AMEO 2015 analysis

Our job here is to find out which factors affect the future salary of an individual. We will find out the insights after processing the data. The dataset has been obtained from Aspiring Minds from the Aspiring Mind Employment Outcome 2015 (AMEO). The study is primarily limited only to students with engineering disciplines.

In [183…

```
#installing packages
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sb
from scipy import stats
from sklearn import preprocessing as preproc
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold, GridSearchCV
import os
from collections import Counter
from numpy import mean
from numpy import std
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import precision_recall_curve

from scipy.stats import ttest_1samp
from pandas import Series;  from numpy.random import randn
from statsmodels.stats.weightstats import ttest_ind
import scipy.stats as stats
from scipy.stats import ttest_ind
import pandas as pd
import numpy as np
import seaborn as sb
%matplotlib inline
from matplotlib import pyplot as plt
from scipy.stats import shapiro
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import LabelEncoder

from scipy.stats import boxcox
from datetime import datetime
from datetime import date
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.linear_model import LassoCV
from sklearn.decomposition import PCA

from scipy.stats import chi2_contingency
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

import lazypredict
from lazypredict.Supervised import LazyRegressor
```

In [2]:

```
#getting the dataset
ameo=pd.read_excel("C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for ML\\Regression\\aspiring_minds_
ameo.head()
```

Out[2]:

	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	...	English	Logical	Quant	Domain	Mark
0	203097	420000	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	...	515	585	525	0.635979	
1	1027655	300000	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	...	560	555	620	-1.000000	
2	947847	300000	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	...	590	435	380	0.356536	
3	87291	600000	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	...	565	455	565	0.765674	
4	606655	270000	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	...	380	580	470	0.229482	

5 rows × 30 columns

In [3]:

```
#examining the dataset
ameo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1961 entries, 0 to 1960
Data columns (total 30 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     1961 non-null   int64
 1   Salary                 1961 non-null   int64
 2   DOJ                    1961 non-null   float64
 3   DOL                    1961 non-null   object
 4   Designation            1961 non-null   object
 5   JobCity                1961 non-null   object
 6   Gender                 1961 non-null   object
 7   DOB                    1961 non-null   int64
 8   Sgradyr                1961 non-null   int64
 9   percentage             1961 non-null   float64
 10  board                  1961 non-null   object
 11  CollegeID              1961 non-null   int64
 12  CollegeTier            1961 non-null   int64
 13  Degree                 1961 non-null   object
 14  Specialization         1961 non-null   object
 15  collegeGPA             1961 non-null   float64
 16  CollegeCityID          1961 non-null   int64
 17  CollegeCityTier        1961 non-null   int64
 18  CollegeState           1961 non-null   object
 19  CGradYear              1961 non-null   int64
 20  English                1961 non-null   int64
 21  Logical                1961 non-null   int64
 22  Quant                  1961 non-null   int64
 23  Domain                 1961 non-null   float64
 24  Marks_obtained         1935 non-null   float64
 25  conscientiousness      1961 non-null   float64
 26  agreeableness          1961 non-null   float64
 27  extraversion           1961 non-null   float64
 28  nueroticism            1961 non-null   float64
 29  openess_to_experience  1961 non-null   float64
dtypes: float64(10), int64(12), object(8)
memory usage: 459.7+ KB
```

In [4]:

```
#examining the shape
ameo.shape
```

Out[4]:

```
(1961, 30)
```

Step 2: Data Pre processing

In this step, we are going to clean our dataset. We are going to look for null values and replace them with mean and mode. We are going to modify some variables if it is necessary and change datatypes for better analysis. We will also remove outliers from the dataset. Outliers hamper the machine learning algorithms and hence they have to be removed.

In [5]:

```
# column name cleaning

ameo.columns = ameo.columns.str.replace(' ', '')
ameo.columns
```

Out[5]:

```
Index(['ID', 'Salary', 'DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB',
       'Sgradyr', 'percentage', 'board', 'CollegeID', 'CollegeTier', 'Degree',
       'Specialization', 'collegeGPA', 'CollegeCityID', 'CollegeCityTier',
       'CollegeState', 'CGradYear', 'English', 'Logical', 'Quant', 'Domain',
       'Marks_obtained', 'conscientiousness', 'agreeableness', 'extraversion',
       'nueroticism', 'openess_to_experience'],
      dtype='object')
```

In [6]:

```
#checking null values
ameo.isna().sum(axis=0)
```

Out[6]:

```
ID                        0
Salary                    0
DOJ                       0
DOL                       0
Designation               0
JobCity                   0
Gender                    0
DOB                       0
Sgradyr                   0
percentage                0
board                     0
CollegeID                 0
CollegeTier               0
Degree                    0
Specialization            0
collegeGPA                0
CollegeCityID             0
CollegeCityTier           0
CollegeState              0
CGradYear                 0
English                   0
Logical                   0
Quant                     0
Domain                    0
Marks_obtained           26
conscientiousness         0
agreeableness             0
extraversion              0
nueroticism               0
openess_to_experience     0
dtype: int64
```

In [149…

```
#changing data type to right format
ameo.DOJ=ameo.DOJ.astype(object)
ameo.DOB=ameo.DOB.astype(object)
ameo.Sgradyr=ameo.Sgradyr.astype(object)
ameo.CGradYear=ameo.CGradYear.astype(object)
ameo.Salary=ameo.Salary.astype(float)



ameo.dtypes
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In [149], line 6
      4 ameo.Sgradyr=ameo.Sgradyr.astype(object)
      5 ameo.CGradYear=ameo.CGradYear.astype(object)
----> 6 ameo.Salary=ameo.Salary.astype(float)
     10 ameo.dtypes

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5583, in NDFrame.__getattr__(self, name)
   5576 if (
   5577     name not in self._internal_names_set
   5578     and name not in self._metadata
   5579     and name not in self._accessors
   5580     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5581 ):
   5582     return self[name]
-> 5583 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'Salary'
```

In [8]:

```
#replacing missing values with numpy nan
ameo.replace("?",np.nan,inplace = True)
ameo.head()
print(ameo.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1961 entries, 0 to 1960
Data columns (total 30 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     1961 non-null   int64
 1   Salary                 1961 non-null   float64
 2   DOJ                    1961 non-null   object
 3   DOL                    1961 non-null   object
 4   Designation            1961 non-null   object
 5   JobCity                1961 non-null   object
 6   Gender                 1961 non-null   object
 7   DOB                    1961 non-null   object
 8   Sgradyr                1961 non-null   object
 9   percentage             1961 non-null   float64
 10  board                  1961 non-null   object
 11  CollegeID              1961 non-null   int64
 12  CollegeTier            1961 non-null   int64
 13  Degree                 1961 non-null   object
 14  Specialization         1961 non-null   object
 15  collegeGPA             1961 non-null   float64
 16  CollegeCityID          1961 non-null   int64
 17  CollegeCityTier        1961 non-null   int64
 18  CollegeState           1961 non-null   object
 19  CGradYear              1961 non-null   object
 20  English                1961 non-null   int64
 21  Logical                1961 non-null   int64
 22  Quant                  1961 non-null   int64
 23  Domain                 1961 non-null   float64
 24  Marks_obtained         1935 non-null   float64
 25  conscientiousness      1961 non-null   float64
 26  agreeableness          1961 non-null   float64
 27  extraversion           1961 non-null   float64
 28  nueroticism            1961 non-null   float64
 29  openess_to_experience  1961 non-null   float64
dtypes: float64(10), int64(8), object(12)
memory usage: 459.7+ KB
None
```

In [9]:

```
#gathering mean values of numerical variables
avg_Marks_obtained= ameo['Marks_obtained'].astype("float").mean(axis=0)
print("The average marks of a student in our Test is",avg_Marks_obtained,"points.")

#replacing numerical missing values with mean
ameo['Marks_obtained'].replace(np.nan, avg_Marks_obtained, inplace=True)
```

```
The average marks of a student in our Test is 678.6428364047317 points.
```

In [10]:

```
#creating a compact variable-aptitude_score
ameo['aptitude_score']=(ameo['English']+ameo['Logical']+ameo['Quant'])/3
ameo['aptitude_score']
ameo.head()
```

Out[10]:

	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	...	Logical	Quant	Domain	Marks_obtai
0	203097	420000.0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	...	585	525	0.635979	
1	1027655	300000.0	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	...	555	620	-1.000000	
2	947847	300000.0	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	...	435	380	0.356536	
3	87291	600000.0	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	...	455	565	0.765674	
4	606655	270000.0	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	...	580	470	0.229482	

5 rows × 31 columns

In [11]:

```
#removing unnecessary variables
ameo=ameo.drop(['English','Logical','Quant','CollegeCityID','CollegeState','ID'],axis=1)
ameo.head()
```

Out[11]:

	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	...	CollegeCityTier	CGradYear	Domain	M
0	420000.0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	cbse	...	0	2011	0.635979	
1	300000.0	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	cbse	...	0	2014	-1.000000	
2	300000.0	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	cbse	...	1	2014	0.356536	
3	600000.0	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	cbse	...	0	2010	0.765674	
4	270000.0	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	cbse	...	0	2013	0.229482	

5 rows × 25 columns

In [12]:

```
#having a look at the new variables
ameo.dtypes
```

Out[12]:

```
Salary                   float64
DOJ                       object
DOL                       object
Designation               object
JobCity                   object
Gender                    object
DOB                       object
Sgradyr                   object
percentage               float64
board                     object
CollegeID                  int64
CollegeTier                int64
Degree                    object
Specialization            object
collegeGPA               float64
CollegeCityTier            int64
CGradYear                 object
Domain                   float64
Marks_obtained           float64
conscientiousness        float64
agreeableness            float64
extraversion             float64
nueroticism              float64
openess_to_experience    float64
aptitude_score           float64
dtype: object
```

In [13]:

```
# Separate categorical and numerical columns
cat_cols = list(ameo.select_dtypes(include=['int64','object']).columns)
num_cols = list(ameo.select_dtypes(include=['float64']).columns)

print('Categorical columns:', cat_cols)
print('Numerical columns:', num_cols)
```

```
Categorical columns: ['DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB', 'Sgradyr', 'board', 'CollegeID',
'CollegeTier', 'Degree', 'Specialization', 'CollegeCityTier', 'CGradYear']
Numerical columns: ['Salary', 'percentage', 'collegeGPA', 'Domain', 'Marks_obtained', 'conscientiousness', 'agr
eeableness', 'extraversion', 'nueroticism', 'openess_to_experience', 'aptitude_score']
```

Univariate analysis

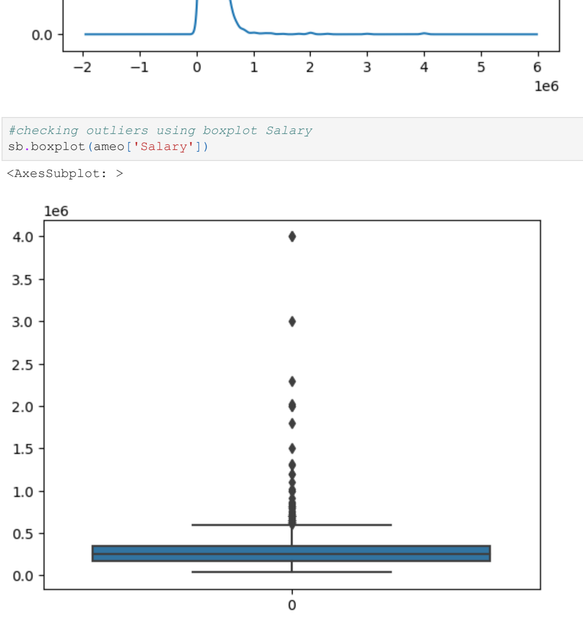
We are using boxplot,histogram and plot to find the outliers,extreme values and distribution of the numerical variables.

In [14]:

```
#checking distribution for Salary
ameo['Salary'].plot.density()
```

Out[14]:

```
<AxesSubplot: ylabel='Density'>
```

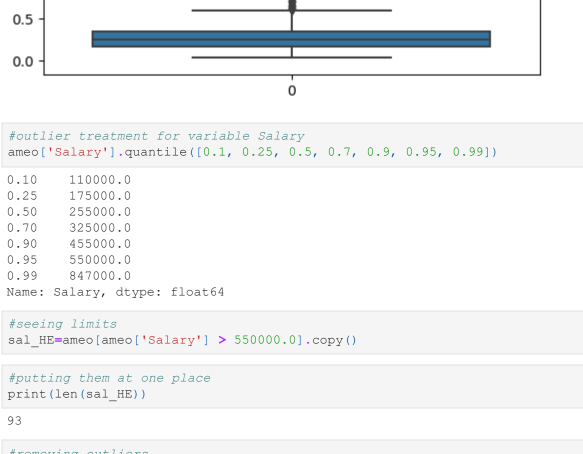


In [15]:

```
#checking outliers using boxplot Salary
sb.boxplot(ameo['Salary'])
```

Out[15]:

```
<AxesSubplot: >
```



In [16]:

```
#outlier treatment for variable Salary
ameo['Salary'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[16]:

```
0.10    110000.0
0.25    175000.0
0.50    255000.0
0.70    325000.0
0.90    455000.0
0.95    550000.0
0.99    847000.0
Name: Salary, dtype: float64
```

In [17]:

```
#seeing limits
sal_HE=ameo[ameo['Salary'] > 550000.0].copy()
```

In [18]:

```
#putting them at one place
print(len(sal_HE))
```

```
93
```

In [19]:

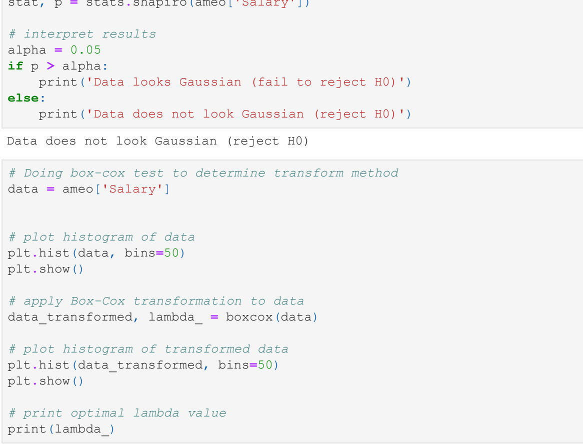
```
#removing outliers
for x in ameo['Salary']:
    if x > 550000.0 :
        ameo['Salary'].replace(x,np.nan,inplace=True)
```

In [20]:

```
#checking if outliers gone
sb.boxplot(ameo['Salary'])
```

Out[20]:

```
<AxesSubplot: >
```



In [21]:

```
#replacing with median values
medsal=ameo['Salary'].median()
ameo['Salary'].replace(np.nan, medsal, inplace=True)
```

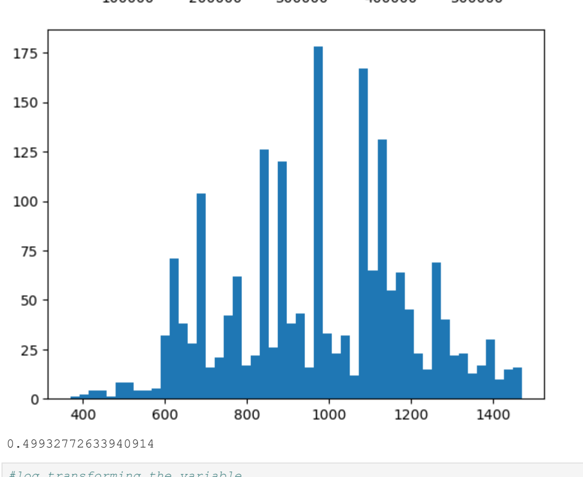
In [22]:

```
# perform Shapiro-Wilk test
stat, p = stats.shapiro(ameo['Salary'])

# interpret results
alpha = 0.05
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')
```

```
Data does not look Gaussian (reject H0)
```

In [23]:

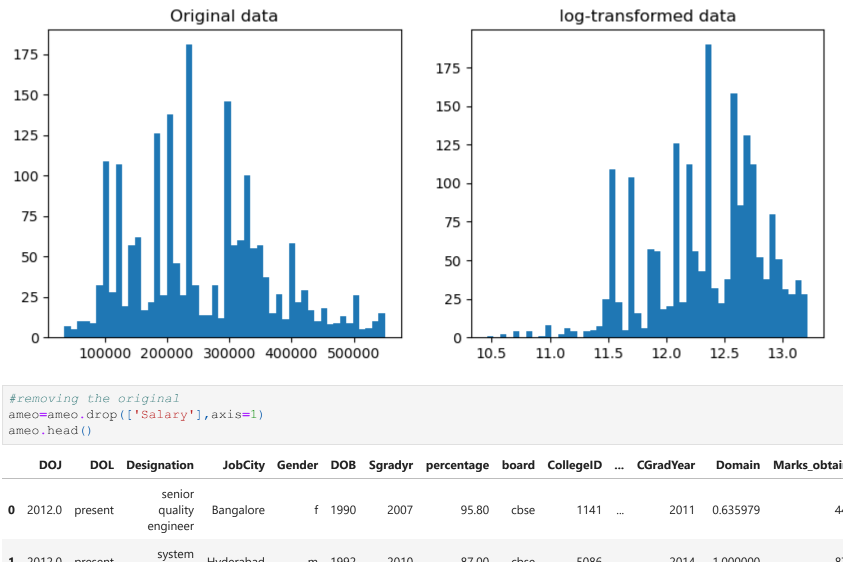
```
# Doing box-cox test to determine transform method
data = ameo['Salary']


# plot histogram of data
plt.hist(data, bins=50)
plt.show()

# apply Box-Cox transformation to data
data_transformed, lambda_ = boxcox(data)

# plot histogram of transformed data
plt.hist(data_transformed, bins=50)
plt.show()

# print optimal lambda value
print(lambda_)
```




```
0.49932772633940914
```

In [24]:

```
#log transforming the variable


# apply log transformation
ameo['Salarytf']= np.log(ameo['Salary'])



fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].hist(data, bins=50)
axs[0].set_title('Original data')

axs[1].hist(ameo['Salarytf'], bins=50)
axs[1].set_title('log-transformed data')



plt.show()
```



In [25]:

```
#removing the original
ameo=ameo.drop(['Salary'],axis=1)
ameo.head()
```

Out[25]:

	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	...	CGradYear	Domain	Marks_obtained
0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	cbse	1141	...	2011	0.635979	445.0
1	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	cbse	5086	...	2014	-1.000000	872.0
2	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	cbse	314	...	2014	0.356536	445.0
3	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	cbse	403	...	2010	0.765674	822.0
4	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	cbse	2665	...	2013	0.229482	420.0

5 rows × 25 columns

Remarks

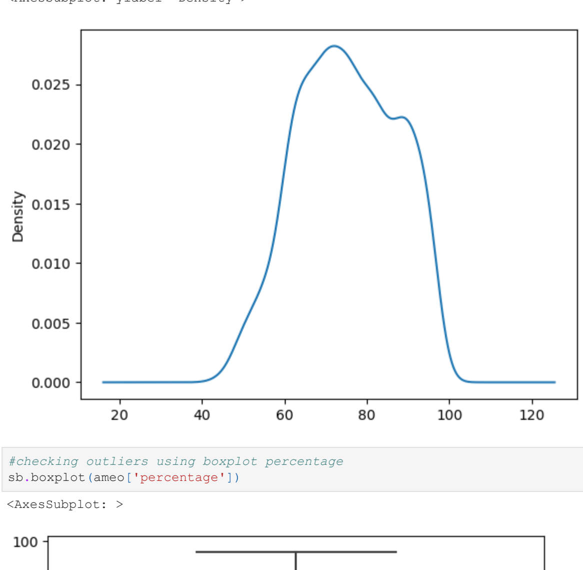
The variable salary was right skewed and had about 93 outliers which was removed and have been replaced with median. The boxcox value was 0.49 which required a log transformation to transform the right skewness of the variable.

In [26]:

```
#checking distribution for percentage
ameo['percentage'].plot.density()
```

Out[26]:

```
<AxesSubplot: ylabel='Density'>
```

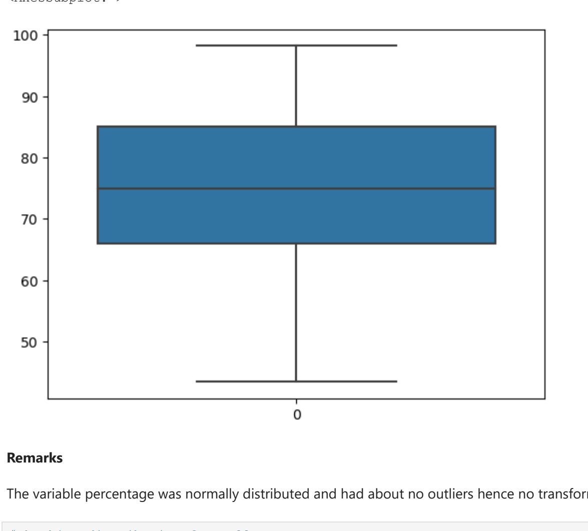


In [27]:

```
#checking outliers using boxplot percentage
sb.boxplot(ameo['percentage'])
```

Out[27]:

```
<AxesSubplot: >
```



Remarks

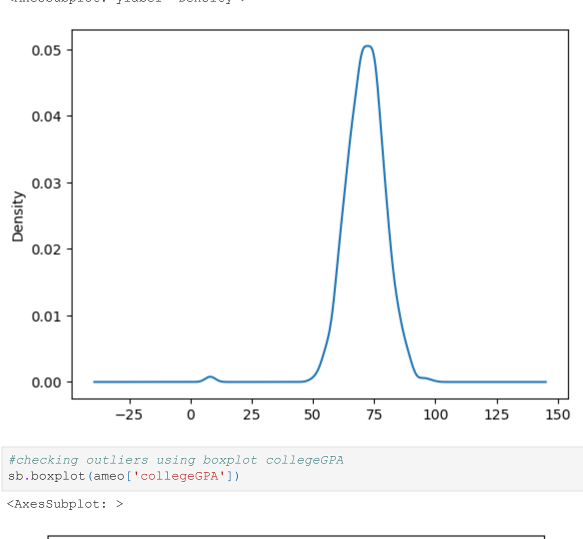
The variable percentage was normally distributed and had about no outliers hence no transformation was needed.

In [28]:

```
#checking distribution for collegeGPA
ameo['collegeGPA'].plot.density()
```

Out[28]:

```
<AxesSubplot: ylabel='Density'>
```

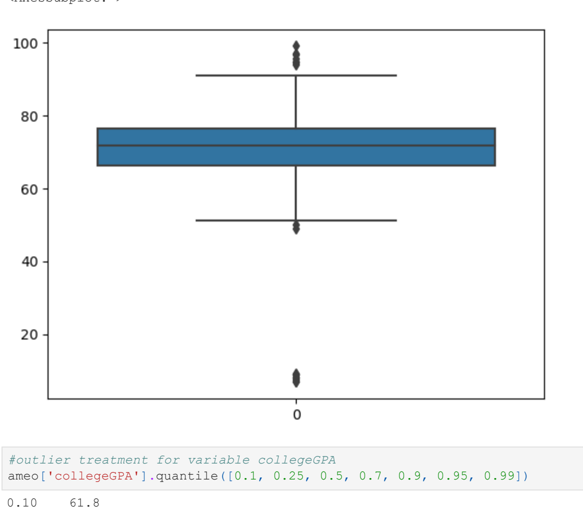


In [29]:

```
#checking outliers using boxplot collegeGPA
sb.boxplot(ameo['collegeGPA'])
```

Out[29]:

```
<AxesSubplot: >
```



In [30]:

```
#outlier treatment for variable collegeGPA
ameo['collegeGPA'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[30]:

```
0.10    61.8
0.25    66.2
0.50    71.7
0.70    75.5
0.90    81.0
0.95    84.0
0.99    89.0
Name: collegeGPA, dtype: float64
```

In [31]:

```
#seeing limits
gpa_HE=ameo[ameo['collegeGPA'] > 80].copy()
gpa_LE=ameo[ameo['collegeGPA'] < 50].copy()

#putting them at one place
print(len(gpa_HE))
print(len(gpa_LE))
```

```
234
9
```

In [32]:

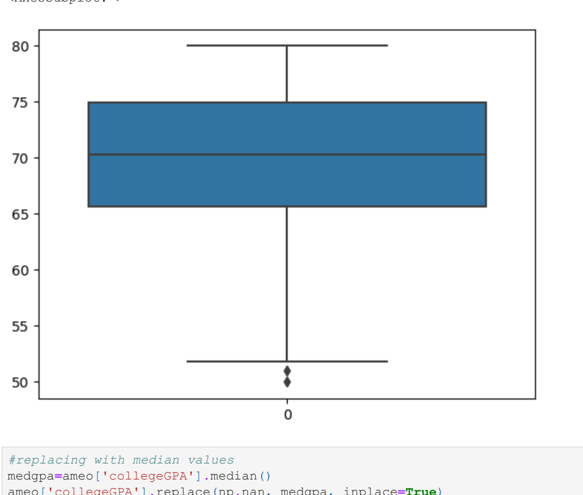
```
#removing outliers
for x in ameo['collegeGPA']:
    if x > 80.0 or x< 50.0:
        ameo['collegeGPA'].replace(x,np.nan,inplace=True)
```

In [33]:

```
#checking if outliers gone
sb.boxplot(ameo['collegeGPA'])
```

Out[33]:

```
<AxesSubplot: >
```



In [34]:

```
#replacing with median values
medgpa=ameo['collegeGPA'].median()
ameo['collegeGPA'].replace(np.nan, medgpa, inplace=True)
```

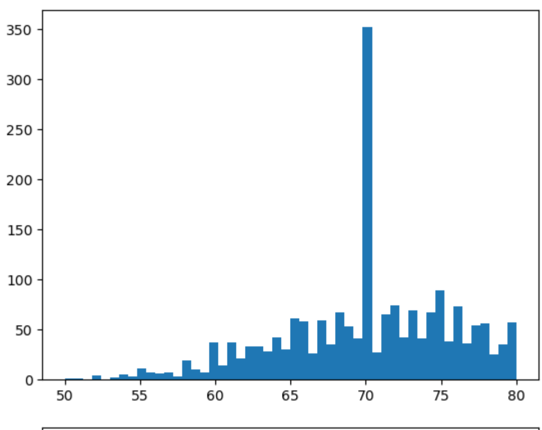
In [35]:

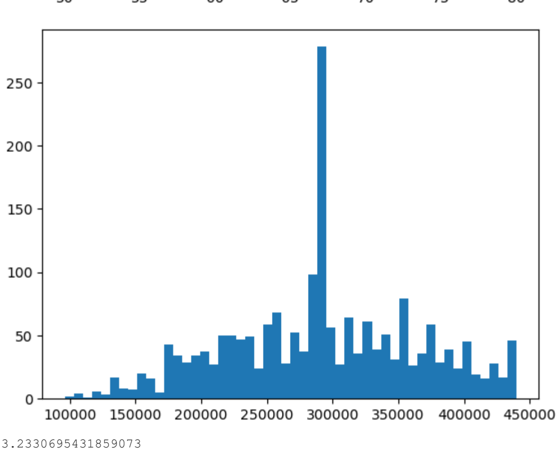
```
# perform Shapiro-Wilk test
stat, p = stats.shapiro(ameo['collegeGPA'])

# interpret results
alpha = 0.05
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')
```

```
Data does not look Gaussian (reject H0)
```

In [36]:

```
# Doing box-cox test to determine transform method
data = ameo['collegeGPA']


# plot histogram of data
plt.hist(data, bins=50)
plt.show()

# apply Box-Cox transformation to data
data_transformed, lambda_ = boxcox(data)

# plot histogram of transformed data
plt.hist(data_transformed, bins=50)
plt.show()

# print optimal lambda value
print(lambda_)
```

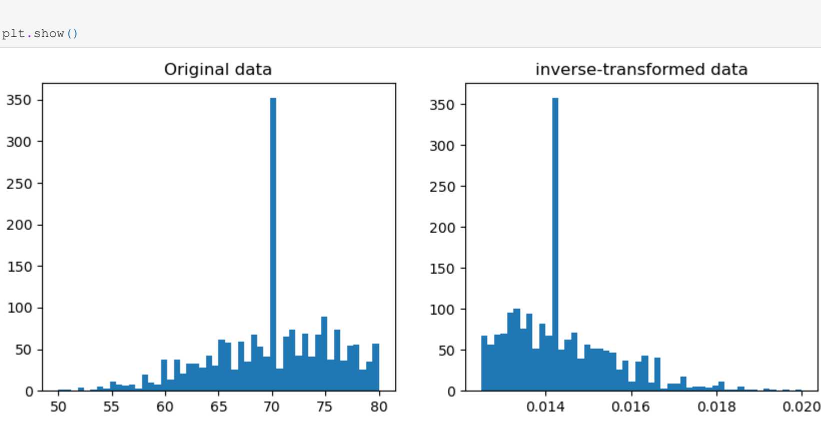




3.2330695431859073

In [37]:

```
#inverse transforming the variable



# apply log transformation
ameo['collegeGPAtf']= 1/ameo['collegeGPA']



fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].hist(data, bins=50)
axs[0].set_title('Original data')

axs[1].hist(ameo['collegeGPAtf'], bins=50)
axs[1].set_title('inverse-transformed data')



plt.show()
```



In [38]:

```
#removing the original
ameo=ameo.drop(['collegeGPA'],axis=1)
ameo.head()
```

Out[38]:

	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	...	Domain	Marks_obtained	conscientiou
0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	cbse	1141	...	0.635979	445.0	0
1	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	cbse	5086	...	-1.000000	872.0	-0
2	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	cbse	314	...	0.356536	445.0	1
3	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	cbse	403	...	0.765674	822.0	0
4	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	cbse	2665	...	0.229482	420.0	0

5 rows × 25 columns

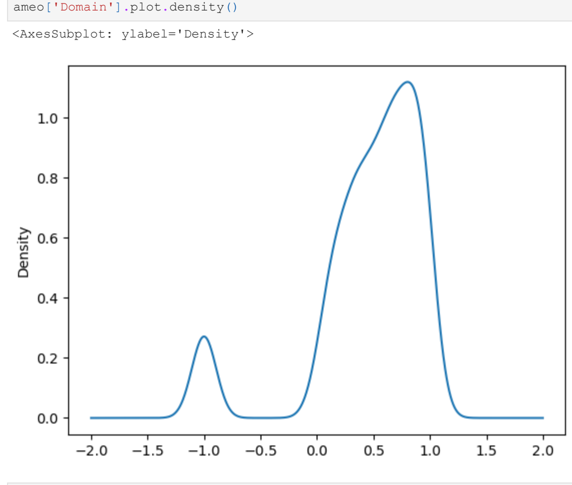
Remarks

The variable GPA was left skewed and had about 243 outliers which was removed and have been replaced with median. The boxcox value was 3 which required an inverse transformation to transform the left skewness of the variable.

In [39]:

```
#checking distribution for Domain
ameo['Domain'].plot.density()
```

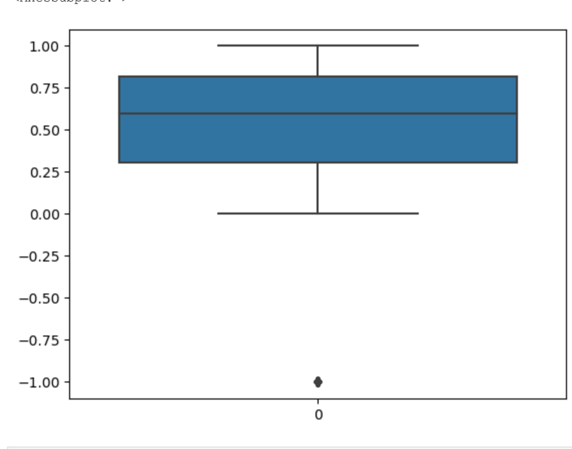
Out[39]: `<AxesSubplot: ylabel='Density'>`



In [40]:

```
#checking outliers using boxplot Domain
sb.boxplot(ameo['Domain'])
```

Out[40]: `<AxesSubplot: >`



In [41]:

```
#outlier treatment for variable Domain
ameo['Domain'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[41]:

```
0.10    0.062221
0.25    0.308401
0.50    0.598281
0.70    0.765674
0.90    0.942117
0.95    0.974396
0.99    0.995614
Name: Domain, dtype: float64
```

In [42]:

```
#seeing limits
dom_LE=ameo[ameo['Domain'] < 0.3].copy()
#putting them at one place
print(len(dom_LE))
```

```
482
```

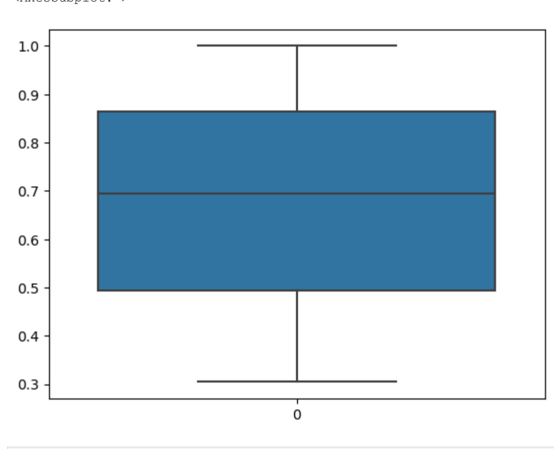
In [43]:

```
#removing outliers
for x in ameo['Domain']:
    if x < 0.3 :
        ameo['Domain'].replace(x,np.nan,inplace=True)
```

In [44]:

```
#checking if outliers gone
sb.boxplot(ameo['Domain'])
```

Out[44]: `<AxesSubplot: >`



In [45]:

```
#replacing with median values
meddom=ameo['Domain'].median()
ameo['Domain'].replace(np.nan, meddom, inplace=True)
```

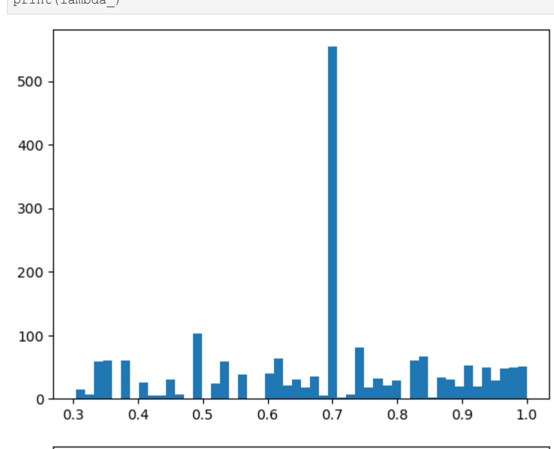
In [46]:

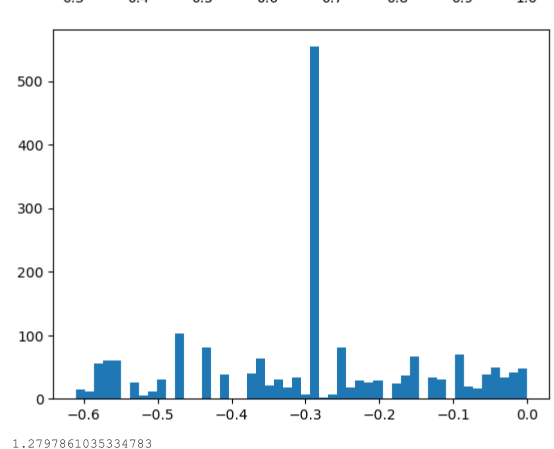
```
# perform Shapiro-Wilk test
stat, p = stats.shapiro(ameo['Domain'])

# interpret results
alpha = 0.05
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')
```

```
Data does not look Gaussian (reject H0)
```

In [47]:

```
# Doing box-cox test to determine transform method
data = ameo['Domain']


# plot histogram of data
plt.hist(data, bins=50)
plt.show()

# apply Box-Cox transformation to data
data_transformed, lambda_ = boxcox(data)

# plot histogram of transformed data
plt.hist(data_transformed, bins=50)
plt.show()

# print optimal lambda value
print(lambda_)
```





1.2797861035334783

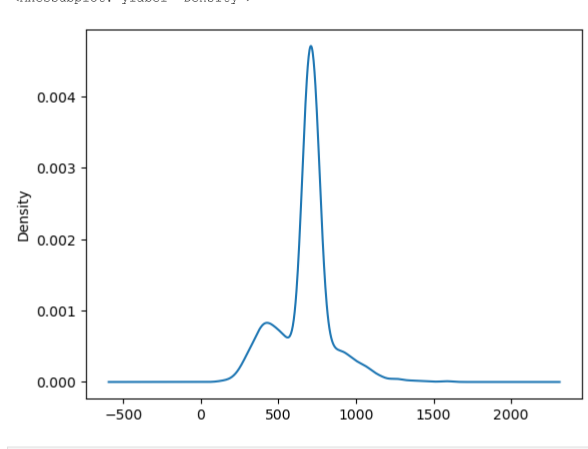
Remarks

The variable Domain was left skewed and had about 482 outliers which was removed and have been replaced with median. The boxcox value was 1.2 which required no transformation.

In [48]:

```
#checking distribution for Marks_obtained
ameo['Marks_obtained'].plot.density()
```

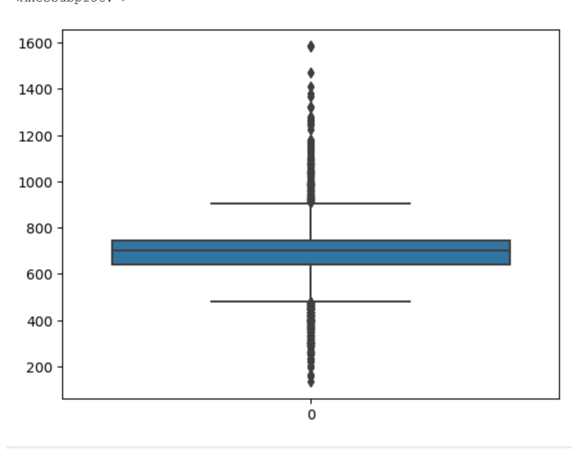
Out[48]: `<AxesSubplot: ylabel='Density'>`



In [49]:

```
#checking outliers using boxplot Marks_obtained
sb.boxplot(ameo['Marks_obtained'])
```

Out[49]: `<AxesSubplot: >`



In [50]:

```
#outlier treatment for variable Marks_obtained
ameo['Marks_obtained'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[50]:

```
0.10     415.000000
0.25     639.000000
0.50     699.941772
0.70     736.940991
0.90     853.000000
0.95     976.000000
0.99    1164.600000
Name: Marks_obtained, dtype: float64
```

In [51]:

```
#seeing limits
marks_HE=ameo[ameo['Marks_obtained'] > 820.0].copy()
marks_LE=ameo[ameo['Marks_obtained'] <640.0].copy()

#putting them at one place
print(len(marks_HE))
print(len(marks_LE))
```

```
230
492
```

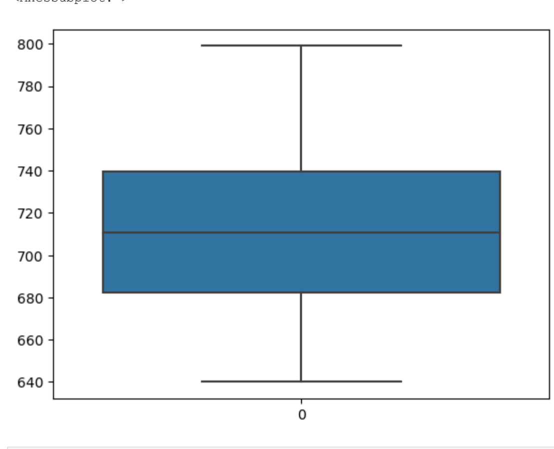
In [52]:

```
#removing outliers
for x in ameo['Marks_obtained']:
    if x >800 or x<640:
        ameo['Marks_obtained'].replace(x,np.nan,inplace=True)
```

In [53]:

```
#checking if outliers gone
sb.boxplot(ameo['Marks_obtained'])
```

Out[53]: `<AxesSubplot: >`



In [54]:

```
#replacing with median values
medmo=ameo['Marks_obtained'].median()
ameo['Marks_obtained'].replace(np.nan, medmo, inplace=True)
```

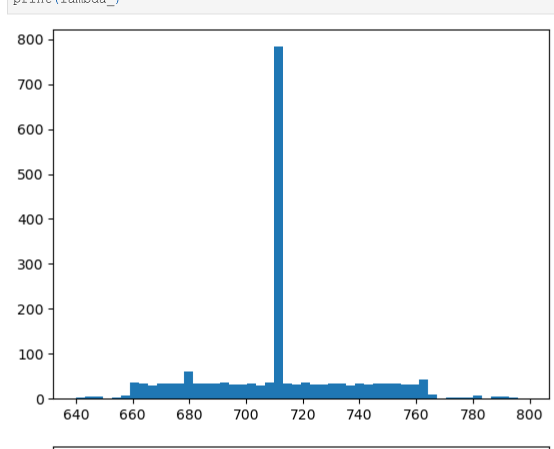
In [55]:

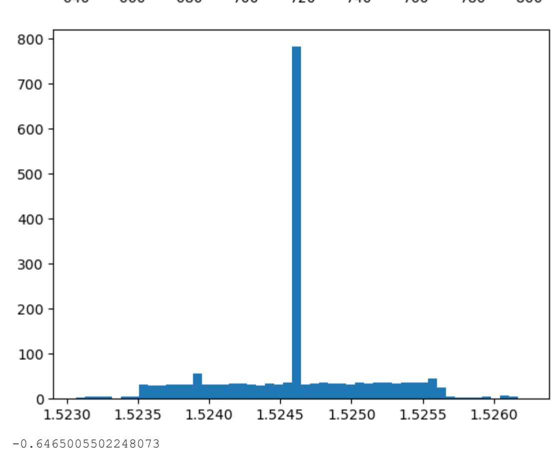
```
# perform Shapiro-Wilk test
stat, p = stats.shapiro(ameo['Marks_obtained'])

# interpret results
alpha = 0.05
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')
```

```
Data does not look Gaussian (reject H0)
```

In [56]:

```
# Doing box-cox test to determine transform method
data = ameo['Marks_obtained']


# plot histogram of data
plt.hist(data, bins=50)
plt.show()

# apply Box-Cox transformation to data
data_transformed, lambda_ = boxcox(data)

# plot histogram of transformed data
plt.hist(data_transformed, bins=50)
plt.show()

# print optimal lambda value
print(lambda_)
```

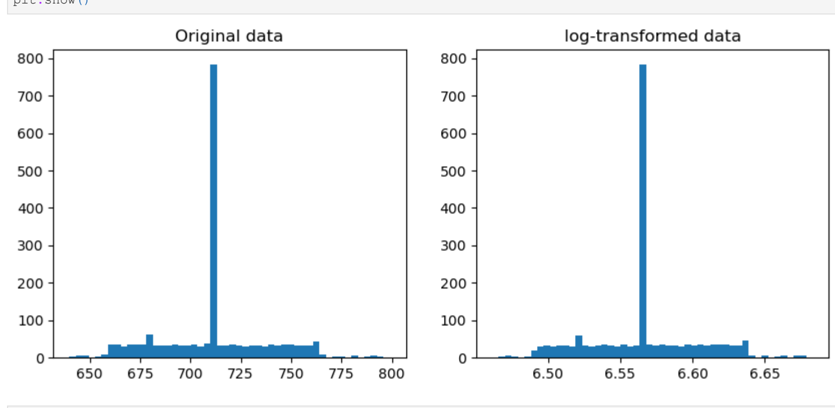




-0.6465005502248073

In [57]:

```
#log transforming the variable



# apply log transformation
ameo['Marks_obtainedtf']= np.log(ameo['Marks_obtained'])



fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].hist(data, bins=50)
axs[0].set_title('Original data')

axs[1].hist(ameo['Marks_obtainedtf'], bins=50)
axs[1].set_title('log-transformed data')



plt.show()
```



In [58]:

```
#removing the original
ameo=ameo.drop(['Marks_obtained'],axis=1)
ameo.head()
```

Out[58]:

	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	...	Domain	conscientiousness	agreeablene
0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	cbse	1141	...	0.635979	0.9737	0.81
1	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	cbse	5086	...	0.694479	-0.3027	-0.62
2	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	cbse	314	...	0.356536	1.7081	-0.10
3	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	cbse	403	...	0.765674	0.0464	0.81
4	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	cbse	2665	...	0.694479	0.1282	0.54

5 rows × 25 columns

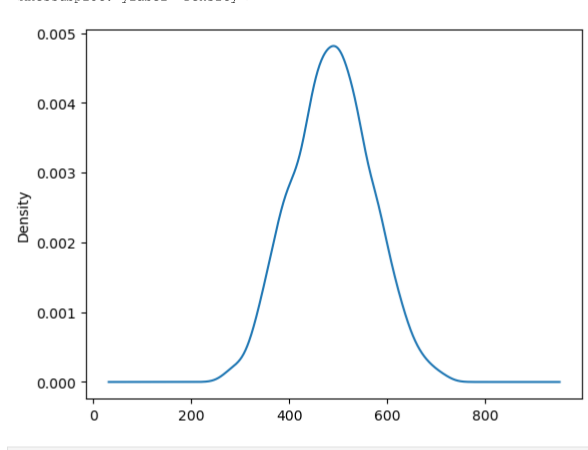
Remarks

The variable Domain was not skewed and had about 728 outliers which was removed and have been replaced with median. The boxcox value was -0.6 which required log transformation.

In [59]:

```
#checking distribution for aptitude_score
ameo['aptitude_score'].plot.density()
```

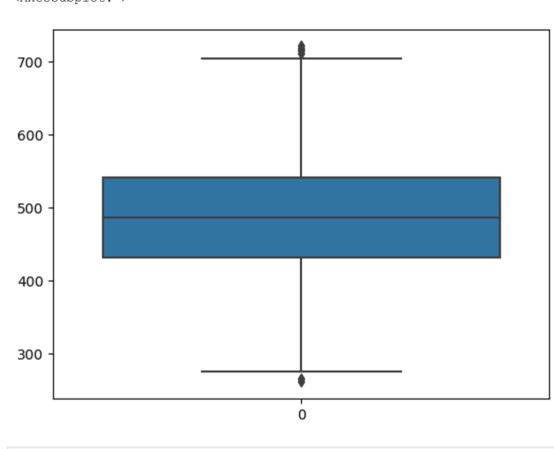
Out[59]: `<AxesSubplot: ylabel='Density'>`



In [60]:

```
#checking outliers using boxplot aptitude_score
sb.boxplot(ameo['aptitude_score'])
```

Out[60]: `<AxesSubplot: >`



In [61]:

```
#outlier treatment for variable aptitude_score
ameo['aptitude_score'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[61]:

```
0.10    380.000000
0.25    431.666667
0.50    486.666667
0.70    530.000000
0.90    591.666667
0.95    620.000000
0.99    671.666667
Name: aptitude_score, dtype: float64
```

In [62]:

```
#seeing limits
as_HE=ameo[ameo['aptitude_score'] >671.0].copy()
as_LE=ameo[ameo['aptitude_score'] <400.0].copy()

#putting them at one place
print(len(as_HE))
print(len(as_LE))
```

```
22
302
```

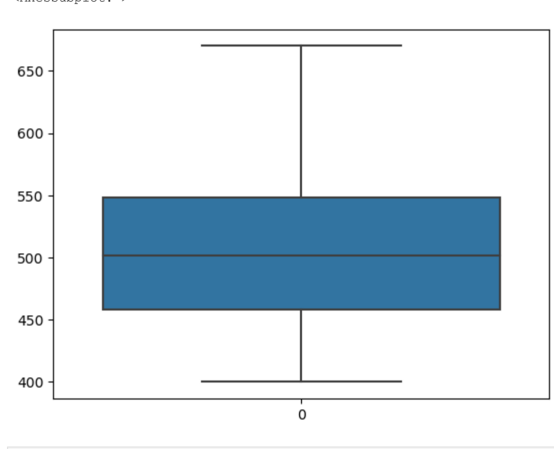
In [63]:

```
#removing outliers
for x in ameo['aptitude_score']:
    if x > 671.0 or x<400.0 :
        ameo['aptitude_score'].replace(x,np.nan,inplace=True)
```

In [64]:

```
#checking if outliers gone
sb.boxplot(ameo['aptitude_score'])
```

Out[64]: `<AxesSubplot: >`



In [65]:

```
#replacing with median values
medas=ameo['aptitude_score'].median()
ameo['aptitude_score'].replace(np.nan, medas, inplace=True)
```

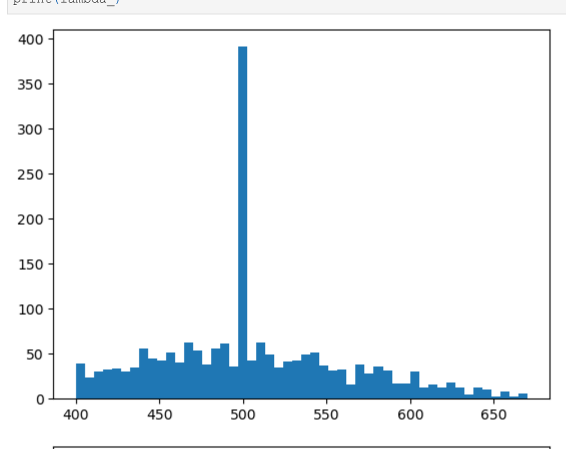
In [66]:

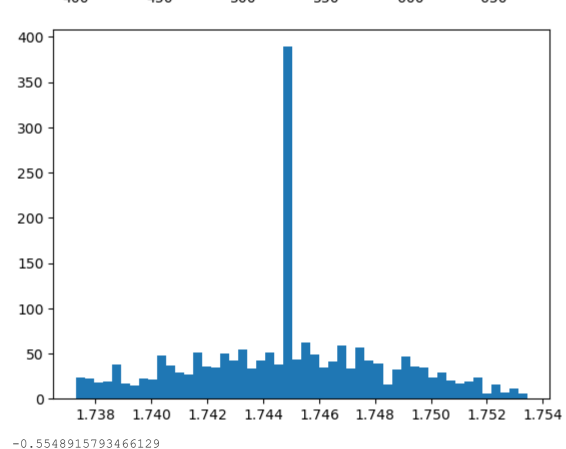
```
# perform Shapiro-Wilk test
stat, p = stats.shapiro(ameo['aptitude_score'])

# interpret results
alpha = 0.05
if p > alpha:
    print('Data looks Gaussian (fail to reject H0)')
else:
    print('Data does not look Gaussian (reject H0)')
```

```
Data does not look Gaussian (reject H0)
```

In [67]:

```
# Doing box-cox test to determine transform method
data = ameo['aptitude_score']


# plot histogram of data
plt.hist(data, bins=50)
plt.show()

# apply Box-Cox transformation to data
data_transformed, lambda_ = boxcox(data)

# plot histogram of transformed data
plt.hist(data_transformed, bins=50)
plt.show()

# print optimal lambda value
print(lambda_)
```

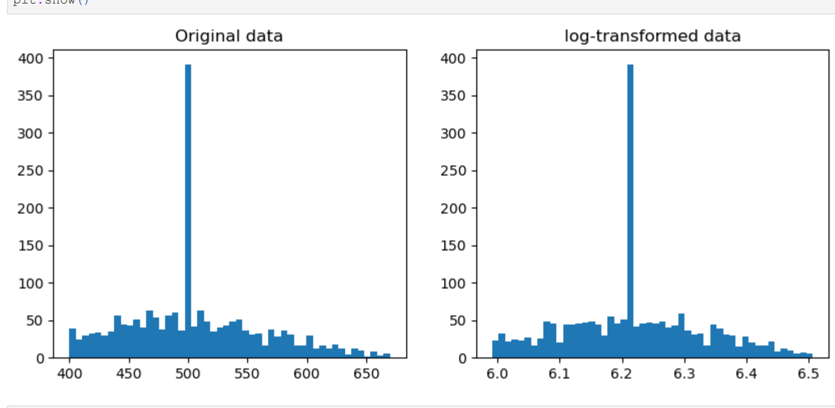




-0.5548915793466129

In [68]:

```
#log transforming the variable



# apply log transformation
ameo['aptitude_scoretf']= np.log(ameo['aptitude_score'])



fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].hist(data, bins=50)
axs[0].set_title('Original data')

axs[1].hist(ameo['aptitude_scoretf'], bins=50)
axs[1].set_title('log-transformed data')



plt.show()
```



In [69]:

```
#removing the original
ameo=ameo.drop(['aptitude_score'],axis=1)
ameo.head()
```

Out[69]:

	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	...	Domain	conscientiousness	agreeablene
0	2012.0	present	senior quality engineer	Bangalore	f	1990	2007	95.80	cbse	1141	...	0.635979	0.9737	0.81
1	2012.0	present	system engineer	Hyderabad	m	1992	2010	87.00	cbse	5086	...	0.694479	-0.3027	-0.62
2	2013.0	2015	java software engineer	Bangalore	m	1993	2010	67.50	cbse	314	...	0.356536	1.7081	-0.10
3	2014.0	2015	senior php developer	Bangalore	m	1989	2006	83.70	cbse	403	...	0.765674	0.0464	0.81
4	2014.0	present	quality assurance engineer	Pune	m	1990	2008	69.83	cbse	2665	...	0.694479	0.1282	0.54

5 rows × 25 columns

Remarks

The variable Domain was not skewed and had about 324 outliers which was removed and have been replaced with median. The boxcox value was -0.55 which required log transformation.

Step 3:Exploratory data analysis

In this step, we are going to explore the dataset. Perform hypothesis tests,bivariate analysis and check for correlation between variables.

In [70]:
```
#summary stats
ameo.describe().round(2)
```

Out[70]:

	percentage	CollegeID	CollegeTier	CollegeCityTier	Domain	conscientiousness	agreeableness	extraversion	nueroticism	openess_to_ex
count	1961.00	1961.00	1961.00	1961.00	1961.00	1961.00	1961.00	1961.00	1961.00	
mean	75.28	5696.33	1.97	0.36	0.69	0.06	0.10	0.04	-0.19	
std	12.00	4827.47	0.18	0.48	0.18	1.01	0.99	0.96	1.03	
min	43.42	17.00	1.00	0.00	0.30	-4.04	-5.78	-4.60	-2.64	
25%	66.00	1125.00	2.00	0.00	0.56	-0.57	-0.45	-0.49	-1.00	
50%	75.00	4609.00	2.00	0.00	0.69	0.13	0.21	0.14	-0.23	
75%	85.00	9481.00	2.00	1.00	0.82	0.82	0.81	0.67	0.53	
max	98.20	18139.00	2.00	1.00	1.00	2.00	1.90	2.32	3.35	

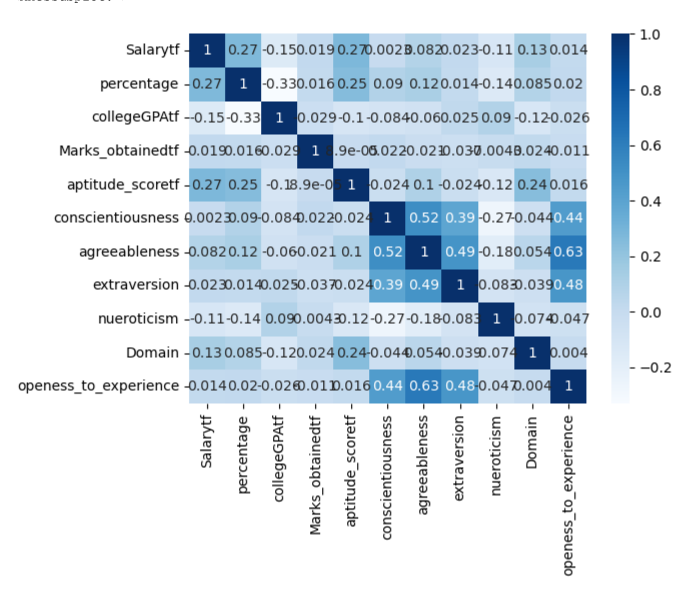
In [71]:
```
#seeing correlation of numerical variables
num_vars=['Salarytf',
 'percentage',
 'collegeGPAtf',
 'Marks_obtainedtf',
 'aptitude_scoretf',
 'conscientiousness',
 'agreeableness',
 'extraversion',
 'nueroticism',
 'Domain',
 'openess_to_experience']
cordata =ameo[num_vars].corr()
cordata.round(2)
```

Out[71]:

	Salarytf	percentage	collegeGPAtf	Marks_obtainedtf	aptitude_scoretf	conscientiousness	agreeableness	extraversion
Salarytf	1.00	0.27	-0.15	0.02	0.27	0.00	0.08	0.02
percentage	0.27	1.00	-0.33	0.02	0.25	0.09	0.12	0.01
collegeGPAtf	-0.15	-0.33	1.00	-0.03	-0.10	-0.08	-0.06	0.03
Marks_obtainedtf	0.02	0.02	-0.03	1.00	0.00	0.02	-0.02	-0.04
aptitude_scoretf	0.27	0.25	-0.10	0.00	1.00	-0.02	0.10	-0.02
conscientiousness	0.00	0.09	-0.08	0.02	-0.02	1.00	0.52	0.39
agreeableness	0.08	0.12	-0.06	-0.02	0.10	0.52	1.00	0.49
extraversion	0.02	0.01	0.03	-0.04	-0.02	0.39	0.49	1.00
nueroticism	-0.11	-0.14	0.09	-0.00	-0.12	-0.27	-0.18	-0.08
Domain	0.13	0.09	-0.12	0.02	0.24	-0.04	0.05	-0.04
openess_to_experience	0.01	0.02	-0.03	-0.01	0.02	0.44	0.63	0.48

In [72]:
```
#heatmap of correlation
sb.heatmap(cordata, annot=True, cmap="Blues")
```

Out[72]: <AxesSubplot: >

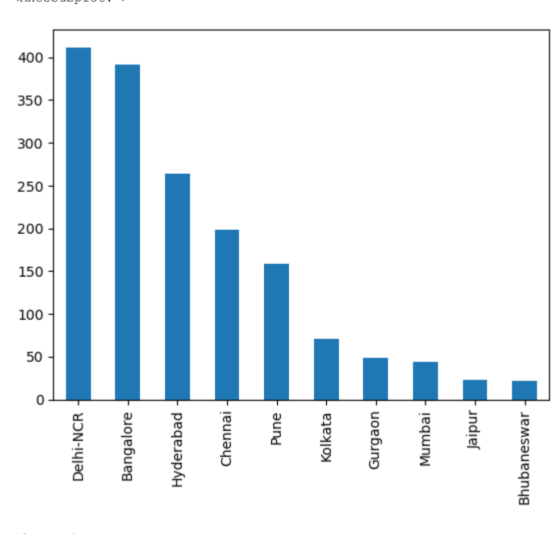


Observation

We see that the Salary has a moderately positive co-relation with the board exam percentage,Domain marks and the AMEO aptitude score. However the college GPA has a mild negative effect on the salary.

In [73]:
```
#seeing the top 10 JobCity
citycnt=ameo['JobCity'].value_counts().head(10)
citycnt.plot(kind = "bar")
```

Out[73]: <AxesSubplot: >

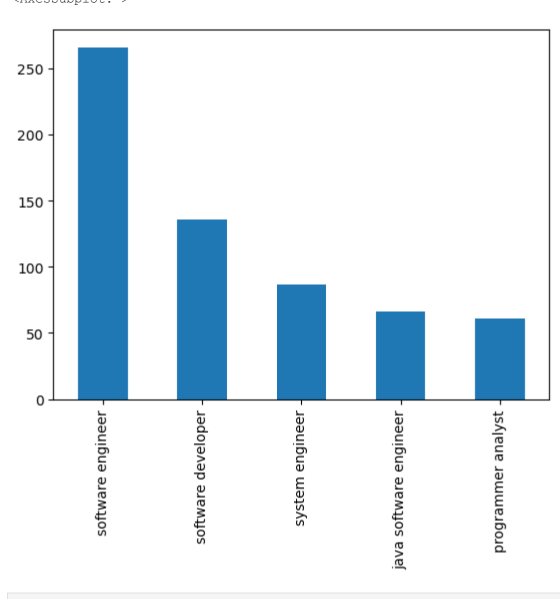


Observation

We find that Delhi-NCR region has the most jobs.

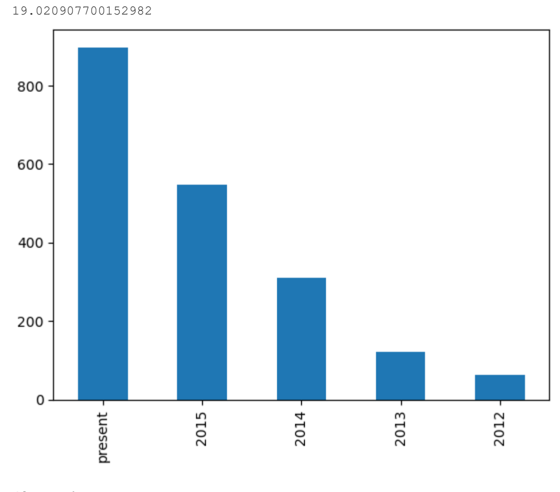
In [74]:
```
#seeing the top 5 Jobs
citycnt=ameo['Designation'].value_counts().head(5)
citycnt.plot(kind = "bar")
```

Out[74]: <AxesSubplot: >



In [75]:
```
#Seeing the retention rare
DOLcnt=ameo['DOL'].value_counts().head(5)
DOLcnt.plot(kind = "bar")

present=ameo[ameo['Specialization'] =='computer science & engineering']
retpercent=(len(present)/len(ameo))*100
print(retpercent)
```

19.020907700152982

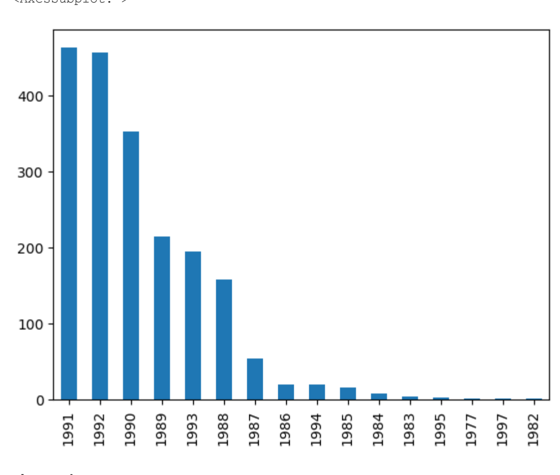


Observation

We can see that 1 in 5 employees is loyal to the company and stays on the job.

In [76]:
```
#seeing when are most people born
DOBcnt=ameo['DOB'].value_counts()
DOBcnt.plot(kind = "bar")
```

Out[76]: <AxesSubplot: >

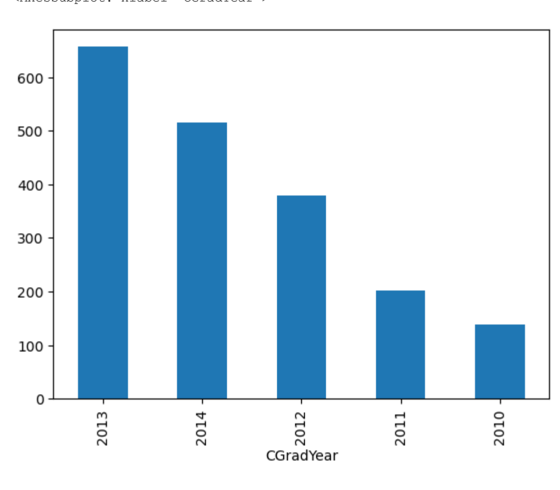


observation

We see that most people are late 80s to early 90s born kids and we can say that they saw better education and economy.

In [77]:
```
#does college completing year guarantee more jobs?
b_by_prod= ameo.groupby("CGradYear")["Designation"].count().nlargest(5)
b_by_prod.plot(kind = "bar")
```

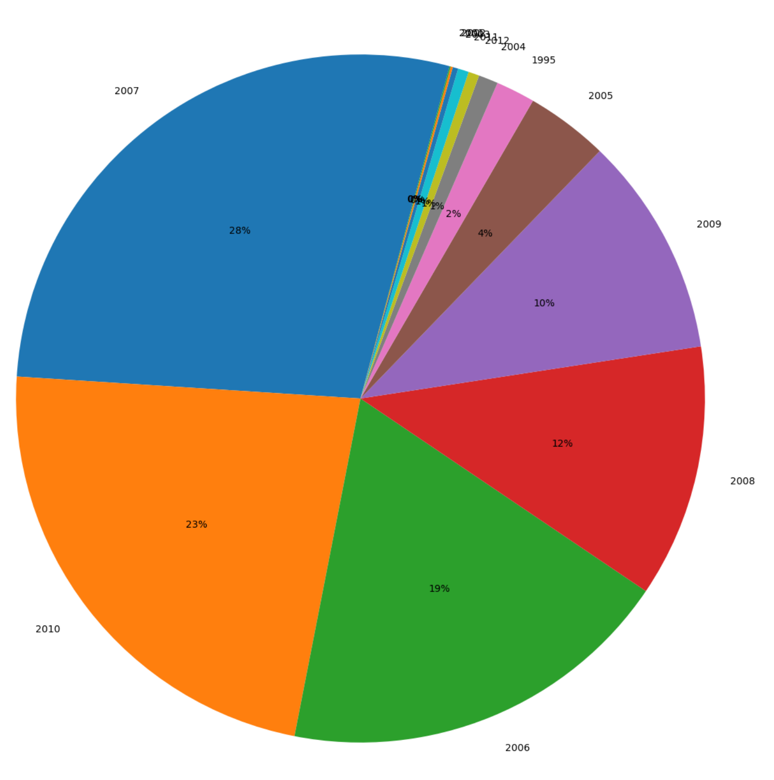
Out[77]: <AxesSubplot: xlabel='CGradYear'>



Observation

We can clearly infer from the graph above that the students graduating in 2013 and 2014 had more jobs then those graduating from college in 2010 and 2011.

In [78]:
```
#Which year did most people leave school
Scleave=ameo['Sgradyr'].unique()
numbers=ameo['Sgradyr'].value_counts()

plt.figure(figsize=(16, 16))
plt.pie(numbers, labels = Scleave, startangle = 75,autopct='%.0f%%')
plt.show()
```

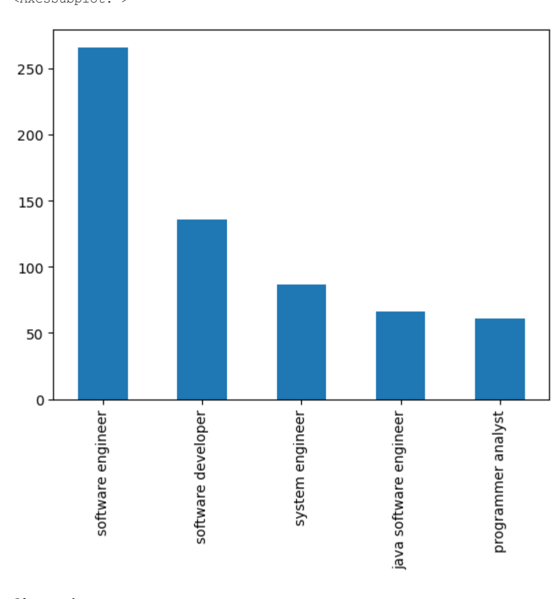


observation

We can infer from the above that 2010 and 2007 saw the most number of school pass outs.

In [79]:
```
#seeing the top 5 Jobs
citycnt=ameo['Designation'].value_counts().head(5)
citycnt.plot(kind = "bar")
```

Out[79]: <AxesSubplot: >

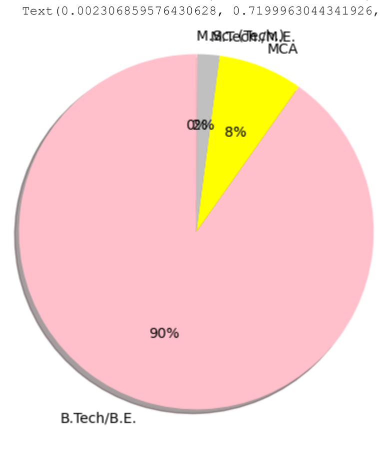


Observation

We see that software engineering is the most liked job followed by developer and programmer analyst.

In [80]:
```
#which type of Degree is the most common?

Degree=ameo['Degree'].unique()
number=ameo['Degree'].value_counts()


# color for each label
colors = ['pink', 'yellow', 'silver']

# plotting the pie chart
plt.pie(number, labels =
Degree, colors=colors,
        startangle=90, shadow = True,
        radius = 1.2, autopct = '%1.0f%%')
```

Out[80]:
```
([<matplotlib.patches.Wedge at 0x23bb7ad35e0>,
  <matplotlib.patches.Wedge at 0x23bb7ad3e50>,
  <matplotlib.patches.Wedge at 0x23bb7ae4760>,
  <matplotlib.patches.Wedge at 0x23bb7ad3580>],
 [Text(-0.39964775388707935, -1.2580467689291255, 'B.Tech/B.E.'),
  Text(0.48135938356043345, 1.2291025766217887, 'MCA'),
  Text(0.09085953844186184, 1.3168692206419481, 'M.Tech./M.E.'),
  Text(0.004229242556789485, 1.3199932247960198, 'M.Sc. (Tech.)')],
 [Text(-0.2179896839384069, -0.6862073285067957, '90%'),
  Text(0.26255966376023643, 0.6704195872482482, '8%'),
  Text(0.04955974824101555, 0.7182923021683353, '2%'),
  Text(0.002306859576430628, 0.7199963044341926, '0%')])
```



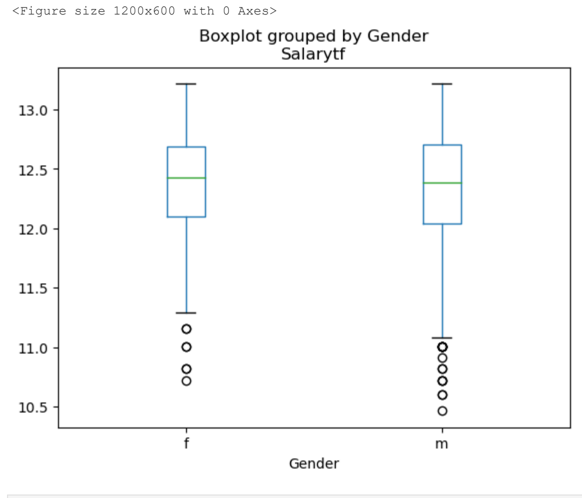
Conclusion

We see that B.Tech and BE are the most popular degrees as it is preferred by every 9 students out of 10. rest most preferred degree is MBA which is preferred by around 8%. Rarely people go for other degrees beyond them.

In [81]:
```
#box plot to see distribution of salary across Gender
plt.figure(figsize=[12,6])
ameo.boxplot(by="Gender",column="Salarytf", grid = False)
```

Out[81]: <AxesSubplot: title={'center': 'Salarytf'}, xlabel='Gender'>

<Figure size 1200x600 with 0 Axes>

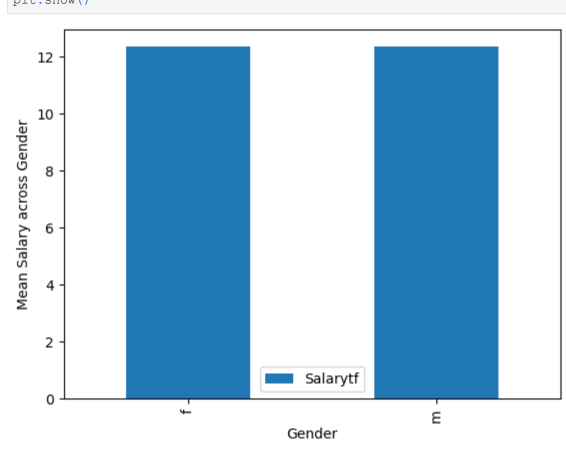


In [82]:
```
#checking if salary differs according to the Gender of the Employee
mod = ols("Salarytf ~ Gender", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[82]:

	df	sum_sq	mean_sq	F	PR(>F)
Gender	1.0	0.007024	0.007024	0.029436	0.863794
Residual	1959.0	467.463855	0.238624	NaN	NaN

In [83]:
```
#seeing the average Salary of Employee according to Gender
grouped = ameo.groupby('Gender').agg({'Salarytf': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Mean Salary across Gender')
plt.show()
```



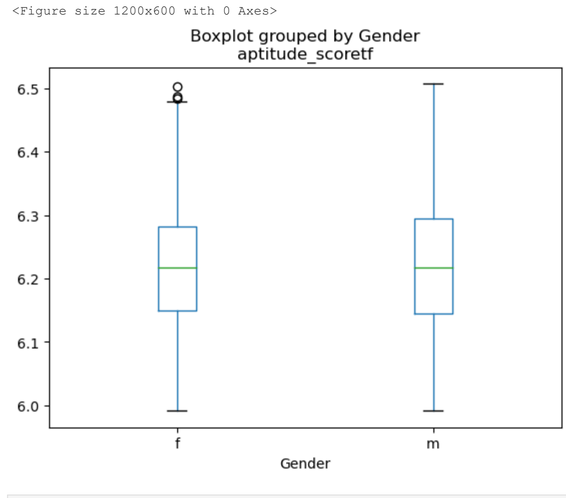
Observation

We see that there is no gender based pay gap between employees as the feminists allege all the time. The mean salary of both the groups are same hence we can say that there is no gender bias.

In [84]:
```
#box plot to see distribution of aptitude score across Gender
plt.figure(figsize=[12,6])
ameo.boxplot(by="Gender",column="aptitude_scoretf", grid = False)
```

Out[84]: <AxesSubplot: title={'center': 'aptitude_scoretf'}, xlabel='Gender'>

<Figure size 1200x600 with 0 Axes>

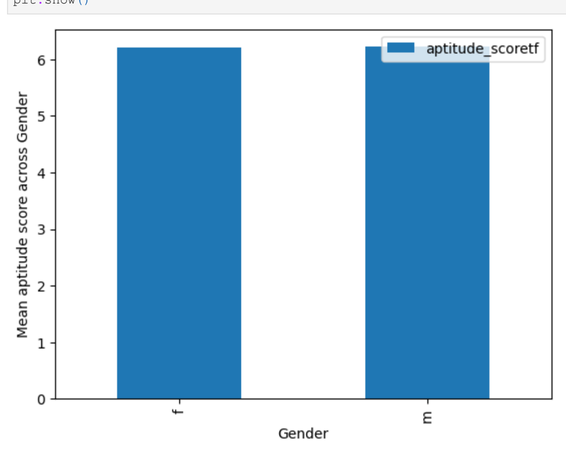


In [85]:
```
#checking if aptitude score differs according to the Gender of the Employee
mod = ols("aptitude_scoretf ~ Gender", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[85]:

	df	sum_sq	mean_sq	F	PR(>F)
Gender	1.0	0.014764	0.014764	1.212114	0.271049
Residual	1959.0	23.862132	0.012181	NaN	NaN

In [86]:
```
#seeing the average aptitude score of Employee according to Gender
grouped = ameo.groupby('Gender').agg({'aptitude_scoretf': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Mean aptitude score across Gender')
plt.show()
```



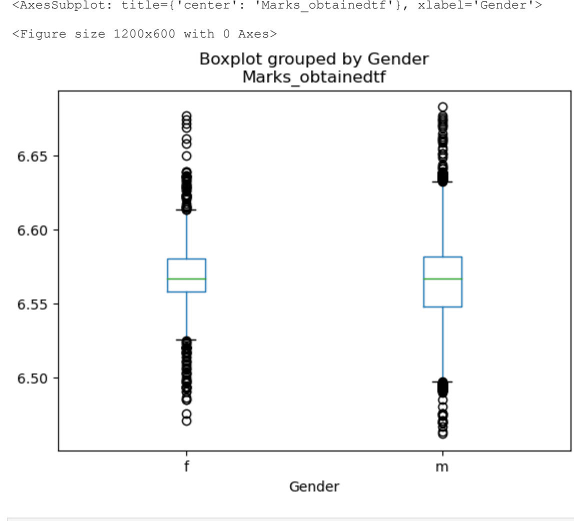
Observation

The aptitude score is same between male and female. Hence we can say that both male and female are having the same mental capacity.

In [87]:
```
#box plot to see distribution of Marks obtained differ across Gender
plt.figure(figsize=[12,6])
ameo.boxplot(by="Gender",column="Marks_obtainedtf", grid = False)
```

Out[87]: <AxesSubplot: title={'center': 'Marks_obtainedtf'}, xlabel='Gender'>

<Figure size 1200x600 with 0 Axes>

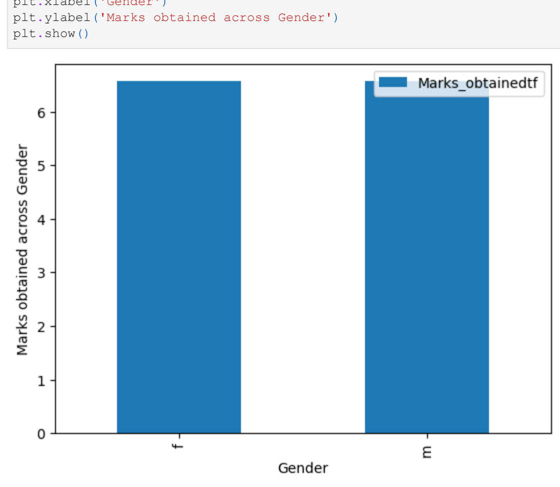


In [88]:
```
#checking if Marks differs according to the Gender of the Employee
mod = ols(" Marks_obtainedtf ~ Gender", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[88]:

	df	sum_sq	mean_sq	F	PR(>F)
Gender	1.0	0.001380	0.001380	1.006111	0.31596
Residual	1959.0	2.686031	0.001371	NaN	NaN

In [89]:
```
#seeing the average Marks obtained of Employee according to Gender
grouped = ameo.groupby('Gender').agg({'Marks_obtainedtf': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Marks obtained across Gender')
plt.show()
```



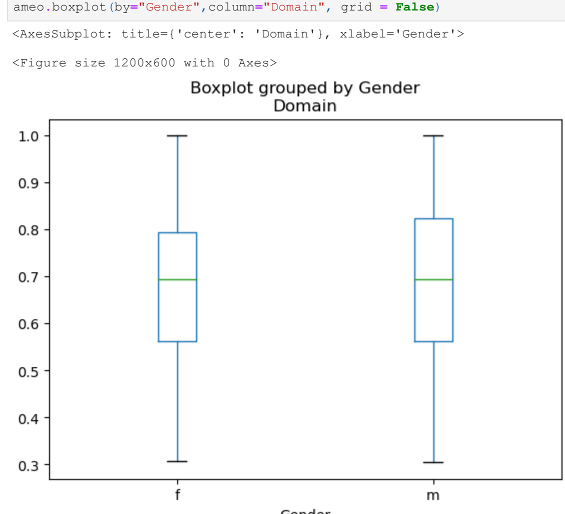
Observation

The marks obtained is almost the same between male and female employees so we cant say that men are better at their tests then women.

In [90]:
```
#box plot to see distribution of Domain score across Gender
plt.figure(figsize=[12,6])
ameo.boxplot(by="Gender",column="Domain", grid = False)
```

Out[90]: <AxesSubplot: title={'center': 'Domain'}, xlabel='Gender'>

<Figure size 1200x600 with 0 Axes>

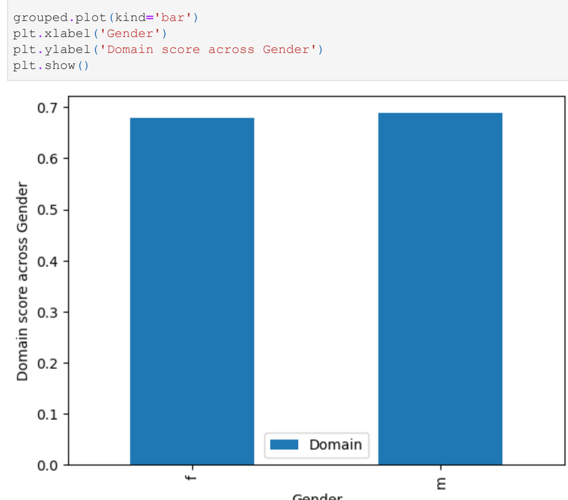


In [91]:
```
#checking if  Domain score differs according to the Gender of the Employee
mod = ols(" Domain ~ Gender", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[91]:

	df	sum_sq	mean_sq	F	PR(>F)
Gender	1.0	0.030046	0.030046	0.937396	0.333068
Residual	1959.0	62.791998	0.032053	NaN	NaN

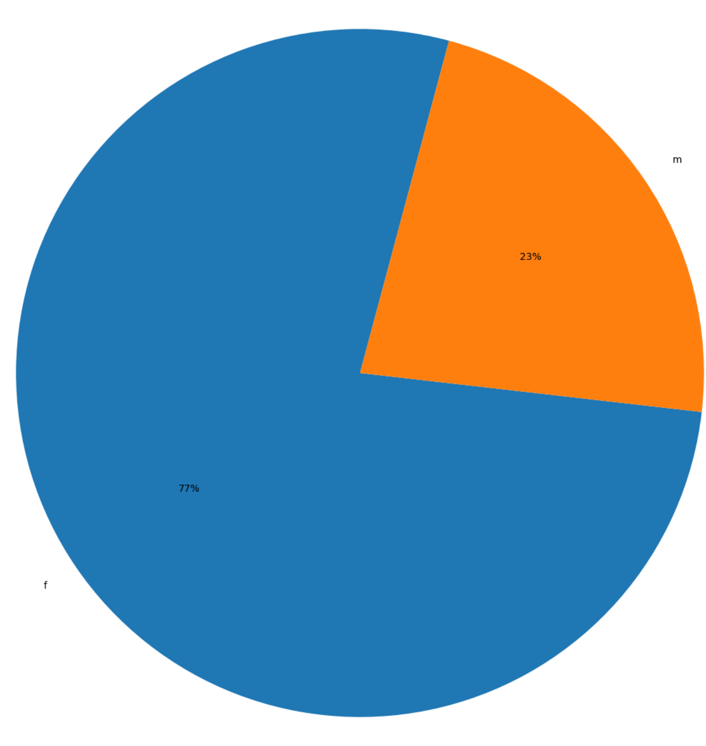
In [92]:
```
#seeing the Domain score of Employee according to Gender
grouped = ameo.groupby('Gender').agg({'Domain': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Domain score across Gender')
plt.show()
```



Observation

The domain score is almost the same between male and female employees so we cant say that men are better at their job then women.

In [93]:
```
#Composition of workforce based on Gender
gender=ameo['Gender'].unique()
numbers=ameo['Gender'].value_counts()

plt.figure(figsize=(16, 16))
plt.pie(numbers, labels = gender, startangle = 75,autopct='%.0f%%')
plt.show()
```

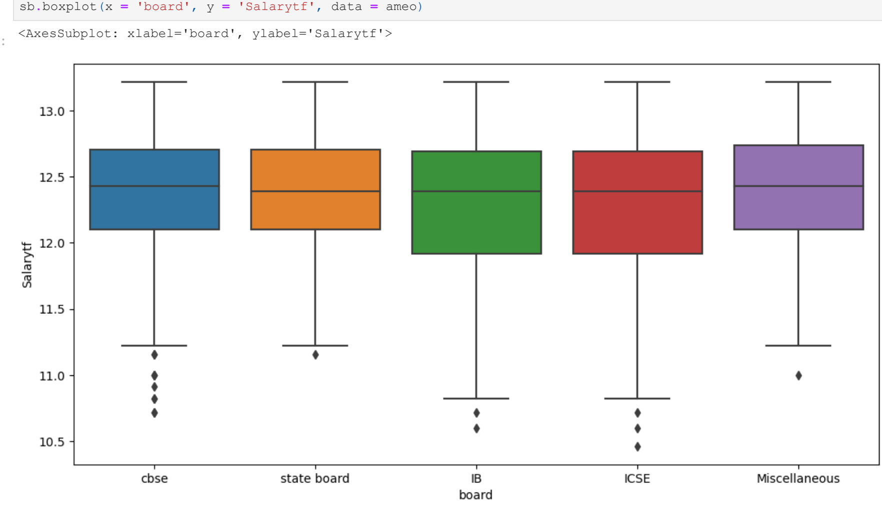


Observation

We can confirm that this is a sampling bias where more women were surveyed as comparison to men.

In [94]:
```
#box plot to see distribution of Salary differ across board
plt.figure(figsize=[12,6])


sb.boxplot(x = 'board', y = 'Salarytf', data = ameo)
```

Out[94]: <AxesSubplot: xlabel='board', ylabel='Salarytf'>

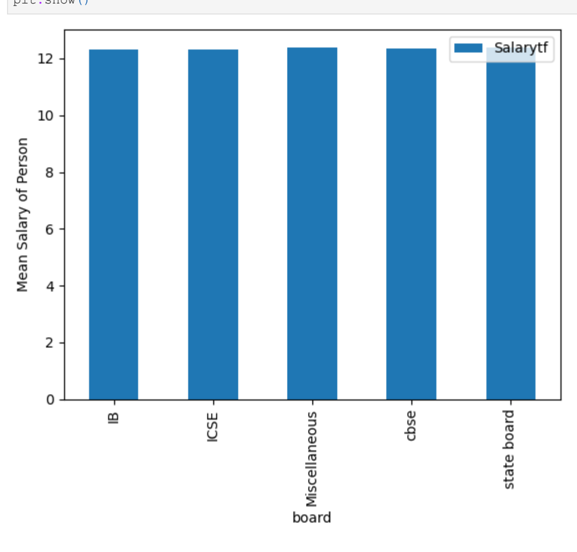


In [95]:
```
#checking if  Salary differs according to the  board of the Employee
mod = ols(" Salarytf ~board", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[95]:

	df	sum_sq	mean_sq	F	PR(>F)
board	4.0	1.901918	0.475480	1.997638	0.092375
Residual	1956.0	465.568961	0.238021	NaN	NaN

In [96]:
```
#seeing the average salary of person according to board
grouped = ameo.groupby('board').agg({'Salarytf': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('board')
plt.ylabel('Mean Salary of Person')
plt.show()
```



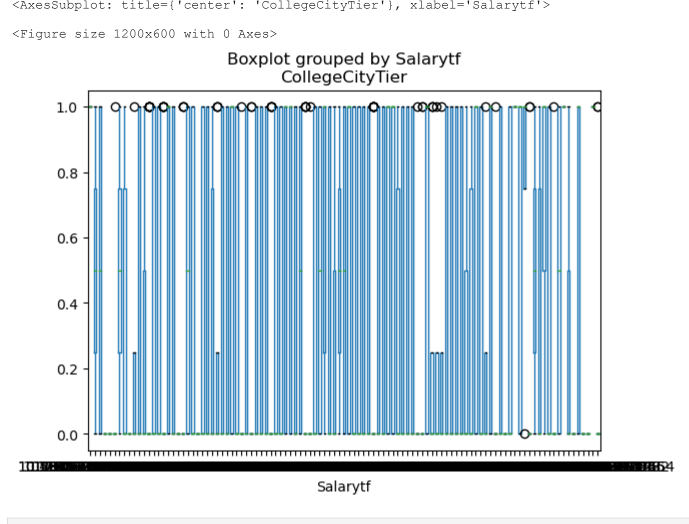
Observation

We can see that there is no significant difference in average salary, hence we can say that the board a person studied from is not a deciding actor of the amount of salary they get.

In [97]:
```
#box plot to see distribution of Salary across CollegeCityTier
plt.figure(figsize=[12,6])
ameo.boxplot(by="Salarytf",column="CollegeCityTier", grid = False)
```

Out[97]: <AxesSubplot: title={'center': 'CollegeCityTier'}, xlabel='Salarytf'>

<Figure size 1200x600 with 0 Axes>

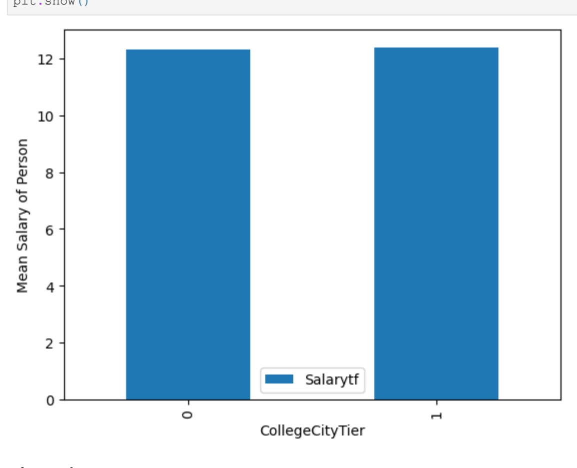


In [98]:
```
#checking if  Salary differs according to the  CollegeCityTier of the Employee
mod = ols(" Salarytf ~CollegeCityTier", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[98]:

	df	sum_sq	mean_sq	F	PR(>F)
CollegeCityTier	1.0	3.044109	3.044109	12.840365	0.000348
Residual	1959.0	464.426770	0.237073	NaN	NaN

In [99]:
```
#seeing the average salary of person according to CollegeCityTier
grouped = ameo.groupby('CollegeCityTier').agg({'Salarytf': 'mean'})

grouped.plot(kind='bar')
plt.xlabel('CollegeCityTier')
plt.ylabel('Mean Salary of Person')
plt.show()
```



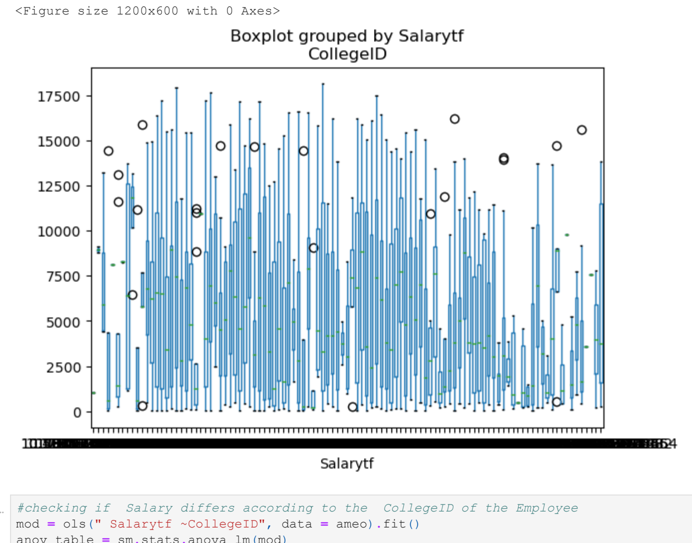
observation

It can be clearly seen that Tier one cities college allow students to get a higher paying job.

In [100]:
```
#box plot to see distribution of Salary across CollegeID
plt.figure(figsize=[12,6])
ameo.boxplot(by="Salarytf",column="CollegeID", grid = False)
```

Out[100]: <AxesSubplot: title={'center': 'CollegeID'}, xlabel='Salarytf'>

<Figure size 1200x600 with 0 Axes>

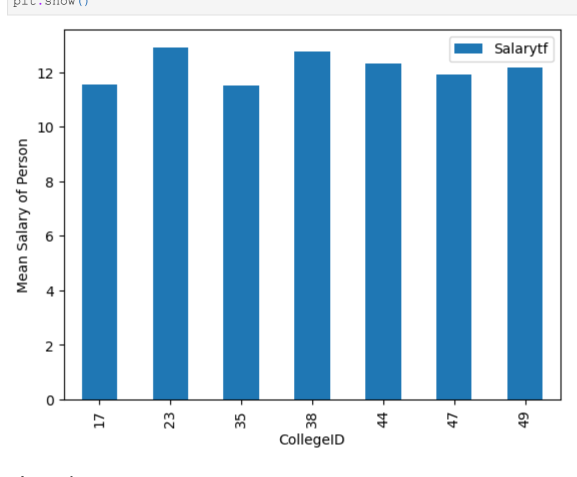


In [101]:
```
#checking if  Salary differs according to the  CollegeID of the Employee
mod = ols(" Salarytf ~CollegeID", data = ameo).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[101]:

	df	sum_sq	mean_sq	F	PR(>F)
CollegeID	1.0	4.876104	4.876104	20.649362	0.000006
Residual	1959.0	462.594776	0.236138	NaN	NaN

In [102]:
```
#seeing the average salary of person according to top 7 CollegeID
grouped = ameo.groupby('CollegeID').agg({'Salarytf': 'mean'}).head(7)

grouped.plot(kind='bar')
plt.xlabel('CollegeID')
plt.ylabel('Mean Salary of Person')
plt.show()
```



observation

We see that the colleges 17,23,35,38,44,47 and 49 offer the best salaries to their students post graduation.

In [103]:
```
#gender differences in personality
male=ameo[ameo['Gender']=='m']
female=ameo[ameo['Gender']=='f']
```

In [104]:
```
#taking only the necessary columns
malepersona=male[['conscientiousness','agreeableness','extraversion','nueroticism','openess_to_experience']]
femalepersona=female[['conscientiousness','agreeableness','extraversion','nueroticism','openess_to_experience']]
```

In [105]:
```
#summary stats for male
malepersona.describe().round(2)
```

Out[105]:

	conscientiousness	agreeableness	extraversion	nueroticism	openess_to_experience
count	1517.00	1517.00	1517.00	1517.00	1517.00
mean	0.00	0.05	0.04	-0.18	-0.18
std	1.04	1.01	0.97	1.02	1.09
min	-4.04	-5.78	-4.60	-2.64	-7.38
25%	-0.59	-0.45	-0.49	-0.88	-0.67
50%	0.05	0.19	0.09	-0.17	-0.09
75%	0.74	0.71	0.67	0.53	0.54
max	2.00	1.90	2.32	3.35	1.63

In [106]:
```
#summary stats for female
femalepersona.describe().round(2)
```

Out[106]:

	conscientiousness	agreeableness	extraversion	nueroticism	openess_to_experience
count	444.00	444.00	444.00	444.00	444.00
mean	0.25	0.28	0.06	-0.23	0.02
std	0.89	0.89	0.91	1.05	0.94
min	-3.35	-5.12	-3.37	-2.64	-5.69
25%	-0.30	-0.28	-0.45	-1.00	-0.48
50%	0.36	0.38	0.16	-0.29	0.10
75%	0.85	0.88	0.67	0.53	0.66
max	2.00	1.75	2.16	2.76	1.63

Observation

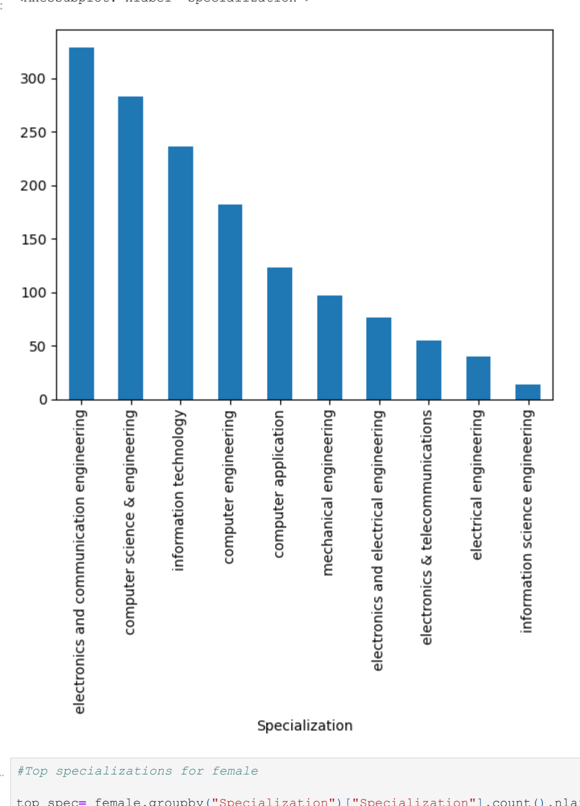
We can conclude that males tend to be more neurotic on average than females. Females on the other hand score higher on conscientiousness,agreeableness,extraversion and they are more open to experiences than their male counterparts.

In [107]:
```
#Seeing if specialization has different number of males and females
contingency_table = pd.crosstab(ameo['Gender'], ameo['Specialization'])

# run the chi-squared test
stat, p, dof, expected = chi2_contingency(contingency_table)

# print the results
print("Chi-squared statistic: ", stat)
print("p-value: ", p)
print("Degrees of freedom: ", dof)
print("Expected frequencies: \n", expected)
```

```
Chi-squared statistic:  55.252020188620804
p-value:  0.04398206049973966
Degrees of freedom:  39
Expected frequencies:
 [[6.79245283e-01 1.13207547e+00 6.79245283e-01 1.13207547e+00
  6.79245283e-01 3.16981132e+00 3.41886792e+01 5.77358491e+01
  2.26415094e-01 4.52830189e-01 8.84528302e+01 6.79245283e-01
  2.26415094e-01 1.13207547e+00 2.26415094e-01 2.94339623e+00
  1.47169811e+01 9.69056604e+01 6.79245283e-01 2.08301887e+01
  1.13207547e+00 3.39622642e+00 2.26415094e-01 2.26415094e-01
  4.52830189e-01 2.26415094e-01 2.26415094e-01 4.30188679e+00
  7.08679245e+01 1.35849057e+00 4.52830189e-01 2.26415094e-01
  2.26415094e-01 2.35471698e+01 4.52830189e-01 2.26415094e-01
  1.81132075e+00 2.26415094e-01 2.26415094e-01 1.13207547e+00]
 [2.32075472e+00 3.86792453e+00 2.32075472e+00 3.86792453e+00
  2.32075472e+00 1.08301887e+01 1.16811321e+02 1.97264151e+02
  7.73584906e-01 1.54716981e+00 2.88547170e+02 2.32075472e+00
  7.73584906e-01 3.86792453e+00 7.73584906e-01 1.00566038e+01
  5.02830189e+01 3.31094340e+02 2.32075472e+00 7.11698113e+01
  3.86792453e+00 1.16037736e+01 7.73584906e-01 7.73584906e-01
  1.54716981e+00 7.73584906e-01 7.73584906e-01 1.46981132e+01
  2.42132075e+02 4.64150943e+00 1.54716981e+00 7.73584906e-01
  7.73584906e-01 8.04528302e+01 1.54716981e+00 7.73584906e-01
  6.18867925e+00 7.73584906e-01 7.73584906e-01 3.86792453e+00]]
```

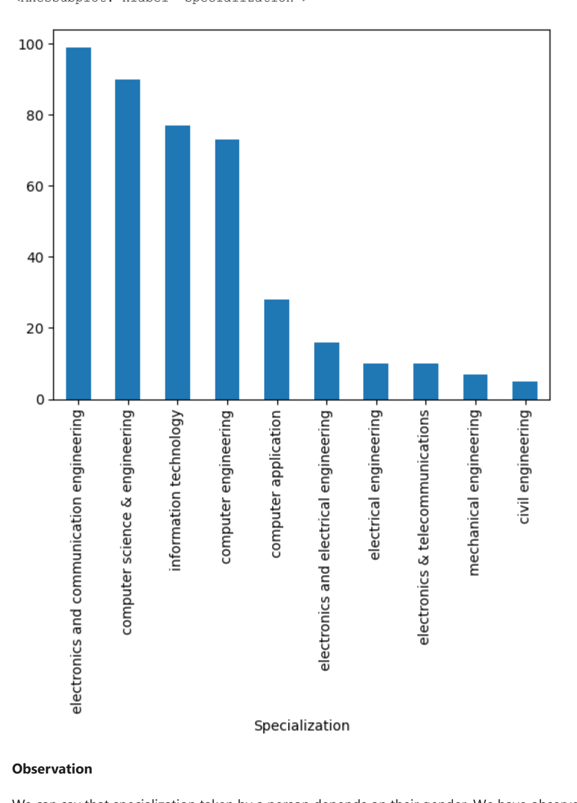
In [108]:
```
#Top specializations for male

top_spec= male.groupby("Specialization")["Specialization"].count().nlargest(10)
top_spec.plot(kind = "bar")
```

Out[108]: <AxesSubplot: xlabel='Specialization'>



In [109]:
```
#Top specializations for female

top_spec= female.groupby("Specialization")["Specialization"].count().nlargest(10)
top_spec.plot(kind = "bar")
```

Out[109]: <AxesSubplot: xlabel='Specialization'>



Observation

We can say that specialization taken by a person depends on their gender. We have observed earlier that there is significant personality difference between males and females which can be a factor of them choosing different specializations.

In [110]:
```
#taking subset for main problem statement


job_list = ['software engineer', 'hardware engineer', 'programmer analyst','associate engineer']
subset = ameo[(ameo['Designation'].isin(job_list)) & (ameo['Specialization'] =='computer science & engineering'
subset.head()
```

Out[110]:

	DOJ	DOL	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	...	Domain	conscientiousness	agreeable
10	2013.0	2015	programmer analyst	Hyderabad	m	1993	2010	90.00	cbse	9173	...	0.356536	0.4155	0
14	2013.0	2015	software engineer	Bangalore	m	1992	2010	86.10	cbse	7282	...	0.622643	-0.0154	1
27	2015.0	present	software engineer	Mangalore	m	1992	2010	90.10	cbse	12832	...	0.486747	-0.3027	0
92	2011.0	present	software engineer	Bangalore	m	1991	2008	70.16	cbse	2353	...	0.563268	-2.1175	-2
120	2013.0	present	software engineer	Hyderabad	m	1992	2009	90.70	cbse	15854	...	0.694479	-2.1698	-0

5 rows × 25 columns

In [111]:
```
#Doing the hypothesis test for the newspaper claim

# set the lower and upper bounds for the salary values
lower_bound = np.log(250000)
upper_bound = np.log(300000)

# perform the one-sample t-test
result = ttest_1samp(subset['Salarytf'].mean(), (lower_bound + upper_bound) / 2)

# print the result
if result.pvalue < 0.05:
    print("The mean salary is significantly different from the midpoint between the lower and upper bounds.")
else:
    print("The mean salary is not significantly different from the midpoint between the lower and upper bounds.
```

```
The mean salary is not significantly different from the midpoint between the lower and upper bounds.
C:\Users\sujoydutta\AppData\Local\Temp\ipykernel_12136\440127959.py:8: RuntimeWarning: Precision loss occurred
in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results
may be unreliable.
  result = ttest_1samp(subset['Salarytf'].mean(), (lower_bound + upper_bound) / 2)
C:\Users\sujoydutta\anaconda3\lib\site-packages\scipy\stats\_stats_py.py:1253: RuntimeWarning: divide by zero e
ncountered in divide
  var *= np.divide(n, n-ddof)  # to avoid error on division by zero
C:\Users\sujoydutta\anaconda3\lib\site-packages\scipy\stats\_stats_py.py:1253: RuntimeWarning: invalid value en
countered in double_scalars
  var *= np.divide(n, n-ddof)  # to avoid error on division by zero
```

Observation

We can conclude that the speculation by the newspaper was right. The mean salary is falling between the range of 2.5 lakhs and 3 lakhs.

Step 4: Model Building

After the data has been cleaned and formatted. Now its time to analyse and get insights. We will uuse Linear Regression to get the factors affecting salary of the person.

In [151]:
```
ameo.dtypes
```

Out[151]:
```
DOJ                       object
DOL                       object
Designation               object
JobCity                   object
Gender                    object
DOB                       object
Sgradyr                   object
percentage               float64
board                     object
CollegeID                  int64
CollegeTier                int64
Degree                    object
Specialization            object
CollegeCityTier            int64
CGradYear                 object
Domain                   float64
conscientiousness        float64
agreeableness            float64
extraversion             float64
nueroticism              float64
openess_to_experience    float64
collegeGPAtf             float64
Marks_obtainedtf         float64
aptitude_scoretf         float64
dtype: object
```

In [113]:
```
#creating target variable and dropping from main
target=ameo['Salarytf']
ameo= ameo.drop(['Salarytf'],axis=1)
```

In [152]:
```
# Identify columns containing non-numeric data
cat_cols = ameo.select_dtypes(include=['object']).columns.tolist()
cat_cols
```

Out[152]:
```
['DOJ',
 'DOL',
 'Designation',
 'JobCity',
 'Gender',
 'DOB',
 'Sgradyr',
 'board',
 'Degree',
 'Specialization',
 'CGradYear']
```

In [156]:
```
# Apply LabelEncoder to categorical columns
le = LabelEncoder()

#encoding categorical variables

ameo['Designation']=le.fit_transform(ameo['Designation'])
ameo['JobCity']=le.fit_transform(ameo['JobCity'])
ameo['Gender']=le.fit_transform(ameo['Gender'])
ameo['Sgradyr']=le.fit_transform(ameo['Sgradyr'])
ameo['board']=le.fit_transform(ameo['board'])
ameo['Degree']=le.fit_transform(ameo['Degree'])
ameo['Specialization']=le.fit_transform(ameo['Specialization'])
ameo['CGradYear']=le.fit_transform(ameo['CGradYear'])
ameo['DOB']=le.fit_transform(ameo['DOB'])
```

In [167]:
```
#dropping the unnecess
ameo=ameo.drop(['DOJ','DOL'],axis=1)
ameo.head()
```

Out[167]:

	Designation	JobCity	Gender	DOB	Sgradyr	percentage	board	CollegeID	CollegeTier	Degree	...	CGradYear	Domain	conscientiousnes
0	237	10	0	9	7	95.80	3	1141	2	0	...	3	0.635979	0.973
1	279	53	1	11	10	87.00	3	5086	2	0	...	6	0.694479	-0.302
2	127	10	1	12	10	67.50	3	314	2	0	...	6	0.356536	1.708
3	235	10	1	8	6	83.70	3	403	2	0	...	2	0.765674	0.046
4	197	104	1	9	8	69.83	3	2665	2	0	...	5	0.694479	0.128

5 rows × 22 columns

In [168]:
```
# Create linear regression object
regressor = LinearRegression()
```

In [169]:
```
#checking missing values

ameo.isnull().sum()
```

Out[169]:
```
Designation              0
JobCity                  0
Gender                   0
DOB                      0
Sgradyr                  0
percentage               0
board                    0
CollegeID                0
CollegeTier              0
Degree                   0
Specialization           0
CollegeCityTier          0
CGradYear                0
Domain                   0
conscientiousness        0
agreeableness            0
extraversion             0
nueroticism              0
openess_to_experience    0
collegeGPAtf             0
Marks_obtainedtf         0
aptitude_scoretf         0
dtype: int64
```

In [170]:
```
#setting values of X and y
X=ameo.values
y= target.values
```

In [171]:
```
# Train, test, split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = .2, random_state= 123)
```

In [180]:
```
#initializing lazy regressor to see all possible regression models
reg= LazyRegressor(verbose=0,ignore_warnings=False, custom_metric=None)
models, predictions= reg.fit(X_train, X_test, y_train, y_test)
```

```
100%|██████████████████████████████████████████████████| 42/42 [01:08<00:00,
1.62s/it]
```

In [181]:
```
#seeing which model will be the most accurate
print(models)
```

```
                               Adjusted R-Squared  R-Squared  RMSE  Time Taken
Model
RandomForestRegressor                        0.16       0.21  0.44        1.45
LGBMRegressor                                0.16       0.20  0.44        0.41
GradientBoostingRegressor                    0.15       0.20  0.44        0.42
HistGradientBoostingRegressor                0.12       0.17  0.44        0.63
ExtraTreesRegressor                          0.12       0.17  0.45        0.94
BayesianRidge                                0.10       0.15  0.45        0.15
RidgeCV                                      0.09       0.15  0.45        0.02
Ridge                                        0.09       0.15  0.45        0.01
TransformedTargetRegressor                   0.09       0.15  0.45        0.01
LinearRegression                             0.09       0.15  0.45        0.01
Lars                                         0.09       0.15  0.45        0.09
PoissonRegressor                             0.09       0.15  0.45        0.01
SGDRegressor                                 0.09       0.15  0.45        0.02
LassoLarsIC                                  0.09       0.15  0.45        0.02
ElasticNetCV                                 0.09       0.14  0.45        0.11
LassoCV                                      0.09       0.14  0.45        0.09
LarsCV                                       0.09       0.14  0.45        0.05
LassoLarsCV                                  0.09       0.14  0.45        0.05
OrthogonalMatchingPursuitCV                  0.09       0.14  0.45        0.02
HuberRegressor                               0.08       0.13  0.45        0.03
TweedieRegressor                             0.08       0.13  0.46        0.11
GammaRegressor                               0.08       0.13  0.46        0.08
NuSVR                                        0.07       0.13  0.46        0.33
LinearSVR                                    0.07       0.12  0.46        0.06
SVR                                          0.06       0.11  0.46        0.31
BaggingRegressor                             0.06       0.11  0.46        0.16
AdaBoostRegressor                            0.06       0.11  0.46        0.45
OrthogonalMatchingPursuit                    0.04       0.09  0.46        0.01
XGBRegressor                                 0.02       0.08  0.47        0.60
DummyRegressor                              -0.06      -0.00  0.49        0.01
ElasticNet                                  -0.06      -0.00  0.49        0.10
Lasso                                       -0.06      -0.00  0.49        0.03
LassoLars                                   -0.06      -0.00  0.49        0.02
KNeighborsRegressor                         -0.07      -0.01  0.49        0.10
QuantileRegressor                           -0.07      -0.01  0.49       59.18
PassiveAggressiveRegressor                  -0.39      -0.32  0.56        0.02
ExtraTreeRegressor                          -0.56      -0.47  0.59        0.02
DecisionTreeRegressor                       -0.68      -0.58  0.61        0.03
RANSACRegressor                             -1.15      -1.03  0.70        0.15
MLPRegressor                                -2.52      -2.33  0.89        1.40
GaussianProcessRegressor                  -585.73    -552.80 11.49        0.39
KernelRidge                               -677.28    -639.22 12.35        0.13
```

Observation

We have to use the Random forest Regressor model as it has the highest R squared value and the least RMSE.

In [184]:
```
# Create the random forest regression model
rfreg = RandomForestRegressor(n_estimators=100, random_state=1)
```

In [186]:
```
# Fit model to training data
rfreg.fit(X_train,y_train)
```

Out[186]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=1)
```

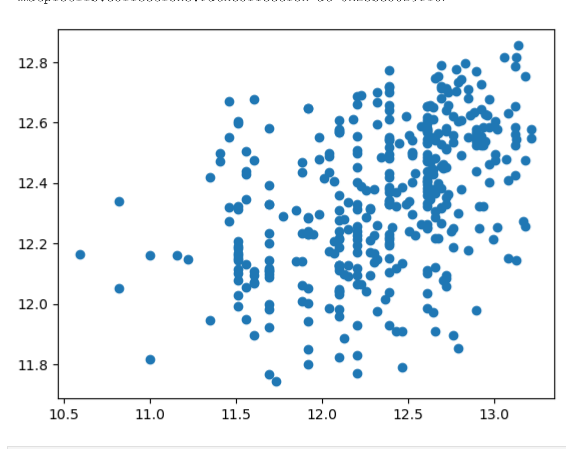
In [187]:
```
# Make predictions on the test data
y_pred = rfreg.predict(X_test)
```

In [188]:
```
# Evaluate the model using the R-squared score
r2 = r2_score(y_test, y_pred)
print('R-squared score:', r2)
```

```
R-squared score: 0.1891325546024577
```

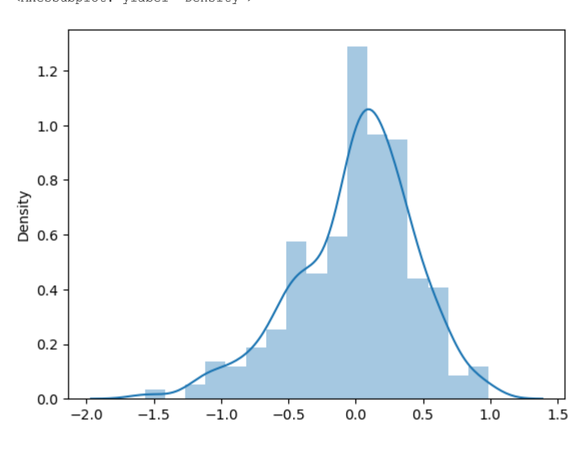
In [189]:
```
# Actual v predictions scatter
plt.scatter(y_test, y_pred)
```

Out[189]: <matplotlib.collections.PathCollection at 0x23bc5829f10>



In [190]:
```
# Histogram of the distribution of residuals
sb.distplot((y_test - y_pred))
```

Out[190]: <AxesSubplot: ylabel='Density'>



Conclusion

The model we developed can only predict 18% of the change in the dependent variable and it means we need more information and new features that can perhaps explain the change in the dependent variable better.

In [ ]: