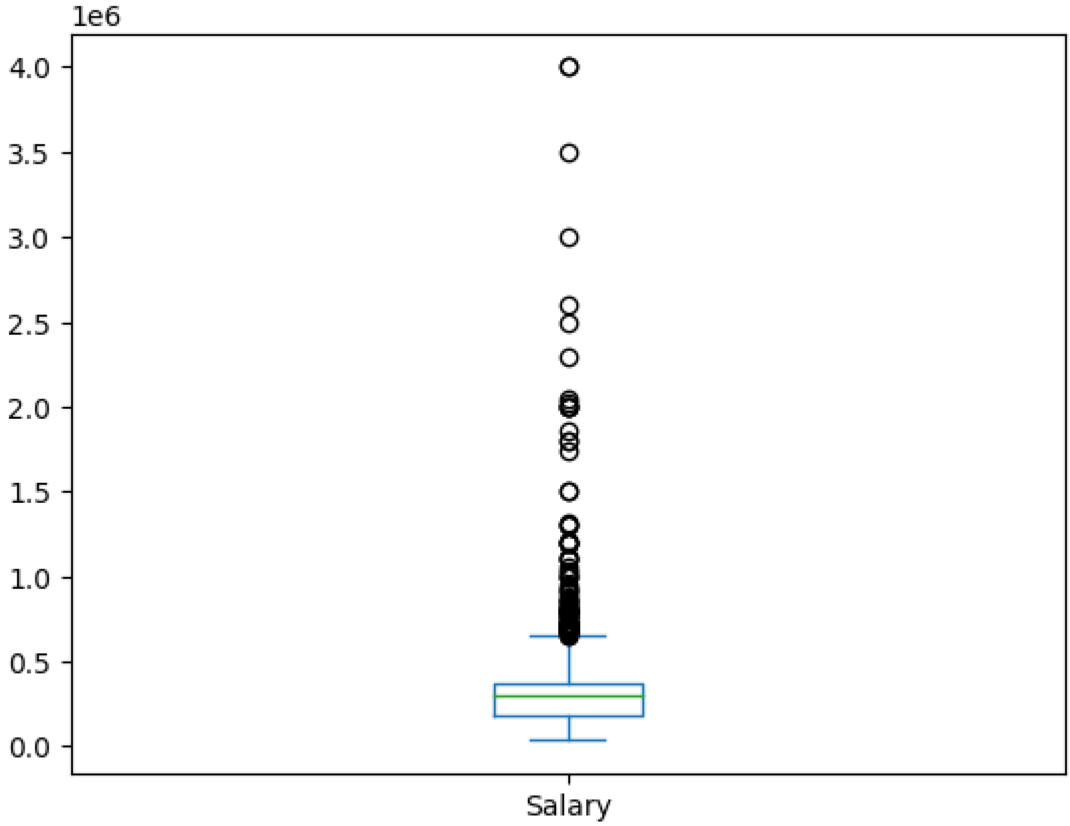
```
180000.0
```

```
print(df['Salary'].quantile(0.75))
```

```
370000.0
```

```
num_df['Salary'].plot(kind='box')
```

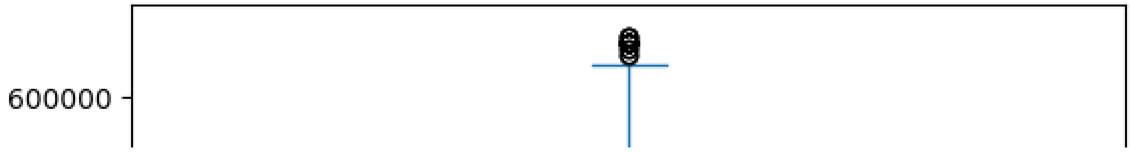
```
&lt;Axes: &gt;
```



we have got many outliers here

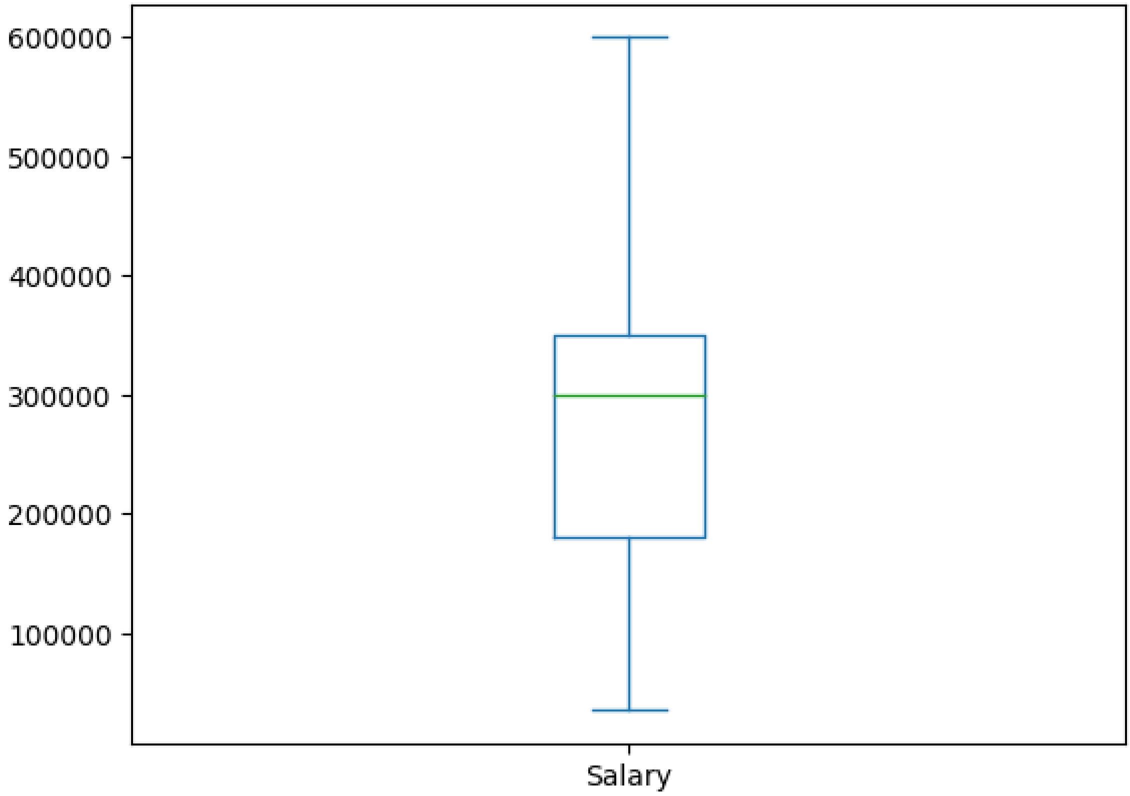
```
#removing outliers using IQR
q1= df['Salary'].quantile(0.25)
q3= df['Salary'].quantile(0.75)
IQR=q3-q1
Salary_lower_bound=q1-1.5*IQR
Salary_upper_bound=q3+1.5*IQR
clean_df=df[(df['Salary']>=Salary_lower_bound)& (df['Salary']<=Salary_upper_bound)]
clean_df['Salary'].plot(kind='box')
```

<Axes: >



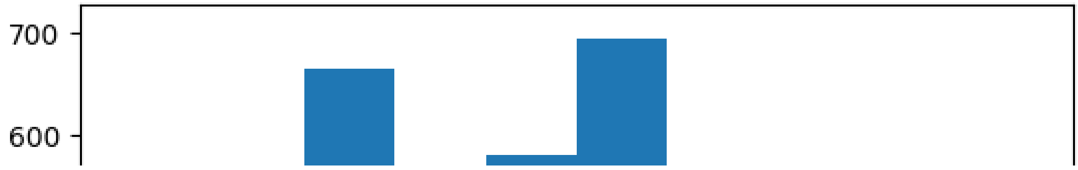

```
# we can still see some outliers above 600000 lets remove them
clean_df=df[df.Salary<=600000]
clean_df['Salary'].plot(kind='box')
```

<Axes: >



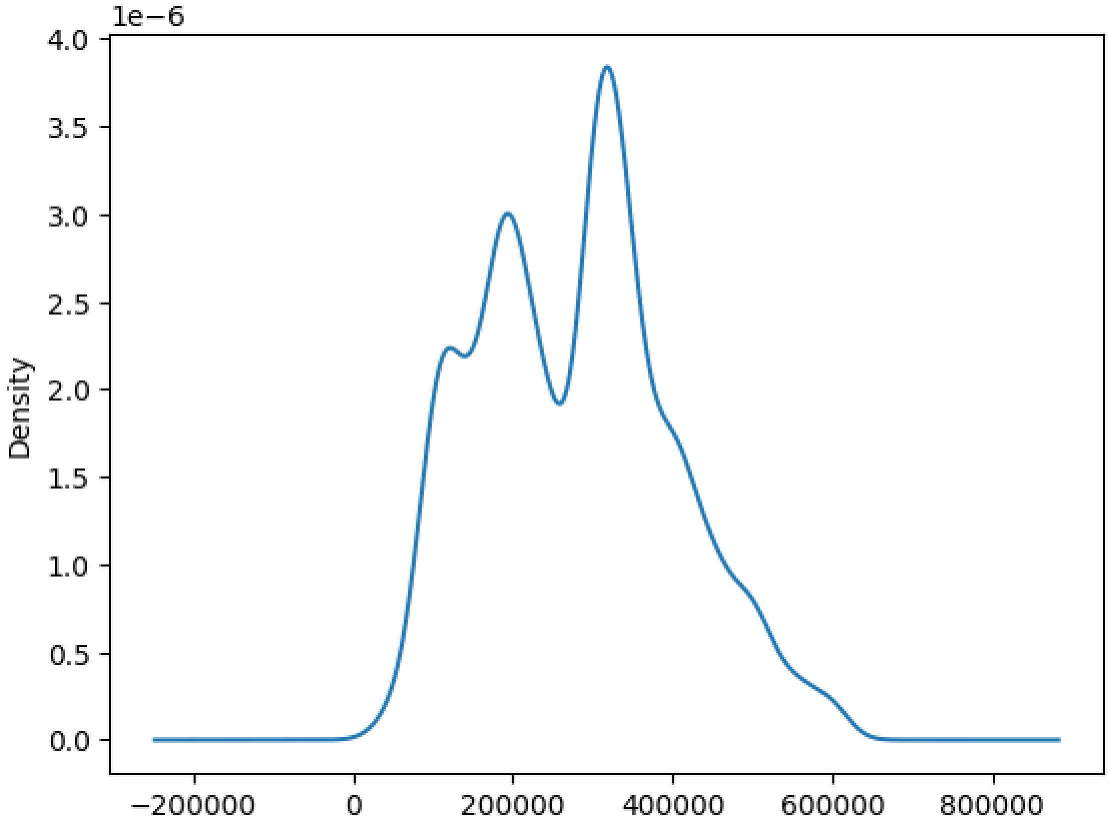

```
#normal distribution of Salary via hist
clean_df['Salary'].plot(kind='hist')
```

<Axes: ylabel='Frequency'>




```
#normal distribution of Salary via kernel distribution
clean_df['Salary'].plot(kind='kde')
```

<Axes: ylabel='Density'>



observations in Salary numerical column

1. we removed outliers via IQR and clipping , clipping is used as we still got a right tail because of the values greater than 600000
2. Then we got a normal distribution apart from the dent

10percentage column

```
df['10percentage'].mean()
```

77.9254427213607

```
df['10percentage'].median()
```

79.15

mean and median are close they may not be much outliers

```
print(df['10percentage'].quantile(0.25))
```

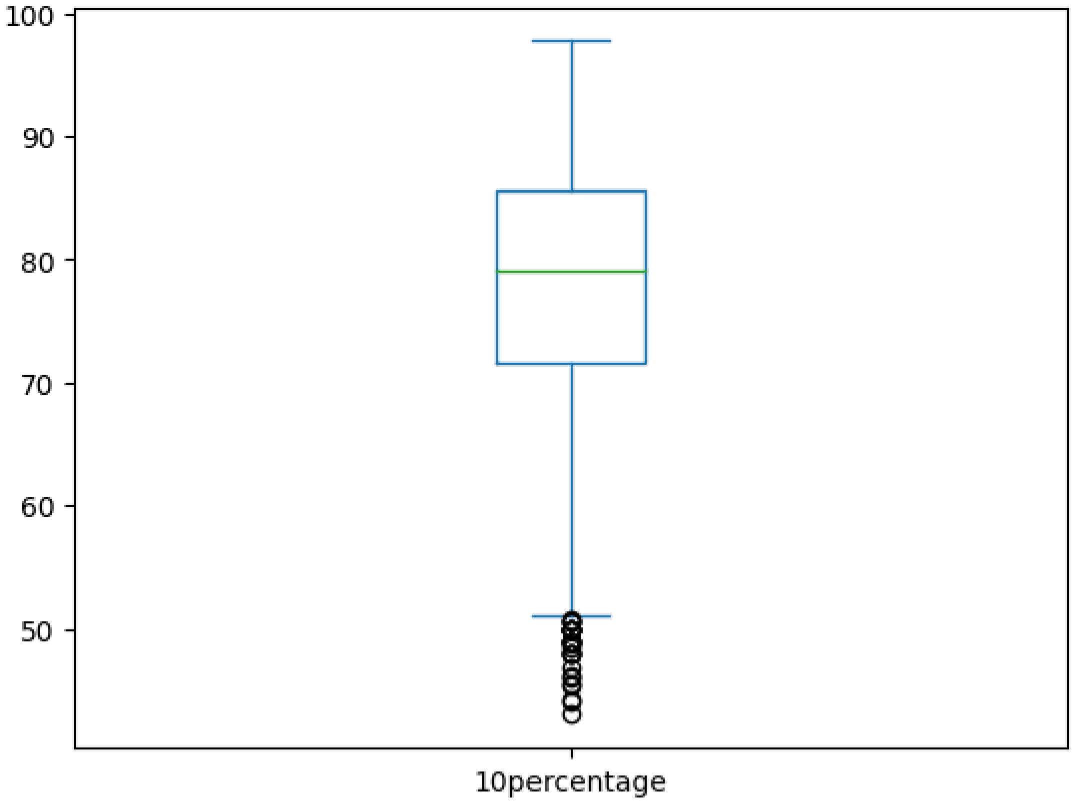
```
71.68
```

```
print(df['10percentage'].quantile(0.75))
```

```
85.67
```

```
num_df['10percentage'].plot(kind='box')
```

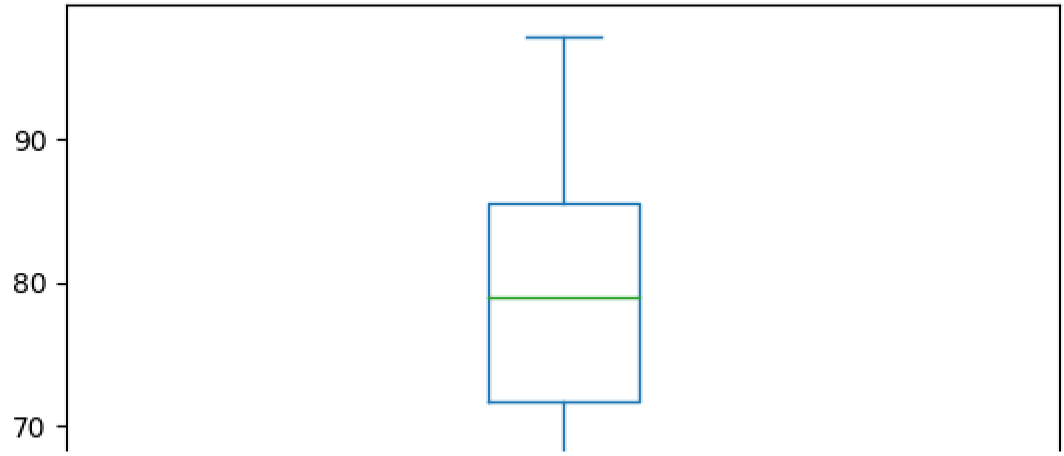
```
<Axes: >
```



we can see the outliers which are below 50 lets remove them by clipping

```
clean_df=clean_df[clean_df['10percentage']>=50]
clean_df['10percentage'].plot(kind='box')
```

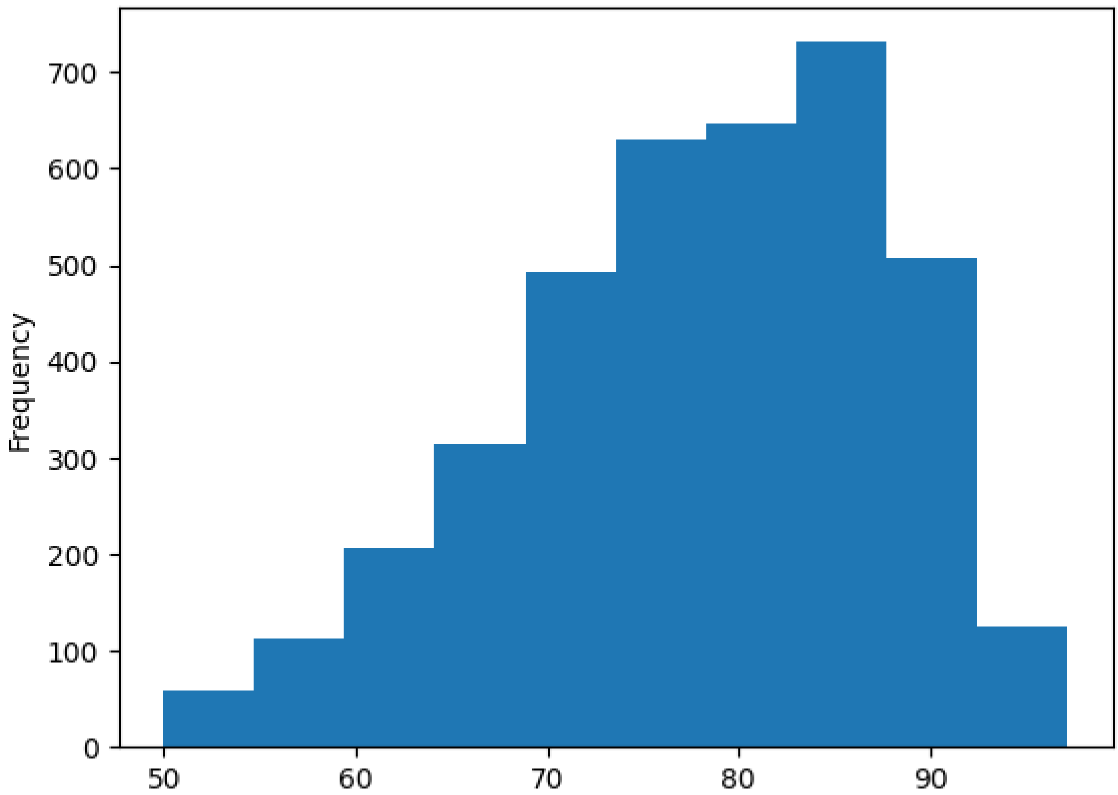
<Axes: >



we can see much less outliers now

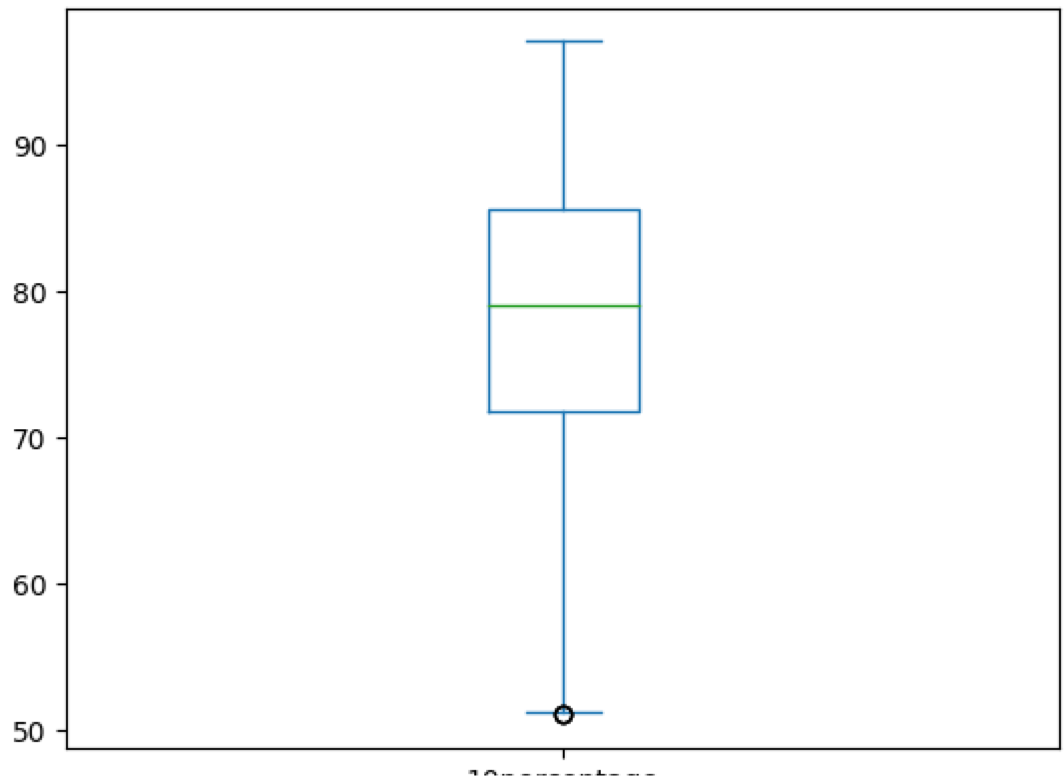
```
#normal distribution of 10percentage via hist
clean_df['10percentage'].plot(kind='hist')
```

<Axes: ylabel='Frequency'>



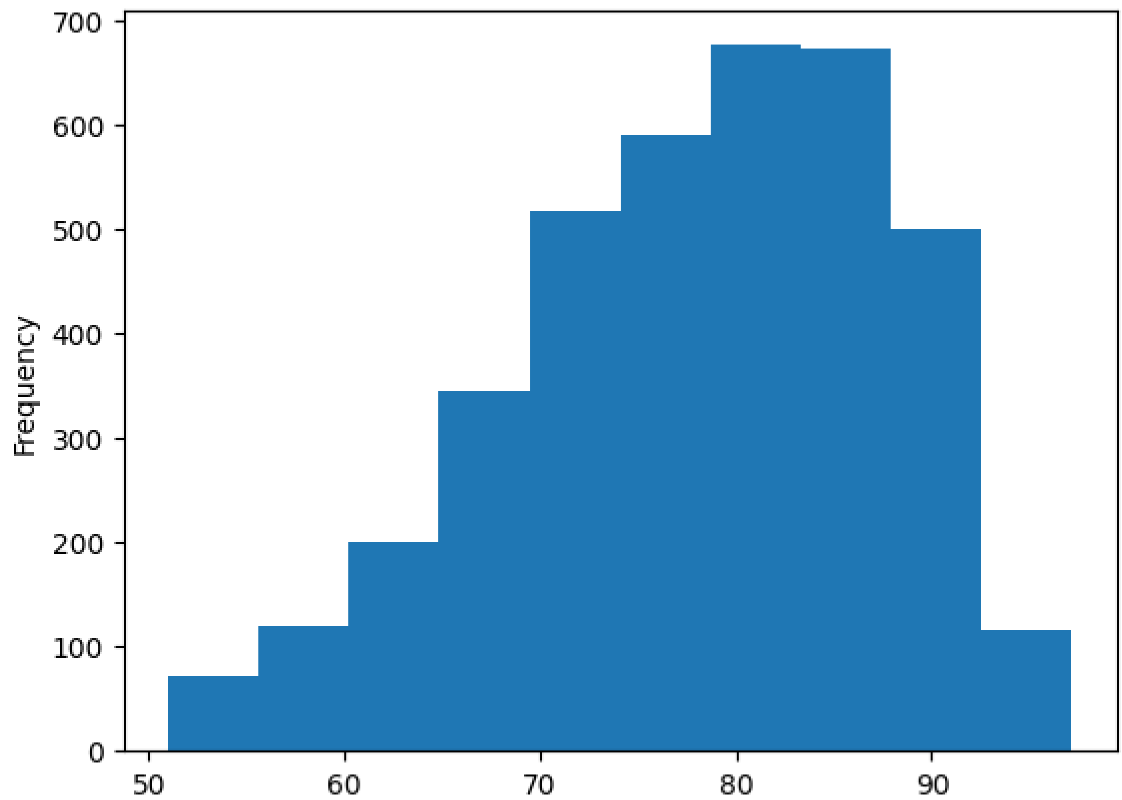

```
#removing outliers using IQR
q1= clean_df['10percentage'].quantile(0.25)
q3= clean_df['10percentage'].quantile(0.75)
IQR=q3-q1
percentage10_lower_bound=q1-1.5*IQR
percentage10_upper_bound=q3+1.5*IQR
clean_df=clean_df[(clean_df['10percentage']>=percentage10_lower_bound)& (clean_df['10perce
clean_df['10percentage'].plot(kind='box')
```

```
<Axes: >
```




```
#normal distribution of 10percentage via hist
clean_df['10percentage'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



we got left tail

observations in 10percentage column

1. outliers are removed (below 50%)
2. the percentage range after removal of outliers is approx(50%) - 100%
3. we got left tail due to less amount of people with that percentage

12graduation column

```
df['12graduation'].mean()
```

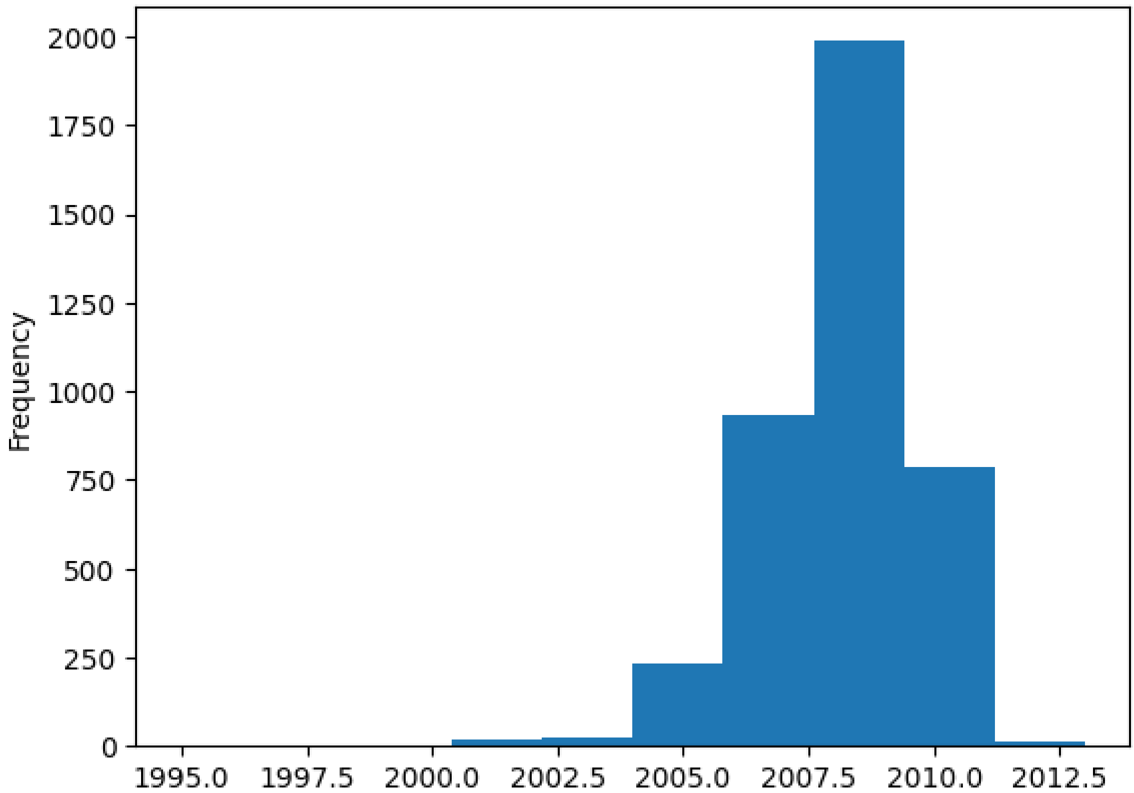
```
2008.087543771886
```

```
df['12graduation'].median()
```

```
2008.0
```

```
num_df['12graduation'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



we can see clearly that 2008 is the year in which most people graduated

12 percentage column

```
df['12percentage'].mean()
```

```
74.46636568284141
```

```
df['12percentage'].median()
```

```
74.4
```

mean and median are close there may not be outliers

```
print(df['12percentage'].quantile(0.25))
```

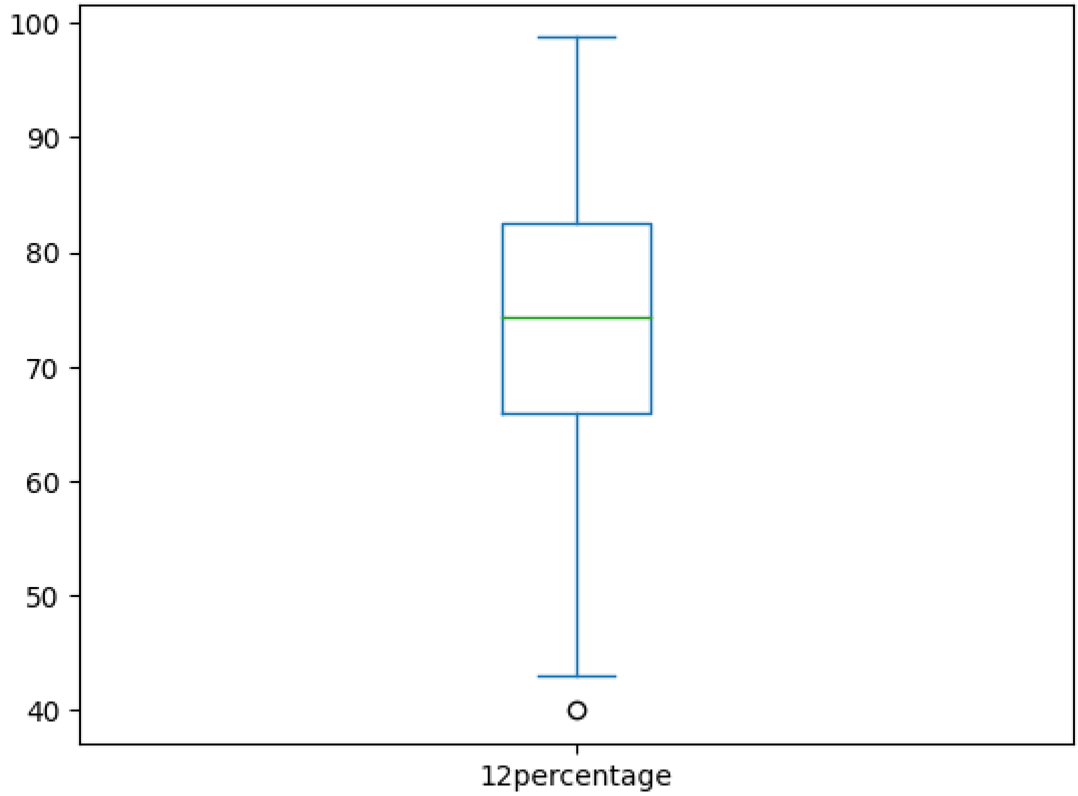
```
66.0
```

```
print(df['12percentage'].quantile(0.75))
```

```
82.6
```

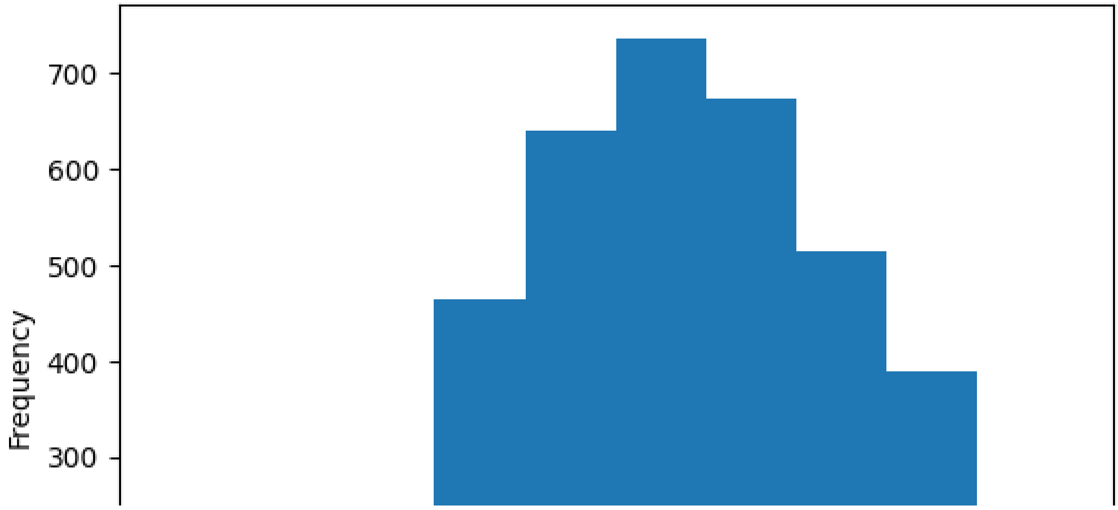
```
num_df['12percentage'].plot(kind='box')
```

```
<Axes: >
```




```
#normal distribution of 12percentage via hist
clean_df['12percentage'].plot(kind='hist')
```

```
&lt;Axes: ylabel='Frequency'&gt;
```



observations in 12percentage column

1. there arent any outliers which make a significant distribution
2. the distribution is normal without any removal of outliers

collegeGPA column

```
df['collegeGPA'].mean()
```

```
71.48617058529265
```

```
df['collegeGPA'].median()
```

```
71.72
```

mean and median arent that close there may be outliers

```
print(df['collegeGPA'].quantile(0.25))
```

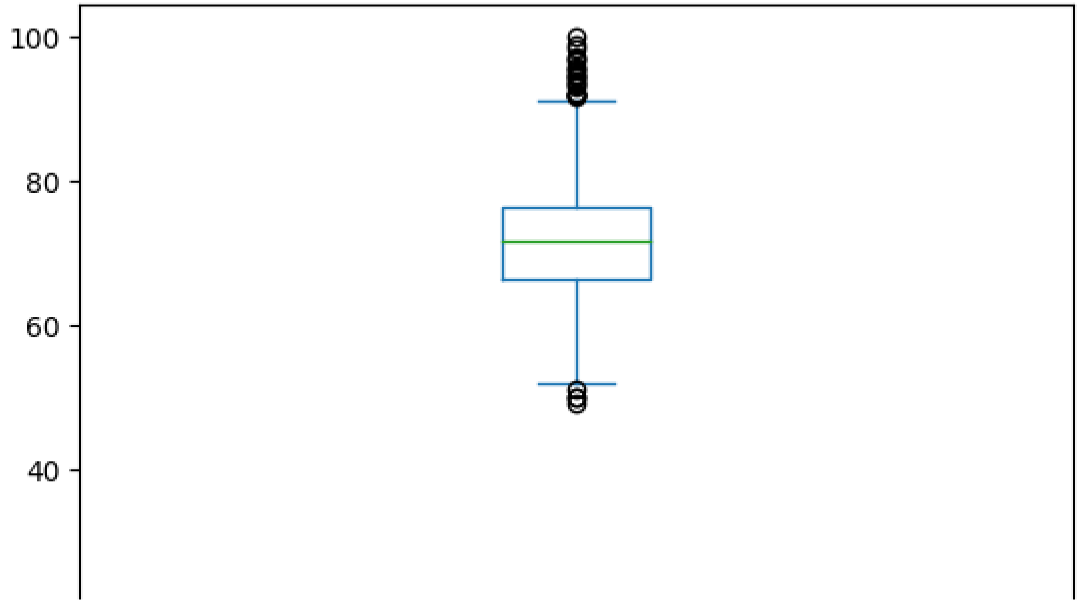
```
66.4075
```

```
print(df['collegeGPA'].quantile(0.75))
```

```
76.3275
```

```
num_df['collegeGPA'].plot(kind='box')
```

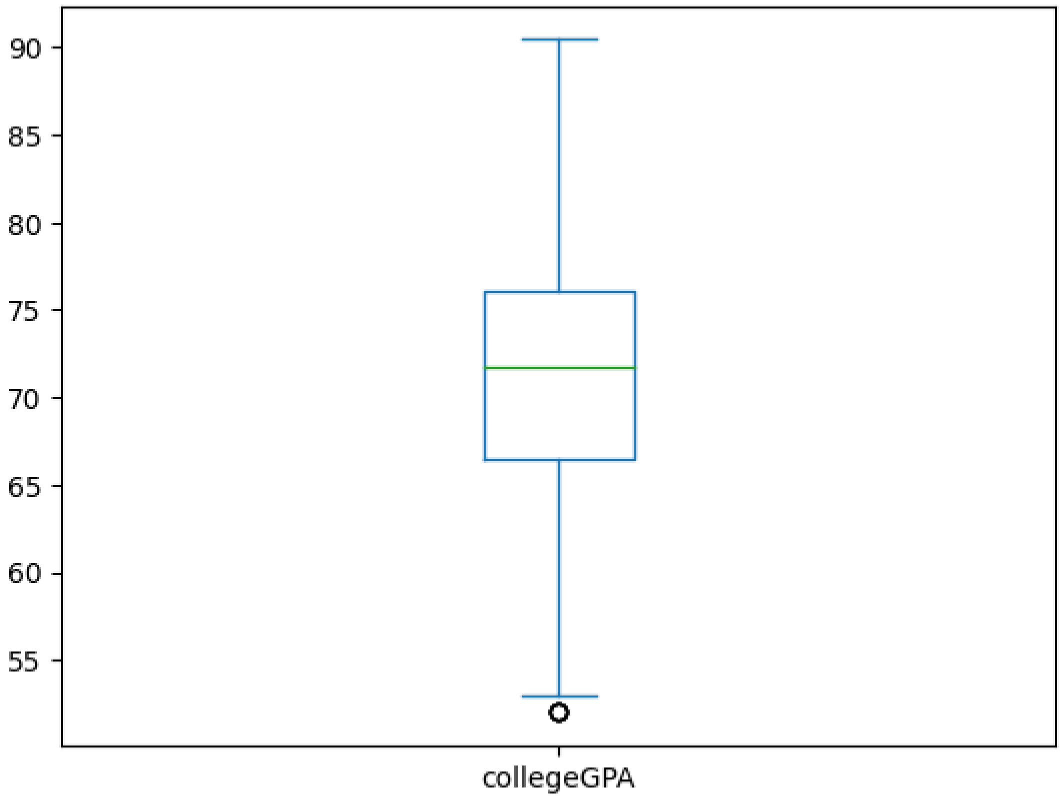
```
<Axes: >
```



lets try to remove these outliers using IQR

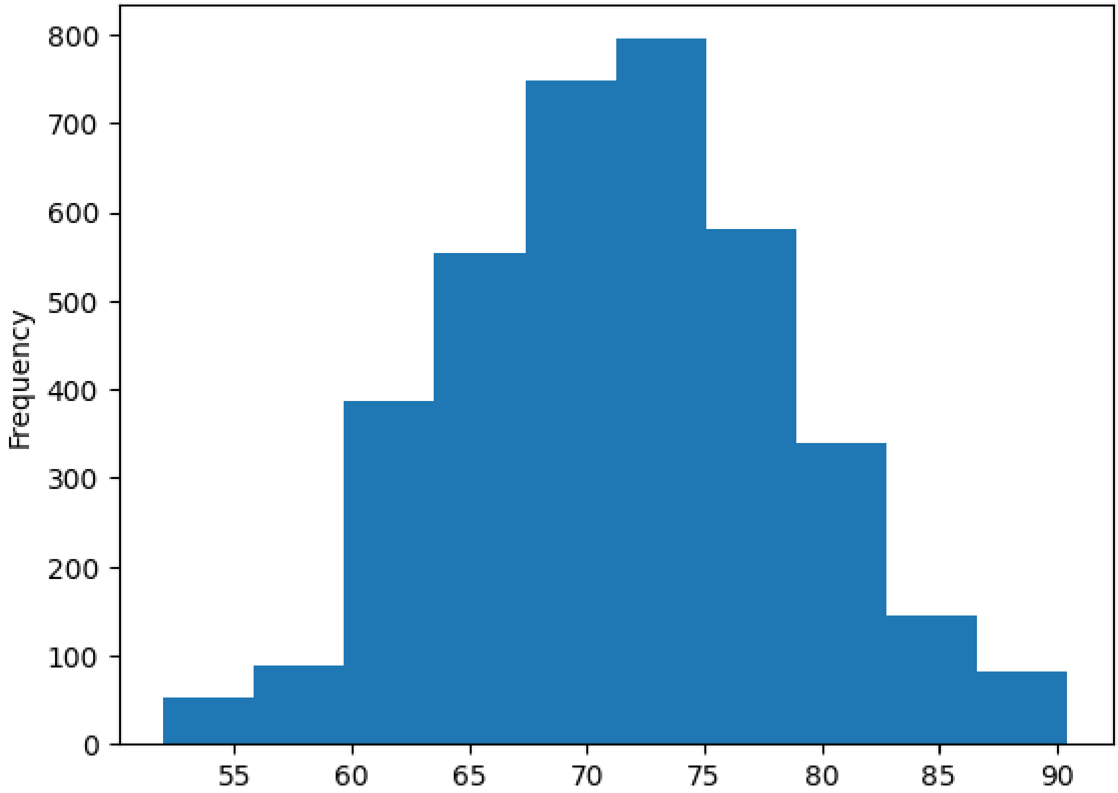
```
#removing outliers using IQR
q1= clean_df['collegeGPA'].quantile(0.25)
q3= clean_df['collegeGPA'].quantile(0.75)
IQR=q3-q1
collegeGPA_lower_bound=q1-1.5*IQR
collegeGPA_upper_bound=q3+1.5*IQR
clean_df=clean_df[(clean_df['collegeGPA']>=collegeGPA_lower_bound)& (clean_df['collegeGPA'
clean_df['collegeGPA'].plot(kind='box')
```

```
<Axes: >
```



```
#normal distribution of collegeGPA via hist
clean_df['collegeGPA'].plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



observations in collegeGPA column

1. outliers are removed using IQR method
2. we can see normal distribution via hist graph
3. college gpa range is reduced to 55-90(approx) from 0-100 as we removed outliers

English, Logical, Quant, Domain columns

```
df['English'].mean()
```

```
501.64907453726863
```

```
df['Logical'].mean()
```

```
501.59879939969983
```

```
df['Quant'].mean()
```

```
513.3781890945472
```

```
df['Domain'].mean()
```

```
0.5104896530075439
```

```
df['English'].median()
```

```
500.0
```

```
df['Logical'].median()
```

```
505.0
```

```
df['Quant'].median()
```

```
515.0
```

```
df['Domain'].median()
```

```
0.622642915849938
```

mean and median of english are close the outliers may be less mean and median of logical,Quant,Domain arent close the outliers may be more

```
print(df['English'].quantile(0.25))
print(df['Logical'].quantile(0.25))
print(df['Quant'].quantile(0.25))
print(df['Domain'].quantile(0.25))
```

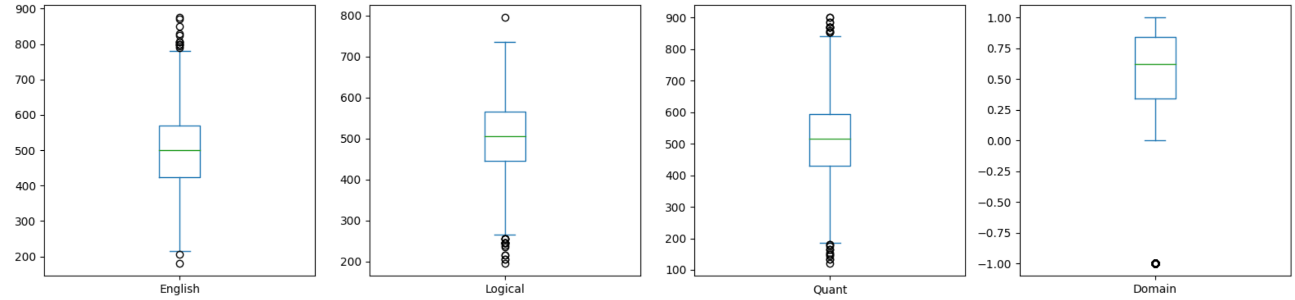
```
425.0
445.0
430.0
0.342314899911815
```

```
print(df['English'].quantile(0.75))
print(df['Logical'].quantile(0.75))
print(df['Quant'].quantile(0.75))
print(df['Domain'].quantile(0.75))
```

```
570.0
565.0
595.0
0.842248322257836
```

```
df4 = pd.DataFrame(df, columns=['English','Logical','Quant','Domain'])
df4.plot(kind='box',subplots=True,layout=(6,6),figsize=(30,30))
```

```
English       Axes(0.125,0.77;0.110714x0.11)
Logical    Axes(0.257857,0.77;0.110714x0.11)
Quant      Axes(0.390714,0.77;0.110714x0.11)
Domain     Axes(0.523571,0.77;0.110714x0.11)
dtype: object
```

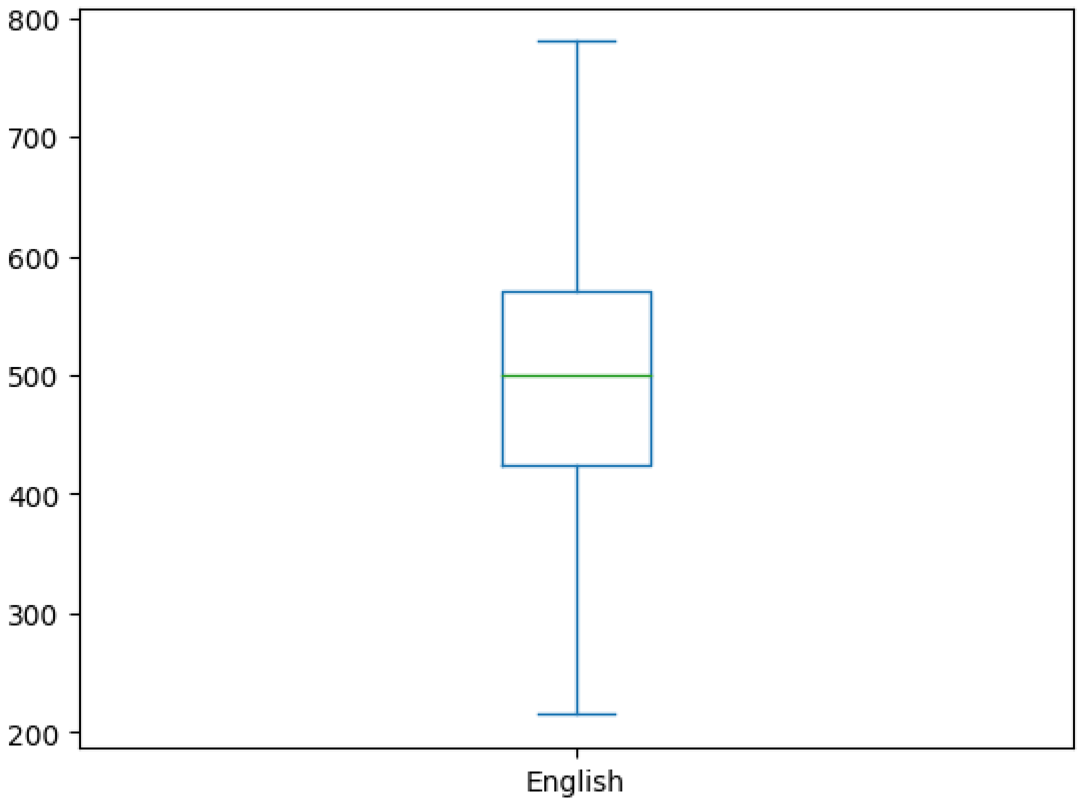



```
df4.plot(kind='hist',subplots=True,layout=(6,6),figsize=(30,30))
```

```
array([[<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
```

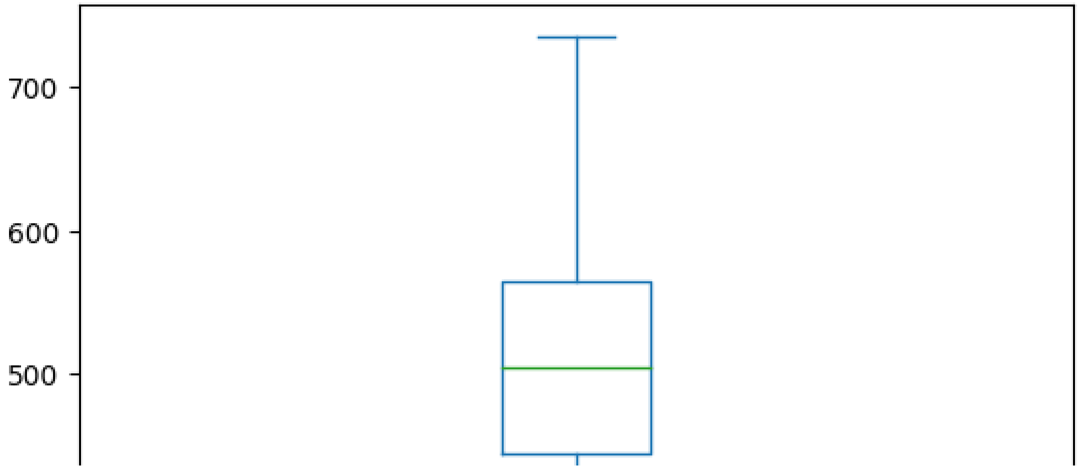
```
#removing outliers using IQR
q1= clean_df['English'].quantile(0.25)
q3= clean_df['English'].quantile(0.75)
IQR=q3-q1
English_lower_bound=q1-1.5*IQR
English_upper_bound=q3+1.5*IQR
clean_df=clean_df[(clean_df['English']>=English_lower_bound)& (clean_df['English']<=Englis
clean_df['English'].plot(kind='box')
```

```
<Axes: >
```



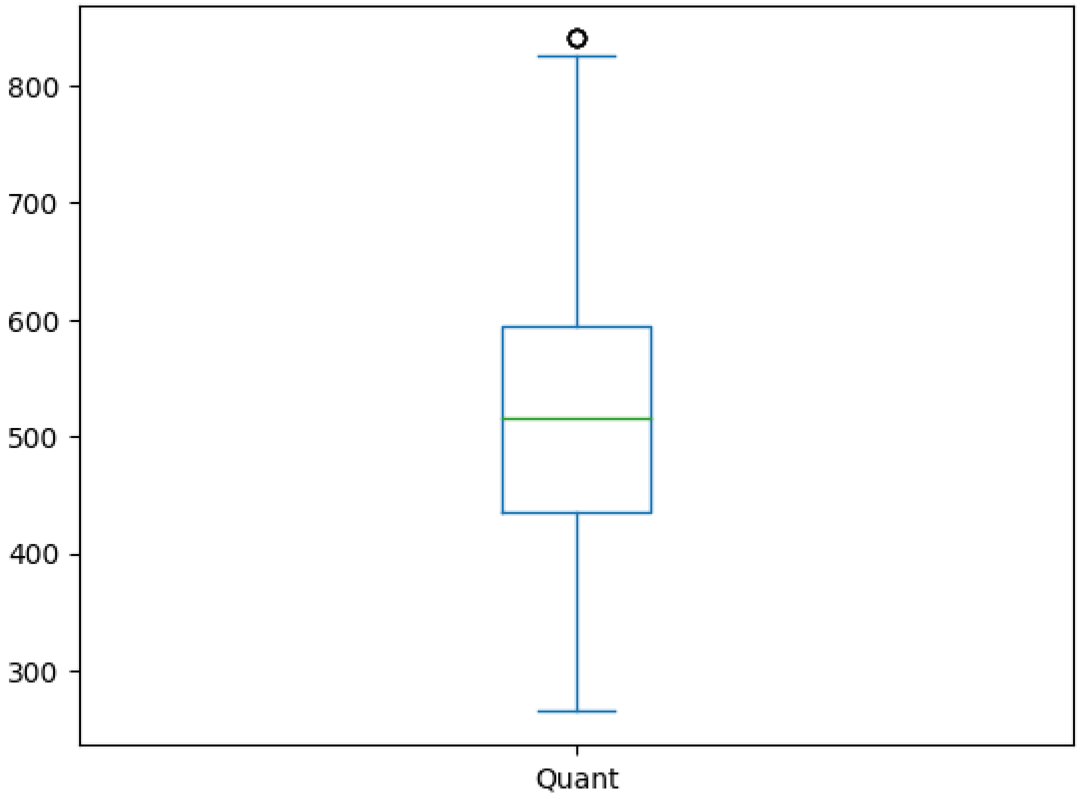

```
#removing outliers using IQR
q1= clean_df['Logical'].quantile(0.25)
q3= clean_df['Logical'].quantile(0.75)
IQR=q3-q1
Logical_lower_bound=q1-1.5*IQR
Logical_upper_bound=q3+1.5*IQR
clean_df=clean_df[(clean_df['Logical']>=Logical_lower_bound)& (clean_df['Logical']<=Logica
clean_df['Logical'].plot(kind='box')
```

<Axes: >



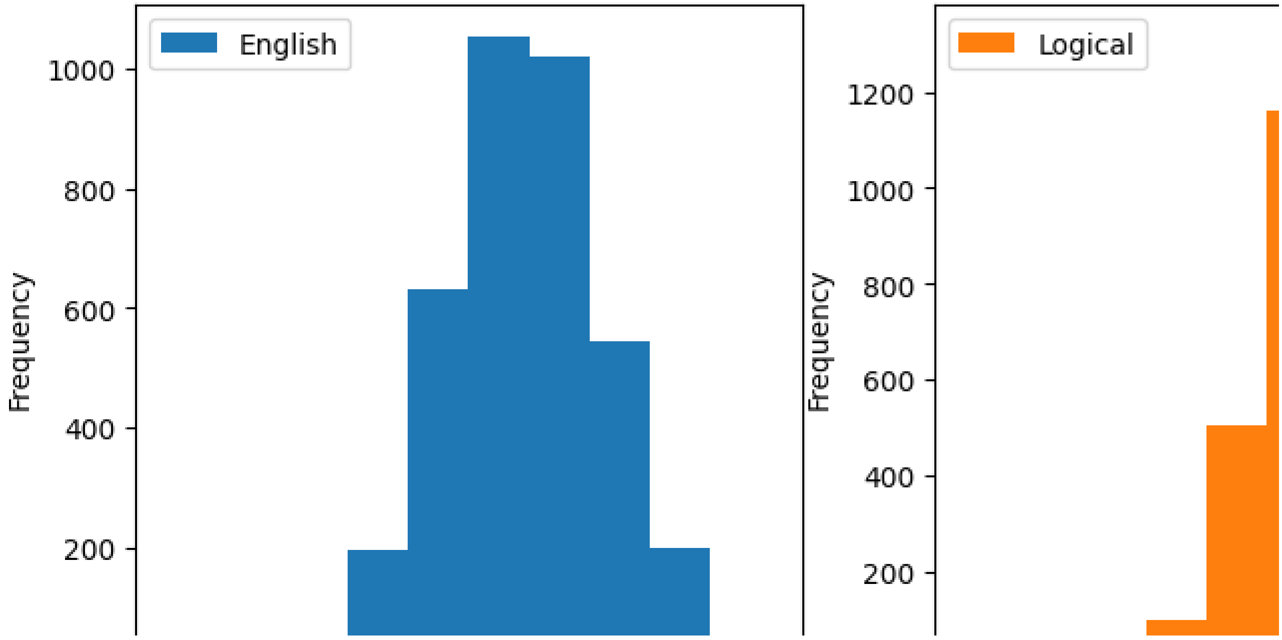

```
#removing outliers using IQR
q1= clean_df['Quant'].quantile(0.25)
q3= clean_df['Quant'].quantile(0.75)
IQR=q3-q1
Quant_lower_bound=q1-1.5*IQR
Quant_upper_bound=q3+1.5*IQR
clean_df=clean_df[(clean_df['Quant']>=Logical_lower_bound)& (clean_df['Quant']<=Quant_uppe
clean_df['Quant'].plot(kind='box')
```

<Axes: >




```
df4_1 = pd.DataFrame(clean_df, columns=['English','Logical','Quant','Domain'])
df4_1.plot(kind='hist',subplots=True,layout=(6,6),figsize=(30,30))
```

```
array([[<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
       [<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
       [<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
       [<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
       [<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>],
       [<Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>,
        <Axes: ylabel='Frequency'>, <Axes: ylabel='Frequency'>]],
      dtype=object)
```

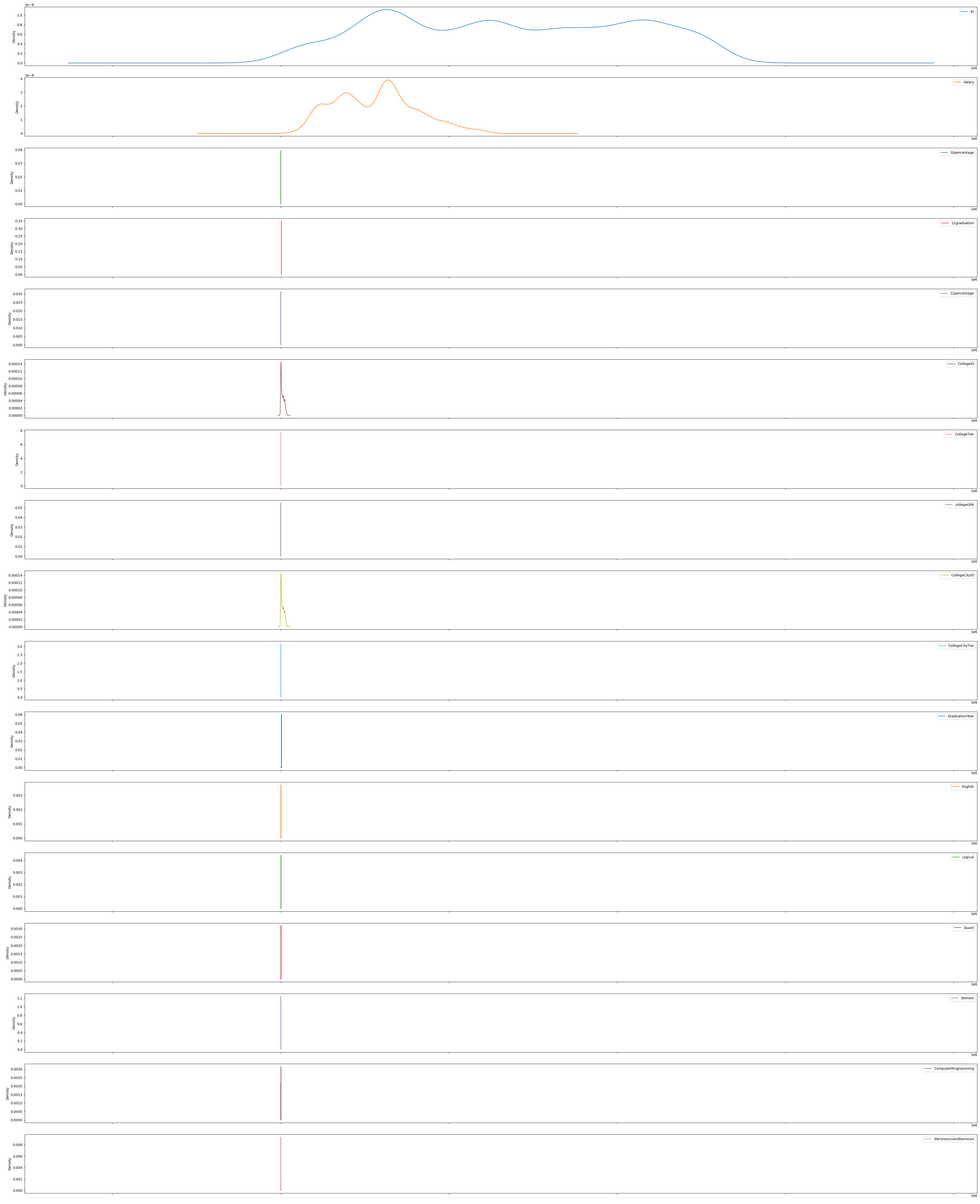


observations in English,Logical,Quant,Domain columns

1. The distribution of data is normal after removal of outliers wherever required
2. No left and right tail observed in columns mentioned above

```
clean_df.plot(kind='kde',subplots=True,figsize=(50,100))
#num_dfpart1=num_df(columns='')
```

```
array([<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
       <Axes: ylabel='Density'>], dtype=object)
```




ComputerScience
MechanicalEngg
ElectricalEngg
TelecomEngg
CivilEngg
conscientiousness
agreeableness
extraversion
nueroticism
openess_to_experience
Density
1e6

Bivariate Analysis

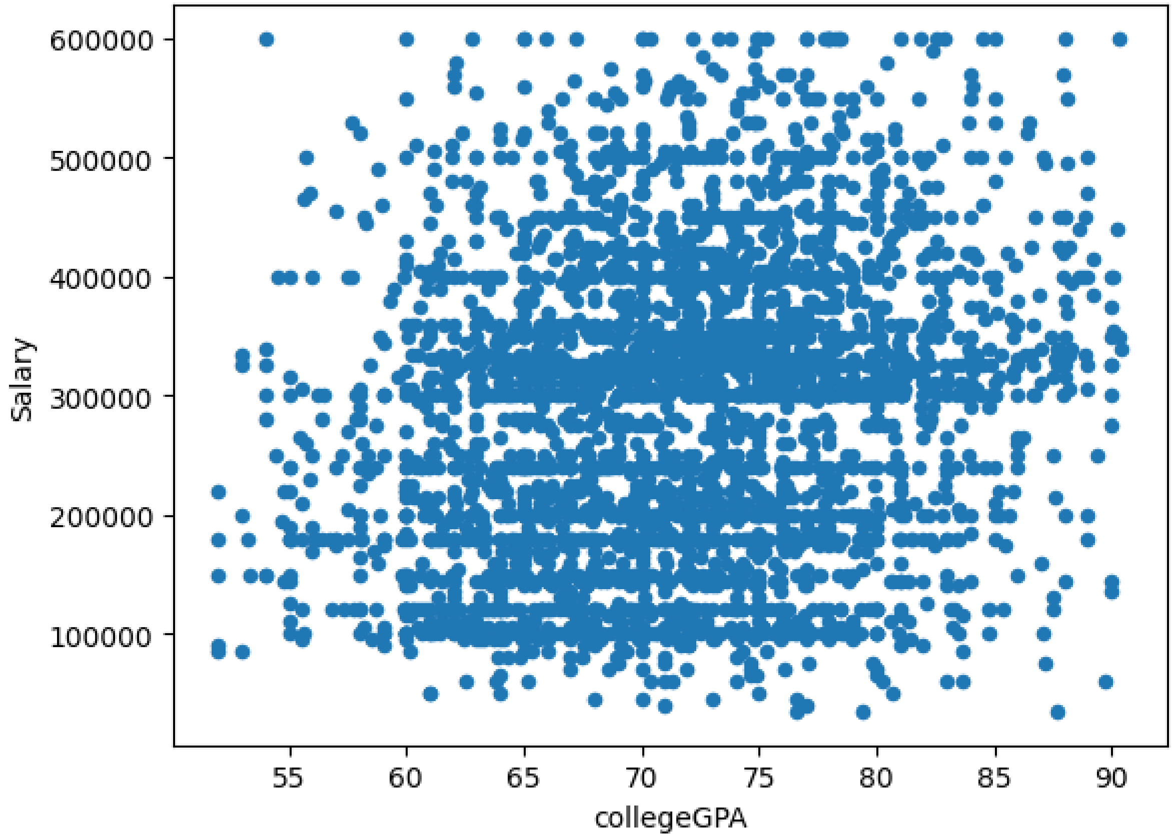
Discover the relationships between numerical columns using Scatter plots, hexbin plots, pair plots, etc..

- Identify the patterns between categorical and numerical columns using swarmplot, boxplot, barplot, etc..

- Mention observations after each plot.

```
clean_df.plot(kind='scatter',x='collegeGPA',y='Salary')
```

```
<Axes: xlabel='collegeGPA', ylabel='Salary'>
```

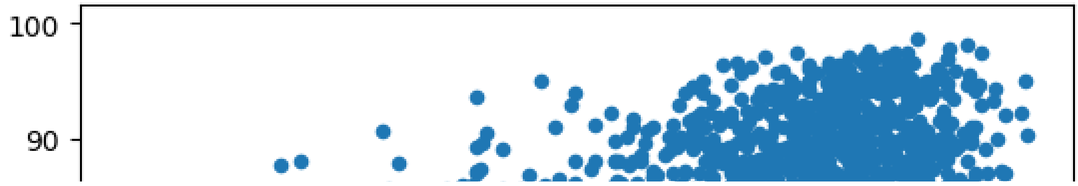


observation

It looks like we dont have a strong relationship between salary and college GPA

```
clean_df.plot(kind='scatter',x='10percentage',y='12percentage')
```

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



observation

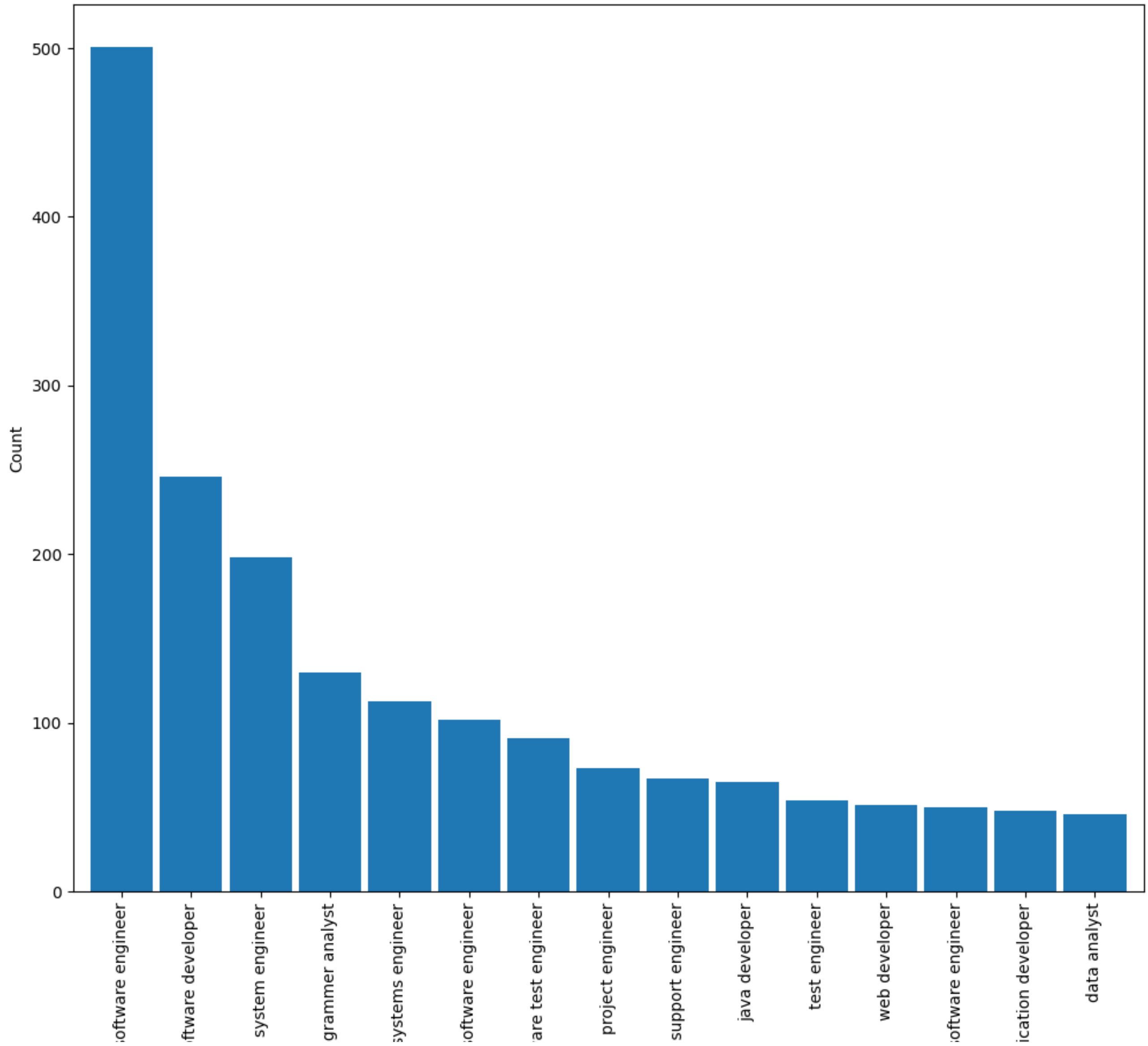
it looks like a mediocre relationship between 10th percentage and 12th percentage

```
clean_df['Designation'].value_counts()
```

```
software engineer           501
software developer          246
system engineer             198
programmer analyst          130
systems engineer            113
                           ...
delivery software engineer    1
graphic designer              1
sales development manager     1
visiting faculty              1
jr. software developer        1
Name: Designation, Length: 406, dtype: int64
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))
clean_df['Designation'].value_counts()[:15].plot(kind='bar' , width=0.9)
plt.xlabel('Designation')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



software engineering is most opted profession as many are employed in that profession


```
clean_df['CollegeTier'].value_counts()
```

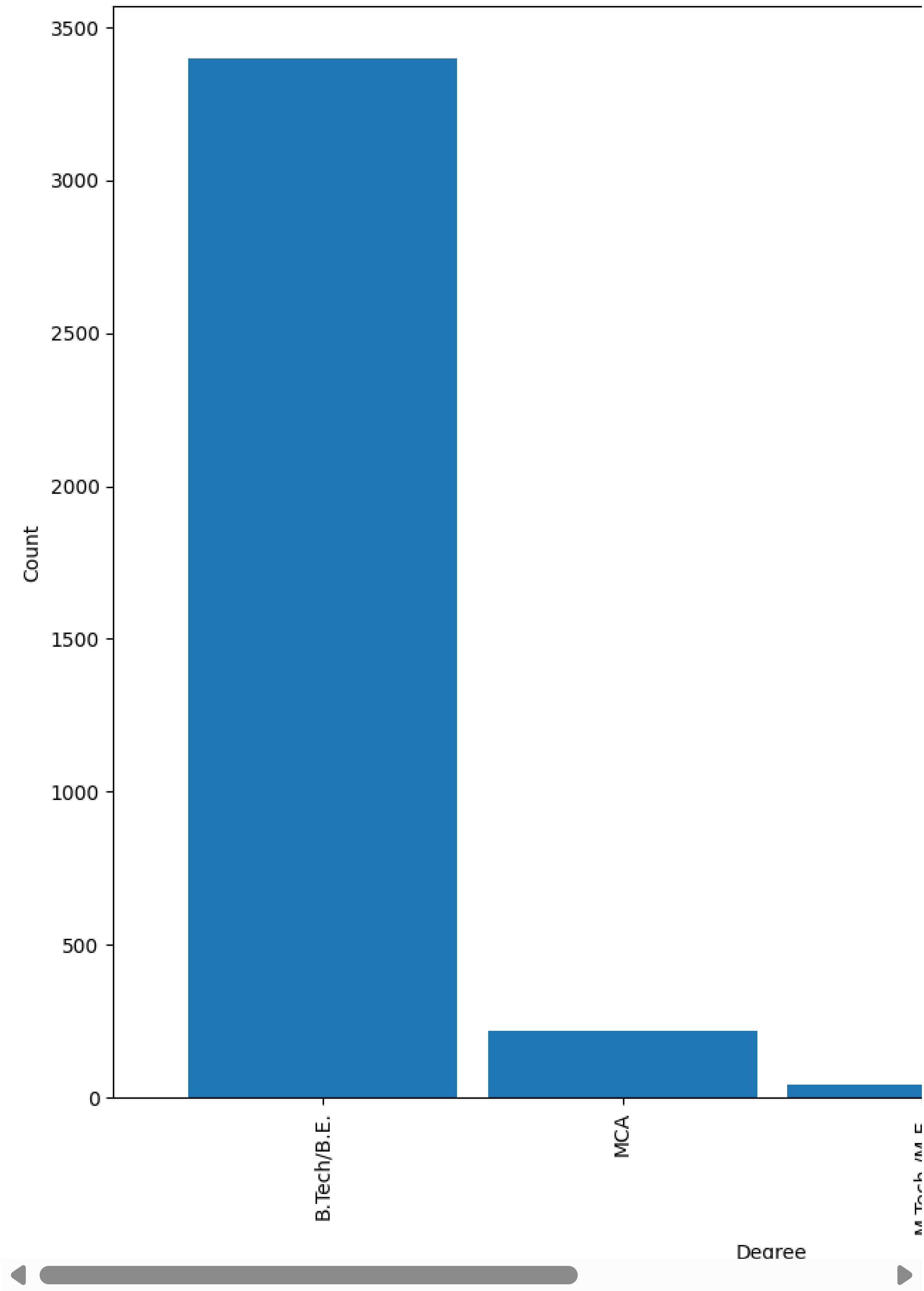
```
2    3421
1     241
Name: CollegeTier, dtype: int64
```

```
clean_df['Degree'].value_counts()
```

```
B.Tech/B.E.      3400
MCA               218
M.Tech./M.E.       42
M.Sc. (Tech.)       2
Name: Degree, dtype: int64
```

```
plt.figure(figsize=(12,10))
clean_df['Degree'].value_counts()[:].plot(kind='bar' , width=0.9)
plt.xlabel('Degree')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



B.Tech/B.E is most opted for graduation as many graduated in that particular domain

```
plt.figure(figsize=(12,10))
clean_df['Specialization'].value_counts()[:].plot(kind='bar' , width=0.9)
```

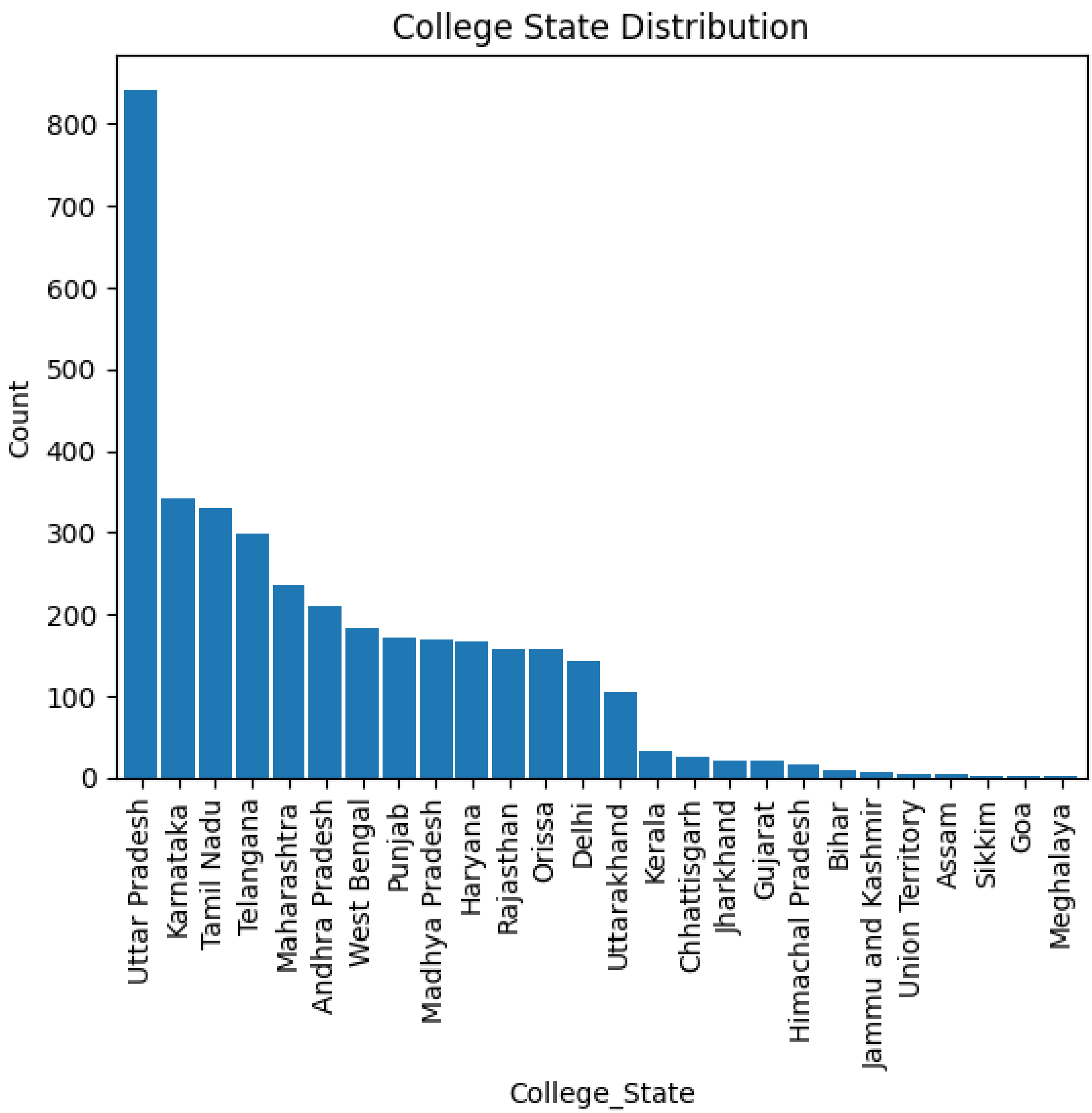
```
plt.xlabel('Specialization')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```

looks like many people are specialised in electronics and communication engineering

```
clean_df['CollegeState'].value_counts()[:].plot(kind='bar' , width=0.9)
plt.xlabel('College_State')
plt.ylabel('Count')
plt.title('College State Distribution')
```

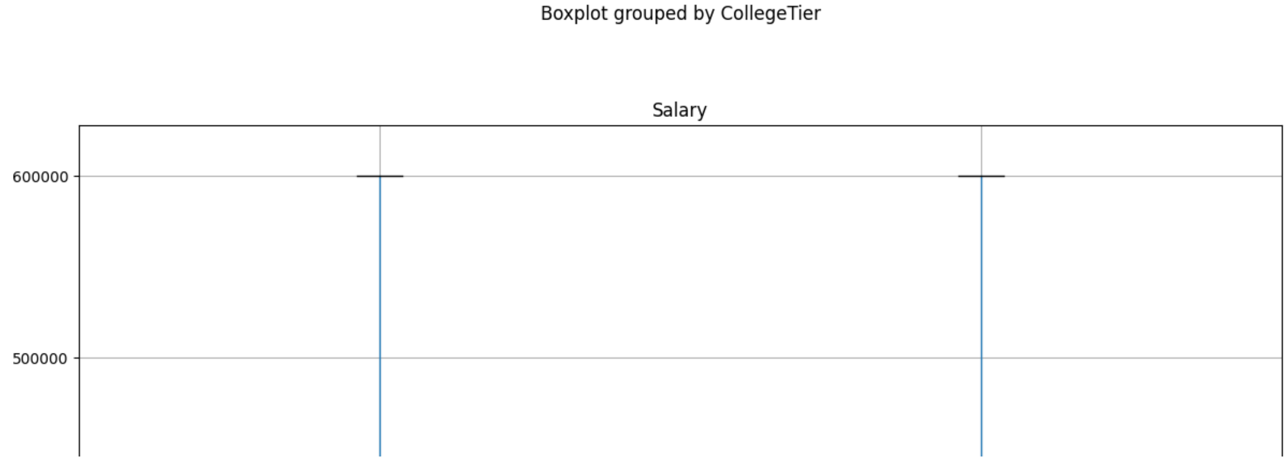
```
Text(0.5, 1.0, 'College State Distribution')
```



uttarpradesh has most number of colleges

```
clean_df.boxplot(by='CollegeTier',column='Salary',figsize=(14,14))
```

```
<Axes: title={'center': 'Salary'}, xlabel='CollegeTier'>
```

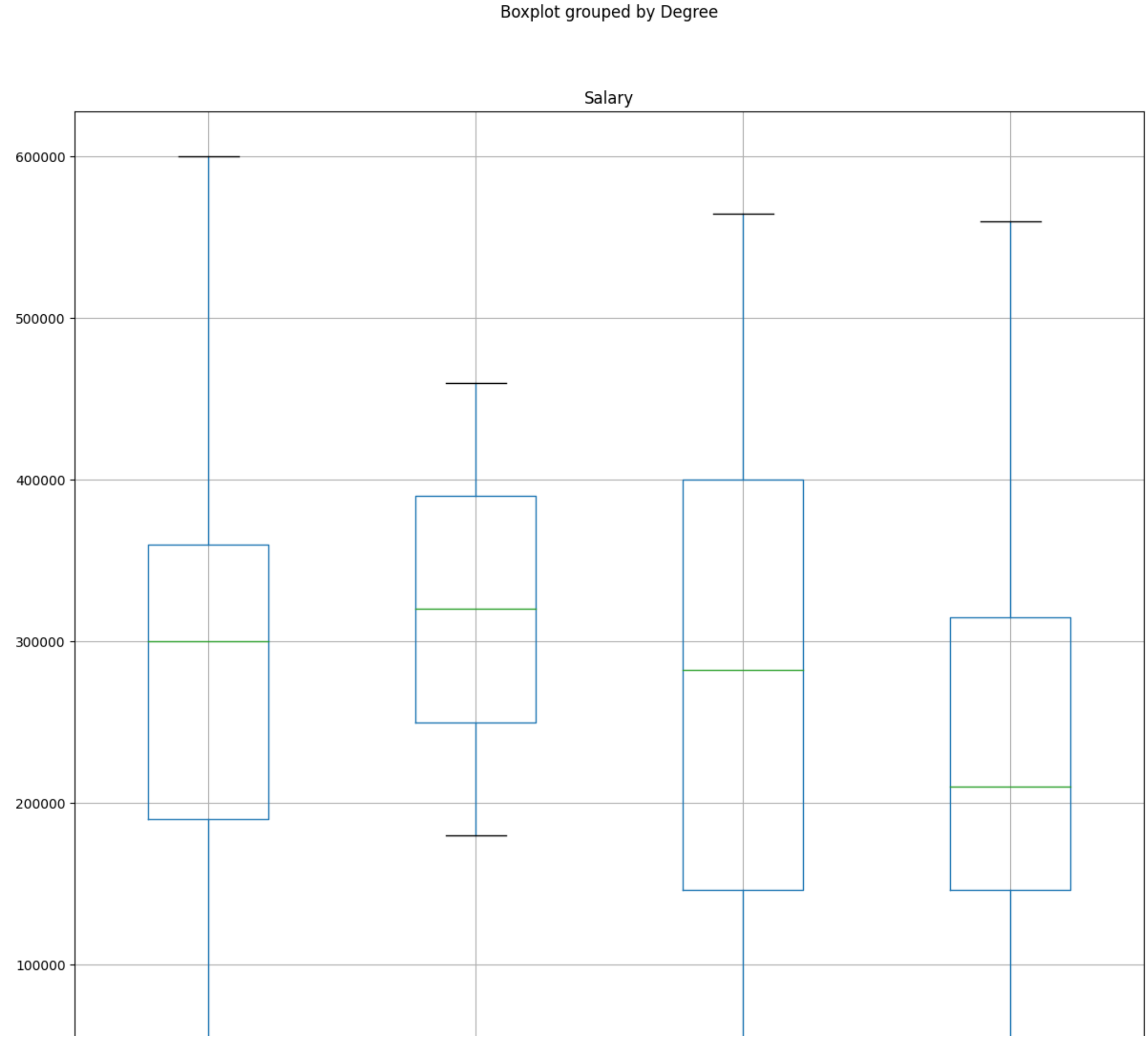


observation

college tier 2's average salary seems to be lesser than college tier 1's salary

```
clean_df.boxplot(by='Degree',column='Salary',figsize=(14,14))
```

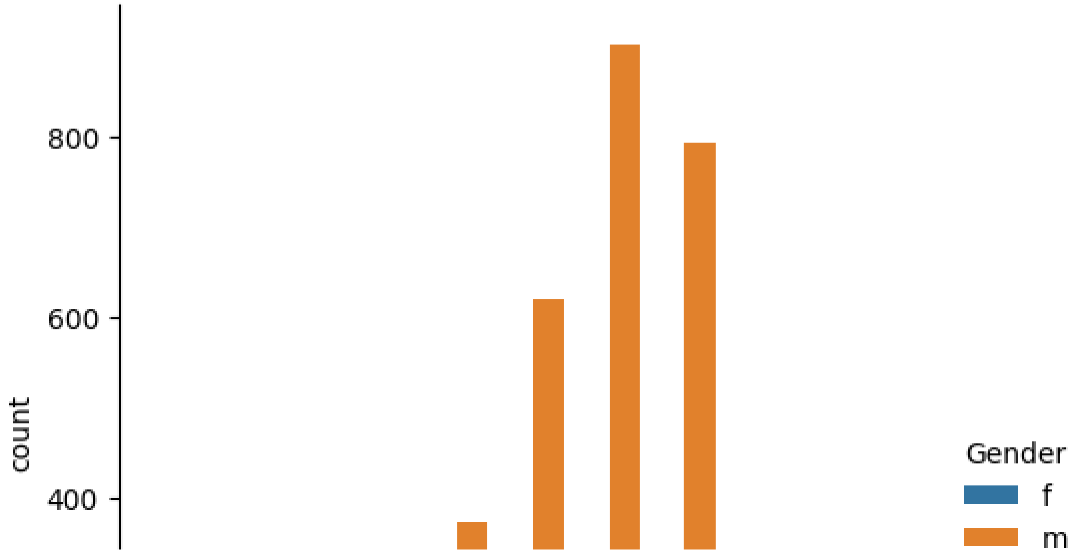
```
<Axes: title={'center': 'Salary'}, xlabel='Degree'>
```



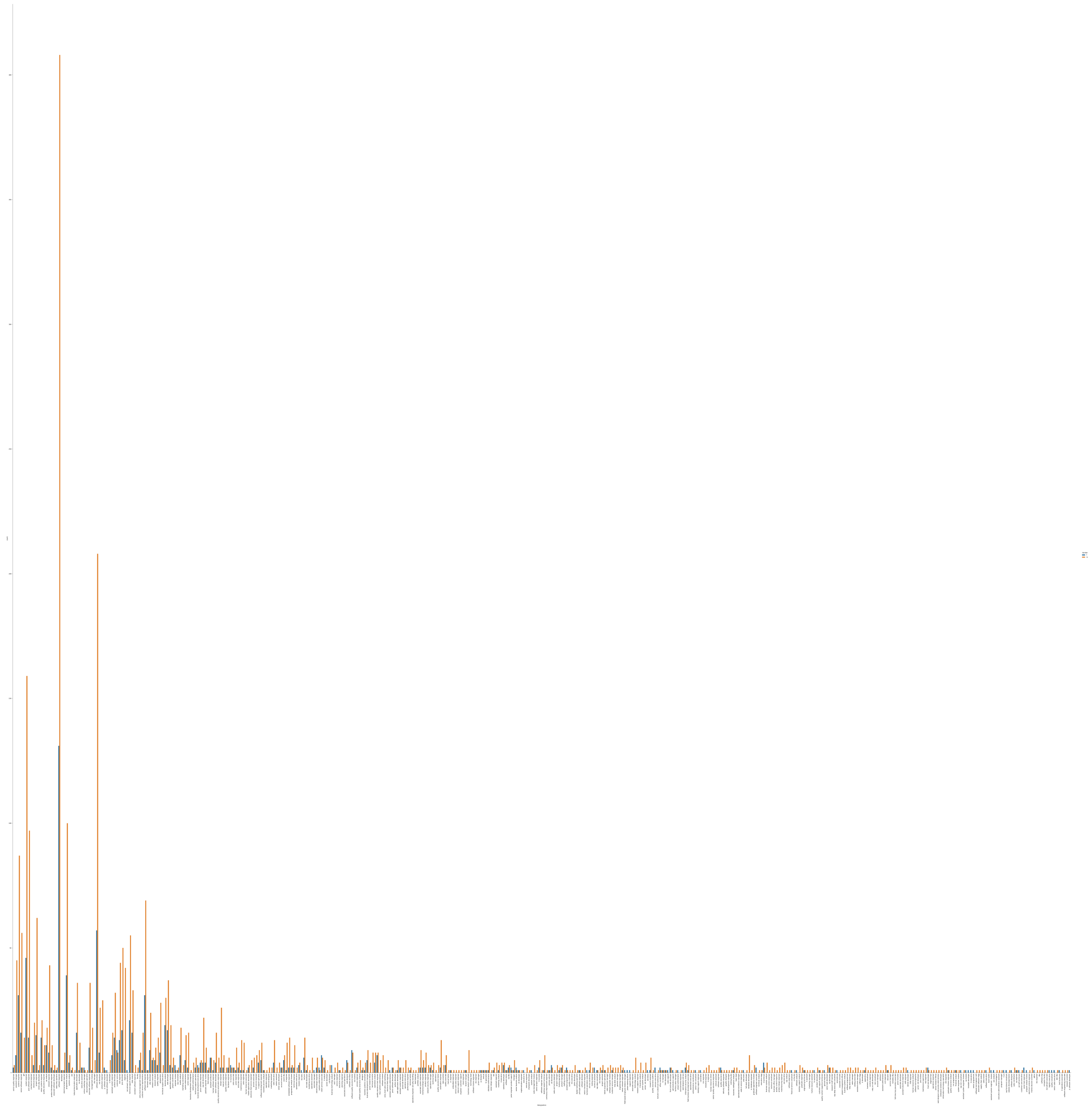
observation

we can see that average salary of M.Sc and B.Tech/B.E is more than M.Tech/M.E and MCA

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




```
sns.catplot(x = "Designation",hue="Gender",data = df,kind='count',height=100)
plt.xticks(rotation=90)
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 given to you.

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

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

```
clean_df['Designation'].value_counts()
```

```
software engineer              501
software developer             246
system engineer                198
programmer analyst             130
systems engineer               113
                              ...
delivery software engineer       1
graphic designer                 1
sales development manager        1
visiting faculty                 1
jr. software developer           1
Name: Designation, Length: 406, dtype: int64
```

```
print('Average Salary :')
print('Programmer Analyst :',round(clean_df['n_sal'][(clean_df['GraduationYear']==2014) &
print('Software Engineer :',round(clean_df['n_sal'][(clean_df['GraduationYear']==2014) & (
print('Hardware Engineer :',round(clean_df['n_sal'][(clean_df['GraduationYear']==2014) &(c
print('Associate Engineer :',round(clean_df['n_sal'][(clean_df['GraduationYear']==2014) &(
```

```
Average Salary :
Programmer Analyst : 3.01
Software Engineer : 3.29
Hardware Engineer : nan
Associate Engineer : 3.32
```

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

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

```
2.565
```

```
# 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
```