Import Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler
from scipy import stats
```

In [ ]:
```
import warnings

# Ignore all warnings
warnings.filterwarnings("ignore")
```

Data Ingestion

In [ ]:
```
df=pd.read_excel('data/ameo.xlsx').copy()
df=df.iloc[:,1:]
df.head()
```

Out[ ]:

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

5 rows × 38 columns

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

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

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

Out[ ]:

	ID	Salary	DOJ	DOB	10percentage	12graduation	12percentage	CollegeID	CollegeTier	collegeGPA	...	Com
count	3.998000e+03	3.998000e+03	3998	3998	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	...	
mean	6.637945e+05	3.076998e+05	2013-07-02 11:04:10.325162496	1990-12-06 06:01:15.637819008	77.925443	2008.087544	74.466366	5156.851426	1.925713	71.486171	...	
min	1.124400e+04	3.500000e+04	1991-06-01 00:00:00	1977-10-30 00:00:00	43.000000	1995.000000	40.000000	2.000000	1.000000	6.450000	...	
25%	3.342842e+05	1.800000e+05	2012-10-01 00:00:00	1989-11-16 06:00:00	71.680000	2007.000000	66.000000	494.000000	2.000000	66.407500	...	
50%	6.396000e+05	3.000000e+05	2013-11-01 00:00:00	1991-03-07 12:00:00	79.150000	2008.000000	74.400000	3879.000000	2.000000	71.720000	...	
75%	9.904800e+05	3.700000e+05	2014-07-01 00:00:00	1992-03-13 18:00:00	85.670000	2009.000000	82.600000	8818.000000	2.000000	76.327500	...	
max	1.298275e+06	4.000000e+06	2015-12-01 00:00:00	1997-05-27 00:00:00	97.760000	2013.000000	98.700000	18409.000000	2.000000	99.930000	...	
std	3.632182e+05	2.127375e+05	NaN	NaN	9.850162	1.653599	10.999933	4802.261482	0.262270	8.167338	...	

8 rows × 29 columns

Data Cleaning And Transformation

Handling duplicates

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
0
```

- There are no duplicates

Handling missing values

In [ ]:
```
df.isna().sum()
```

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

- There are no missing values

Handling columns

Salary

In [ ]:
```
df['Salary']
```

Out[ ]:
```
0        420000
1        500000
2        325000
3       1100000
4        200000
          ...
3993     280000
3994     100000
3995     320000
3996     200000
3997     400000
Name: Salary, Length: 3998, dtype: int64
```

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

```
<class 'pandas.core.series.Series'>
RangeIndex: 3998 entries, 0 to 3997
Series name: Salary
Non-Null Count  Dtype
--------------  -----
3998 non-null   int64
dtypes: int64(1)
memory usage: 31.4 KB
```

In [ ]:
```
#convert salary in lakhs
df['Salary']=df['Salary'].apply(lambda x:x/100000)
df['Salary']
```

```
Out[ ]: 0        4.20
        1        5.00
        2        3.25
        3       11.00
        4        2.00
                ...
        3993     2.80
        3994     1.00
        3995     3.20
        3996     2.00
        3997     4.00
        Name: Salary, Length: 3998, dtype: float64
```

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

```
Out[ ]: count    3998.000000
        mean        3.076998
        std         2.127375
        min         0.350000
        25%         1.800000
        50%         3.000000
        75%         3.700000
        max        40.000000
        Name: Salary, dtype: float64
```

```
In [ ]: np.percentile(df['Salary'],99)
```

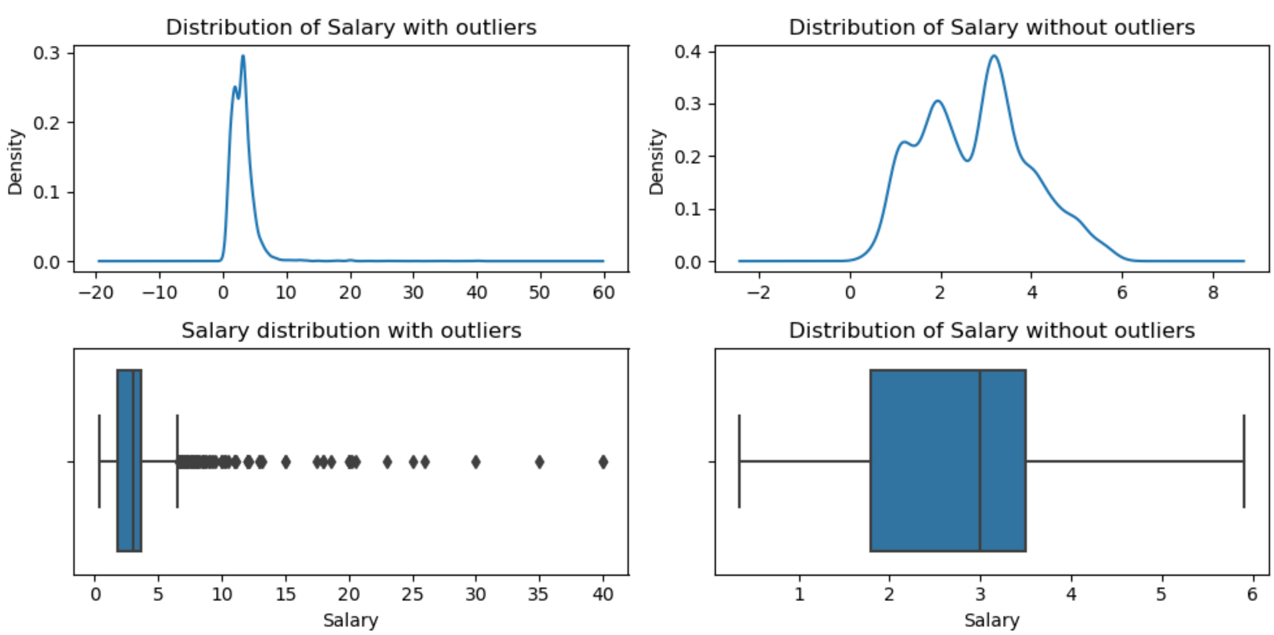
```
Out[ ]: 9.305999999999995
```

```
In [ ]: clean_df=pd.DataFrame(df['Salary'][df['Salary']<6])
```

```
In [ ]: num_salary_1To4=(clean_df['Salary'][(clean_df['Salary']>1)&(clean_df['Salary']<4)]).count()
        (num_salary_1To4/len(df))*100
```

```
Out[ ]: 72.28614307153578
```

```
In [ ]: plt.figure(figsize=(10,5))
        plt.subplot(221)
        df['Salary'].plot(kind='kde')
        plt.title('Distribution of Salary with outliers')
        plt.subplot(222)
        clean_df['Salary'].plot(kind='kde')
        plt.title('Distribution of Salary without outliers')
        plt.subplot(223)
        sns.boxplot(data=df,x='Salary')
        plt.title('Salary distribution with outliers')
        plt.subplot(224)
        sns.boxplot(data=clean_df,x='Salary')
        plt.title('Distribution of Salary without outliers')
        plt.tight_layout()
```



Observation

- Maximum Salary is 40 lakh and minimum salary is 0.35 lakh
- 72% of candidates earn between 1 and 4 Lakhs and 99 % of candidates earn less than 9.3 L
- Salary with outliers is right skewed but Salary without outliers is normally distributed but is bimodal.

DOJ

In [ ]:
```python
df['DOJ']
```

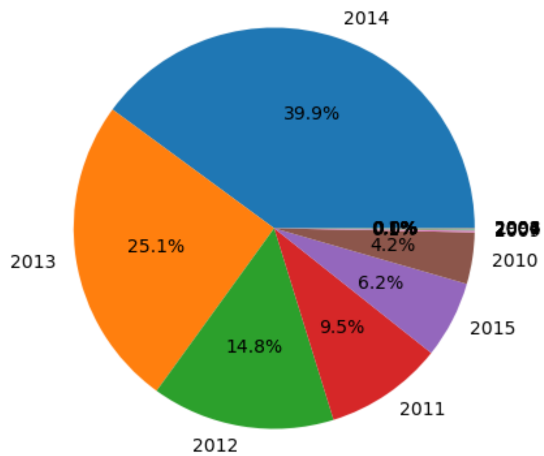
Out[ ]:
```
0      2012-06-01
1      2013-09-01
2      2014-06-01
3      2011-07-01
4      2014-03-01
          ...
3993   2011-10-01
3994   2013-07-01
3995   2013-07-01
3996   2014-07-01
3997   2013-02-01
Name: DOJ, Length: 3998, dtype: datetime64[ns]
```

In [ ]:
```python
years=df['DOJ'].apply(lambda x: x.year).value_counts()
clean_df['DOJ']=df['DOJ']
years
```

Out[ ]:
```
DOJ
2014    1596
2013    1004
2012     590
2011     381
2015     248
2010     166
2009       5
2007       4
2004       1
2008       1
2006       1
1991       1
Name: count, dtype: int64
```

In [ ]:
```python
plt.figure(figsize=(5,5))
plt.pie(years.to_list(),labels=years.index,autopct='%1.1f%%',)
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138bd30220>,
  <matplotlib.patches.Wedge at 0x2138bd30160>,
  <matplotlib.patches.Wedge at 0x2138bd30ee0>,
  <matplotlib.patches.Wedge at 0x2138bd5e5b0>,
  <matplotlib.patches.Wedge at 0x2138bd5ec40>,
  <matplotlib.patches.Wedge at 0x2138bd4b340>,
  <matplotlib.patches.Wedge at 0x2138bd4b9d0>,
  <matplotlib.patches.Wedge at 0x2138bd490a0>,
  <matplotlib.patches.Wedge at 0x2138bd49730>,
  <matplotlib.patches.Wedge at 0x2138bd49dc0>,
  <matplotlib.patches.Wedge at 0x2138bd03490>,
  <matplotlib.patches.Wedge at 0x2138bd03b20>],
 [Text(0.34254820698146204, 1.0453041308125524, '2014'),
  Text(-1.0867128648875153, -0.17045571063466722, '2013'),
  Text(-0.1781365704064419, -1.0854802449993417, '2012'),
  Text(0.6214194055216289, -0.9076551781602668, '2011'),
  Text(0.9776230027201093, -0.5042353265614355, '2015'),
  Text(1.0875044650991692, -0.16533008918333644, '2010'),
  Text(1.099850238191389, -0.018150855361238904, '2009'),
  Text(1.0999510978212879, -0.01037219368041271, '2007'),
  Text(1.099983359811554, -0.006050465906487639, '2004'),
  Text(1.099991510204029, -0.004321744909190228, '2008'),
  Text(1.0999969437604584, -0.00259301323774564, '2006'),
  Text(1.0999996604674225, -0.000864275161895414, '1991')],
 [Text(0.18684447653534292, 0.5701658895341194, '39.9%'),
  Text(-0.5927524717568264, -0.09297584216436393, '25.1%'),
  Text(-0.09716540203987739, -0.5920801336360045, '14.8%'),
  Text(0.3389560393754339, -0.49508464263287266, '9.5%'),
  Text(0.533248910574605, -0.27503745085169207, '6.2%'),
  Text(0.5931842536904559, -0.09018004864545624, '4.2%'),
  Text(0.5999183117407576, -0.009900466560675765, '0.1%'),
  Text(0.5999733260843387, -0.005657560189316023, '0.1%'),
  Text(0.5999909235335749, -0.0033002541308114392, '0.0%'),
  Text(0.5999953692021975, -0.0023573154050128514, '0.0%'),
  Text(0.59999833296025, -0.001414370856952167, '0.0%'),
  Text(0.5999998148004122, -0.00047142281557931663, '0.0%')])
```



Observation

- 39.9% joined in 2014 and only 1 person joined in following years(2004,2008,2006,1991)

DOL

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

```
<class 'pandas.core.series.Series'>
RangeIndex: 3998 entries, 0 to 3997
Series name: DOL
Non-Null Count  Dtype
--------------  -----
3998 non-null   object
dtypes: object(1)
memory usage: 31.4+ KB
```

In [ ]:
```
df['DOL'].unique()
```

Out[ ]:
```
array(['present', datetime.datetime(2015, 3, 1, 0, 0),
       datetime.datetime(2015, 5, 1, 0, 0),
       datetime.datetime(2015, 7, 1, 0, 0),
       datetime.datetime(2015, 4, 1, 0, 0),
       datetime.datetime(2014, 10, 1, 0, 0),
       datetime.datetime(2014, 9, 1, 0, 0),
       datetime.datetime(2014, 6, 1, 0, 0),
       datetime.datetime(2012, 9, 1, 0, 0),
       datetime.datetime(2013, 12, 1, 0, 0),
       datetime.datetime(2015, 6, 1, 0, 0),
       datetime.datetime(2013, 10, 1, 0, 0),
       datetime.datetime(2015, 1, 1, 0, 0),
       datetime.datetime(2014, 4, 1, 0, 0),
       datetime.datetime(2013, 6, 1, 0, 0),
       datetime.datetime(2012, 3, 1, 0, 0),
       datetime.datetime(2014, 7, 1, 0, 0),
       datetime.datetime(2013, 2, 1, 0, 0),
       datetime.datetime(2014, 1, 1, 0, 0),
       datetime.datetime(2013, 4, 1, 0, 0),
       datetime.datetime(2012, 7, 1, 0, 0),
       datetime.datetime(2014, 5, 1, 0, 0),
       datetime.datetime(2013, 9, 1, 0, 0),
       datetime.datetime(2015, 2, 1, 0, 0),
       datetime.datetime(2012, 1, 1, 0, 0),
       datetime.datetime(2015, 8, 1, 0, 0),
       datetime.datetime(2014, 8, 1, 0, 0),
       datetime.datetime(2015, 12, 1, 0, 0),
       datetime.datetime(2014, 12, 1, 0, 0),
       datetime.datetime(2012, 5, 1, 0, 0),
       datetime.datetime(2011, 3, 1, 0, 0),
       datetime.datetime(2011, 7, 1, 0, 0),
       datetime.datetime(2014, 2, 1, 0, 0),
       datetime.datetime(2011, 12, 1, 0, 0),
       datetime.datetime(2015, 10, 1, 0, 0),
       datetime.datetime(2014, 11, 1, 0, 0),
       datetime.datetime(2014, 3, 1, 0, 0),
       datetime.datetime(2011, 11, 1, 0, 0),
       datetime.datetime(2013, 5, 1, 0, 0),
       datetime.datetime(2013, 7, 1, 0, 0),
       datetime.datetime(2013, 11, 1, 0, 0),
       datetime.datetime(2011, 1, 1, 0, 0),
       datetime.datetime(2011, 5, 1, 0, 0),
       datetime.datetime(2012, 2, 1, 0, 0),
       datetime.datetime(2012, 11, 1, 0, 0),
       datetime.datetime(2012, 6, 1, 0, 0),
       datetime.datetime(2013, 8, 1, 0, 0),
       datetime.datetime(2005, 3, 1, 0, 0),
       datetime.datetime(2013, 3, 1, 0, 0),
       datetime.datetime(2012, 10, 1, 0, 0),
       datetime.datetime(2011, 2, 1, 0, 0),
       datetime.datetime(2010, 2, 1, 0, 0),
       datetime.datetime(2013, 1, 1, 0, 0),
       datetime.datetime(2011, 6, 1, 0, 0),
       datetime.datetime(2015, 9, 1, 0, 0),
       datetime.datetime(2012, 4, 1, 0, 0),
       datetime.datetime(2012, 8, 1, 0, 0),
       datetime.datetime(2011, 4, 1, 0, 0),
       datetime.datetime(2011, 10, 1, 0, 0),
       datetime.datetime(2015, 11, 1, 0, 0),
       datetime.datetime(2012, 12, 1, 0, 0),
       datetime.datetime(2011, 9, 1, 0, 0),
       datetime.datetime(2010, 8, 1, 0, 0),
       datetime.datetime(2011, 8, 1, 0, 0),
       datetime.datetime(2009, 6, 1, 0, 0),
       datetime.datetime(2008, 3, 1, 0, 0),
       datetime.datetime(2010, 10, 1, 0, 0)], dtype=object)
```

In [ ]:
```
(df['DOL']=='present').sum()
```

Out[ ]:
```
1875
```

In [ ]:
```
def replacePresent(x):
    if x=='present':
        return pd.to_datetime('2050-12-31',format='%Y-%m-%d')
    return x

df['DOL']=df['DOL'].apply(replacePresent)
df['DOL'].value_counts()
```

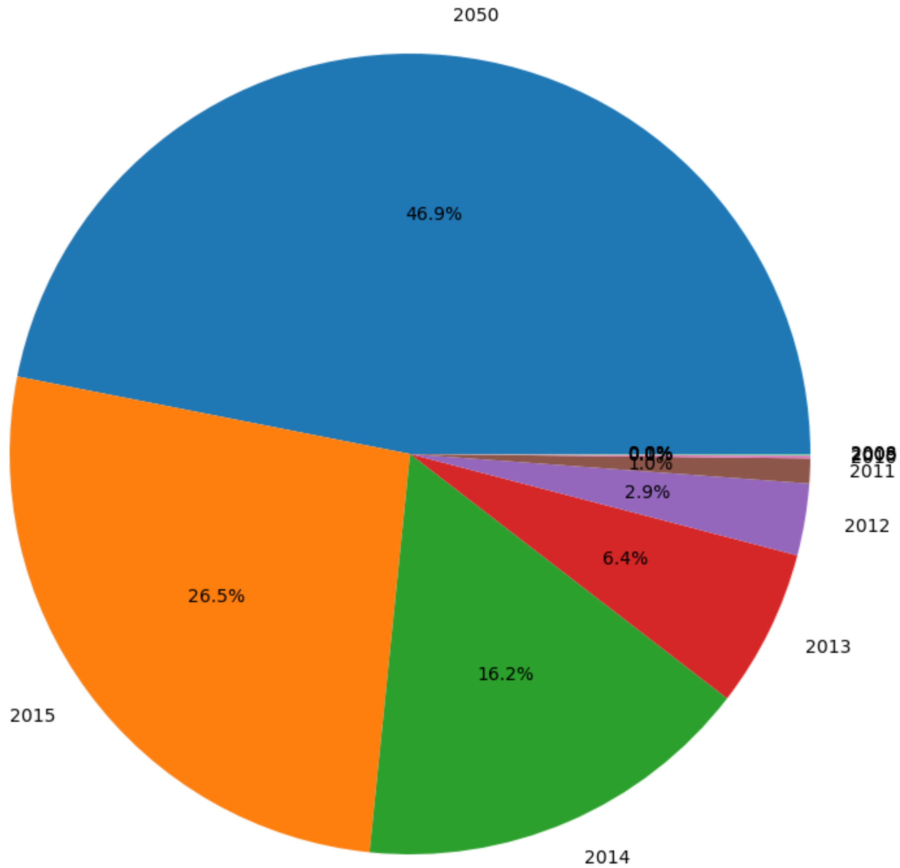
Out[ ]:
```
DOL
2050-12-31    1875
2015-04-01     573
2015-03-01     124
2015-05-01     112
2015-01-01      99
              ...
2005-03-01       1
2015-10-01       1
2010-02-01       1
2011-02-01       1
2010-10-01       1
Name: count, Length: 67, dtype: int64
```

In [ ]:
```
clean_df['DOL']=df['DOL']
df['DOL'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 3998 entries, 0 to 3997
Series name: DOL
Non-Null Count  Dtype
--------------  -----
3998 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 31.4 KB
```

In [ ]:
```
y=df['DOL'].apply(lambda x:x.year)
plt.figure(figsize=(10,10))
plt.pie(x=y.value_counts().to_list(),labels=y.value_counts().index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138bd94070>,
  <matplotlib.patches.Wedge at 0x2138bd9ff70>,
  <matplotlib.patches.Wedge at 0x2138bd94ca0>,
  <matplotlib.patches.Wedge at 0x2138babc370>,
  <matplotlib.patches.Wedge at 0x2138babca00>,
  <matplotlib.patches.Wedge at 0x2138bab4100>,
  <matplotlib.patches.Wedge at 0x2138bab4790>,
  <matplotlib.patches.Wedge at 0x2138bab4e50>,
  <matplotlib.patches.Wedge at 0x2138bad3520>,
  <matplotlib.patches.Wedge at 0x2138bad3bb0>],
 [Text(0.10701240624020007, 1.0947823276390072, '2050'),
  Text(-0.8840911579485944, -0.6545096060693943, '2015'),
  Text(0.4346880468048309, -1.0104683577257634, '2014'),
  Text(0.9873392480683092, -0.48493423185407947, '2013'),
  Text(1.0850572648795893, -0.1806951353302708, '2012'),
  Text(1.099009850785698, -0.04666206034882836, '2011'),
  Text(1.0999660388607693, -0.008643688619392145, '2010'),
  Text(1.0999915095338455, -0.004321915484121818, '2005'),
  Text(1.0999969433583485, -0.0025931838135198285, '2009'),
  Text(1.0999996603333866, -0.0008644457380909009, '2008')],
 [Text(0.05837040340374549, 0.5971539968940038, '46.9%'),
  Text(-0.48223154069923324, -0.3570052396742151, '26.5%'),
  Text(0.2371025709844532, -0.5511645587595072, '16.2%'),
  Text(0.5385486807645322, -0.26450958101131605, '6.4%'),
  Text(0.5918494172070486, -0.09856098290742042, '2.9%'),
  Text(0.5994599186103807, -0.02545203291754274, '1.0%'),
  Text(0.5999814757422377, -0.00471473924694117, '0.1%'),
  Text(0.599995368836643, -0.0023574084458846275, '0.0%'),
  Text(0.5999983327409173, -0.0014144638982835427, '0.0%'),
  Text(0.5999998147273017, -0.0004715158571404914, '0.0%')])
```



Observation

- 46.9% of candidates assumed to be still working in their companies
- 26.5% candidates left company in 2015

Designation

In [ ]:
```
df['Designation'].value_counts()
```

Out[ ]:
```
Designation
software engineer                  539
software developer                 265
system engineer                    205
programmer analyst                 139
systems engineer                   118
                                  ...
cad drafter                          1
noc engineer                         1
human resources intern               1
senior quality assurance engineer    1
jr. software developer               1
Name: count, Length: 419, dtype: int64
```

In [ ]:
```
df['Designation'].value_counts()
```

Out[ ]:

```
Designation
software engineer                  539
software developer                 265
system engineer                    205
programmer analyst                 139
systems engineer                   118
                                  ...
cad drafter                          1
noc engineer                         1
human resources intern               1
senior quality assurance engineer    1
jr. software developer               1
Name: count, Length: 419, dtype: int64
```


In [ ]:
```
clean_df['Designation']=df['Designation']
df['Designation'].value_counts()[:15]
```

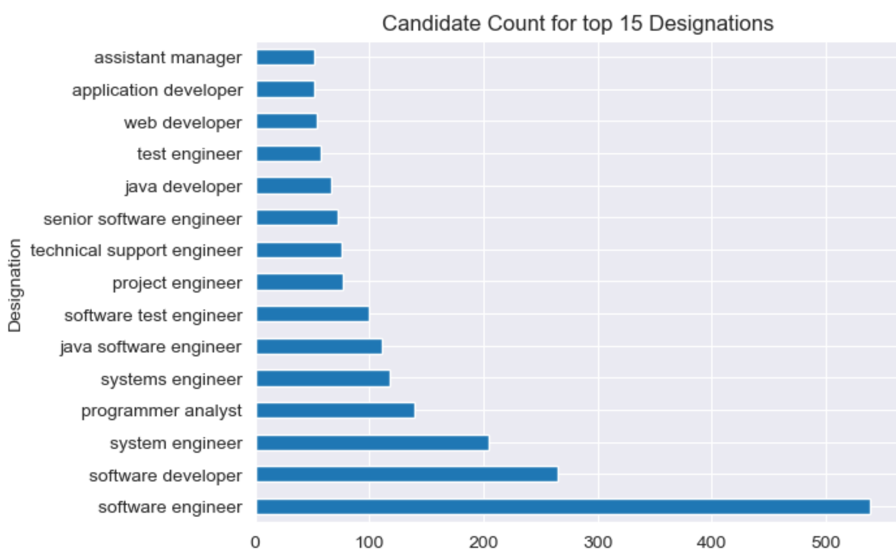
Out[ ]:
```
Designation
software engineer            539
software developer           265
system engineer              205
programmer analyst           139
systems engineer             118
java software engineer       111
software test engineer       100
project engineer              77
technical support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

In [ ]:
```
# Percentage of candidates who are working in these top 15 job roles
(df['Designation'].value_counts()[:15].sum()/len(df))*100
```

Out[ ]:
```
49.6248124062031
```

In [ ]:
```
sns.set_style('darkgrid')
plt.title('Candidate Count for top 15 Designations')
df['Designation'].value_counts()[:15].plot(kind='barh')
```

Out[ ]:
```
<Axes: title={'center': 'Candidate Count for top 15 Designations'}, ylabel='Designation'>
```



Observation

- There are 390+ unique job roles
- 49.62 % of candidates are working in these 15 designations most notably as software engineer.

Job City

In [ ]:
```
df['JobCity']
```

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

In [ ]:
```
df['JobCity'].value_counts()
```

Out[ ]:
```
JobCity
Bangalore           627
-1                  461
Noida               368
Hyderabad           335
Pune                290
                   ...
Tirunelvelli          1
Ernakulam             1
Nanded                1
Dharmapuri            1
Asifabadbanglore      1
Name: count, Length: 339, dtype: int64
```

In [ ]:
```
df['JobCity']=df['JobCity'].str.lower().str.strip()
clean_df['JobCity']=df['JobCity']
df['JobCity'].unique()
```

Out[ ]:
```
array(['bangalore', 'indore', 'chennai', 'gurgaon', 'manesar',
       'hyderabad', 'banglore', 'noida', 'kolkata', 'pune', nan, 'mohali',
       'jhansi', 'delhi', 'bhubaneswar', 'navi mumbai', 'mumbai',
       'new delhi', 'mangalore', 'rewari', 'gaziabaad', 'bhiwadi',
       'mysore', 'rajkot', 'greater noida', 'jaipur', 'thane',
       'maharajganj', 'thiruvananthapuram', 'punchkula', 'bhubaneshwar',
       'coimbatore', 'dhanbad', 'lucknow', 'trivandrum', 'gandhi nagar',
       'una', 'daman and diu', 'gurgoan', 'vsakhapttnam', 'nagpur',
       'bhagalpur', 'new delhi - jaisalmer', 'ahmedabad', 'kochi/cochin',
       'bankura', 'bengaluru', 'kanpur', 'vijayawada', 'kochi', 'beawar',
       'alwar', 'siliguri', 'raipur', 'bhopal', 'faridabad', 'jodhpur',
       'udaipur', 'muzaffarpur', 'kolkata`', 'bulandshahar', 'haridwar',
       'raigarh', 'visakhapatnam', 'jabalpur', 'unnao', 'aurangabad',
       'belgaum', 'dehradun', 'rudrapur', 'jamshedpur', 'vizag', 'nouda',
       'dharamshala', 'banagalore', 'hissar', 'ranchi', 'madurai',
       'gurga', 'chandigarh', 'australia', 'cheyyar', 'sonepat',
       'ghaziabad', 'pantnagar', 'jagdalpur', 'angul', 'baroda',
       'ariyalur', 'jowai', 'kochi/cochin, chennai and coimbatore',
       'neemrana', 'tirupathi', 'bhubneshwar', 'calicut', 'gandhinagar',
       'dubai', 'ahmednagar', 'nashik', 'bellary', 'ludhiana',
       'muzaffarnagar', 'gagret', 'indirapuram, ghaziabad', 'gwalior',
       'chennai & mumbai', 'rajasthan', 'sonipat', 'bareli', 'hospete',
       'miryalaguda', 'dharuhera', 'meerut', 'ganjam', 'hubli', 'ncr',
       'agra', 'trichy', 'kudankulam ,tarapur', 'ongole', 'sambalpur',
       'pondicherry', 'bundi', 'sadulpur,rajgarh,distt-churu,rajasthan',
       'am', 'bikaner', 'vadodara', 'india', 'asansol', 'tirunelvelli',
       'ernakulam', 'bilaspur', 'chandrapur', 'nanded', 'dharmapuri',
       'vandavasi', 'rohtak', 'patna', 'salem', 'nasikcity',
       'technopark, trivandrum', 'bharuch', 'tornagallu', 'jaspur',
       'burdwan', 'shimla', 'gajiabaad', 'jammu', 'shahdol',
       'muvattupuzha', 'al jubail,saudi arabia', 'kalmar, sweden',
       'secunderabad', 'a-64,sec-64,noida', 'ratnagiri', 'jhajjar',
       'gulbarga', 'hyderabad(bhadurpally)', 'nalagarh',
       'jeddah saudi arabia', 'chennai, bangalore', 'jamnagar',
       'tirupati', 'gonda', 'orissa', 'kharagpur',
       'navi mumbai , hyderabad', 'joshimath', 'bathinda', 'johannesburg',
       'kala amb', 'karnal', 'london', 'kota', 'panchkula', 'baddi hp',
       'nagari', 'mettur, tamil nadu', 'durgapur', 'pondi', 'surat',
       'kurnool', 'kolhapur', 'bhilai', 'hderabad', 'bahadurgarh',
       'rayagada, odisha', 'kakinada', 'varanasi', 'punr', 'nellore',
       'sahibabad', 'howrah', 'trichur', 'ambala', 'khopoli', 'keral',
       'roorkee', 'allahabad', 'delhi/ncr', 'jalandhar', 'vapi', 'pilani',
       'muzzafarpur', 'ras al khaimah', 'bihar', 'singaruli', 'pondy',
       'phagwara', 'guragaon', 'baripada', 'yamuna nagar', 'shahibabad',
       'sampla', 'guwahati', 'rourkela', 'banaglore', 'vellore', 'dausa',
       'latur (maharashtra )', 'mainpuri', 'dammam', 'haldia',
       'rae bareli', 'patiala', 'gorakhpur', 'new dehli', 'ambala city',
       'karad', 'rajpura', 'haryana', 'asifabadbanglore'], dtype=object)
```

In [ ]:
```
df['JobCity'].isna().sum()
```

Out[ ]:
```
461
```

In [ ]:
```
df['JobCity'].value_counts()[:10]
```

Out[ ]:
```
JobCity
bangalore    665
noida        389
hyderabad    368
pune         327
chennai      313
gurgaon      217
new delhi    204
kolkata      119
mumbai       119
jaipur        53
Name: count, dtype: int64
```

In [ ]:
```
(df['JobCity'].value_counts()[:10].sum()/len(df))*100
```

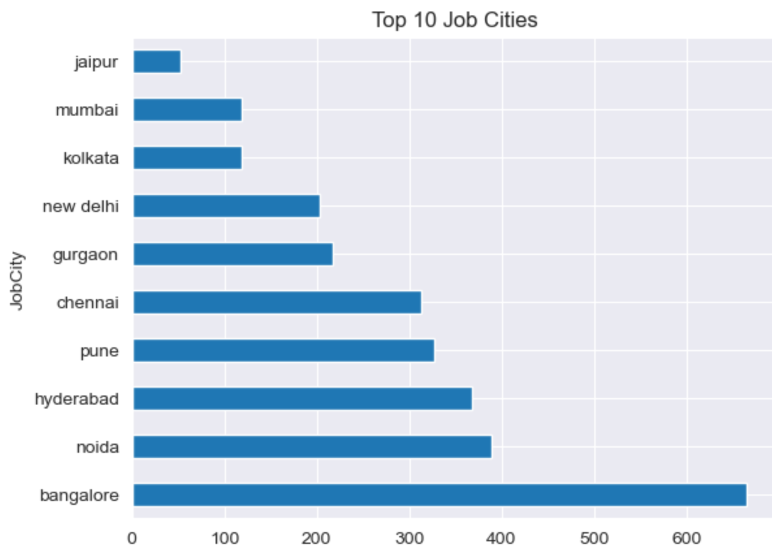
Out[ ]:
```
69.38469234617308
```

In [ ]:
```
sns.set_style('darkgrid')
plt.title('Top 10 Job Cities')
```

```
df['JobCity'].value_counts()[:10].plot(kind='barh')
```

Out[ ]:

```
<Axes: title={'center': 'Top 10 Job Cities'}, ylabel='JobCity'>
```



Observations

- 69.4% of candidates work in top 10 cities
- Banglore is the most popular career destination for candidates

Gender

In [ ]:

```
clean_df['Gender']=df['Gender']
df['Gender'].value_counts()
```

Out[ ]:

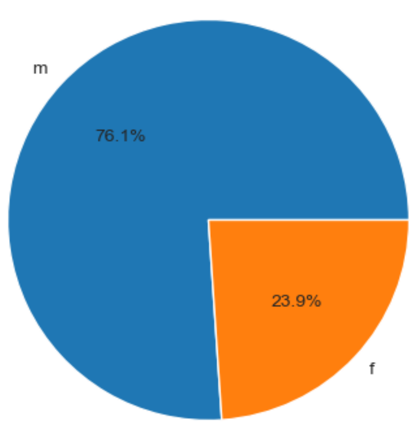
```
Gender
m    3041
f     957
Name: count, dtype: int64
```

In [ ]:

```
plt.figure(figsize=(5,5))
plt.pie(df['Gender'].value_counts().to_list(),labels=df['Gender'].value_counts().index,autopct='%1.1f%%',)
```

Out[ ]:

```
([<matplotlib.patches.Wedge at 0x2138c5a50a0>,
  <matplotlib.patches.Wedge at 0x2138bcd6fa0>],
 [Text(-0.803354993182058, 0.7514125065032227, 'm'),
  Text(0.803354993182058, -0.7514125065032227, 'f')],
 [Text(-0.4381936326447589, 0.40986136718357596, '76.1%'),
  Text(0.4381936326447589, -0.40986136718357596, '23.9%')])
```



Observation

- 76.1% of candidates are male
- 23.9% candidates are female

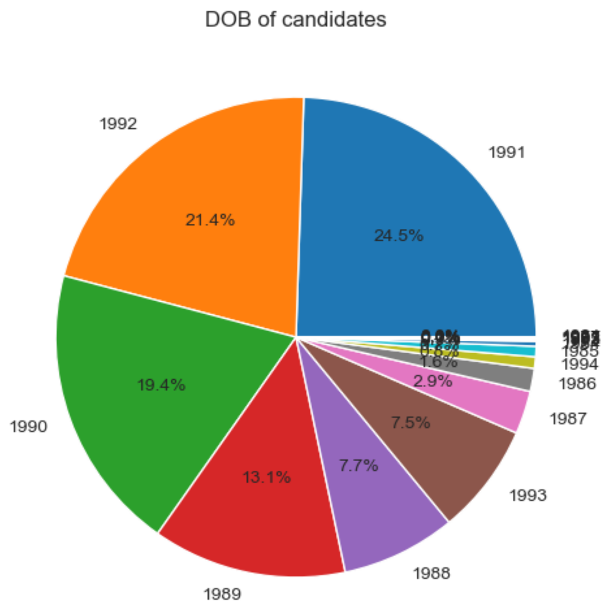
DOB

In [ ]:
```
clean_df['DOB']=df['DOB']
df['DOB']
```

Out[ ]:
```
0      1990-02-19
1      1989-10-04
2      1992-08-03
3      1989-12-05
4      1991-02-27
          ...
3993   1987-04-15
3994   1992-08-27
3995   1991-07-03
3996   1992-03-20
3997   1991-02-26
Name: DOB, Length: 3998, dtype: datetime64[ns]
```

In [ ]:
```
years=df['DOB'].apply(lambda x:x.year)
plt.figure(figsize=(6,6))
plt.title('DOB of candidates')
plt.pie(x=years.value_counts().to_list(),labels=years.value_counts().index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138c6081c0>,
  <matplotlib.patches.Wedge at 0x2138c5dc2e0>,
  <matplotlib.patches.Wedge at 0x2138c608df0>,
  <matplotlib.patches.Wedge at 0x2138c5fb4c0>,
  <matplotlib.patches.Wedge at 0x2138c5fbaf0>,
  <matplotlib.patches.Wedge at 0x2138c5ea1c0>,
  <matplotlib.patches.Wedge at 0x2138c5ea850>,
  <matplotlib.patches.Wedge at 0x2138c5eaee0>,
  <matplotlib.patches.Wedge at 0x2138c6235b0>,
  <matplotlib.patches.Wedge at 0x2138c623c40>,
  <matplotlib.patches.Wedge at 0x2138c631310>,
  <matplotlib.patches.Wedge at 0x2138c6319a0>,
  <matplotlib.patches.Wedge at 0x2138c63f070>,
  <matplotlib.patches.Wedge at 0x2138c63f700>,
  <matplotlib.patches.Wedge at 0x2138c63fdc0>,
  <matplotlib.patches.Wedge at 0x2138c64c490>,
  <matplotlib.patches.Wedge at 0x2138c64cb20>],
 [Text(0.7902456431886813, 0.765187443324384, '1991'),
  Text(-0.65659196678506, 0.8825457433772637, '1992'),
  Text(-1.0337275509450496, -0.3760416870735877, '1990'),
  Text(-0.2223295609579297, -1.0772973435056148, '1989'),
  Text(0.47716089756887287, -0.9911193055486649, '1988'),
  Text(0.8789208408970779, -0.6614364334059424, '1993'),
  Text(1.0444937204113747, -0.34501140274084485, '1987'),
  Text(1.082393040122605, -0.19602374012895737, '1986'),
  Text(1.093816465754284, -0.11647119491448411, '1994'),
  Text(1.0980390143502332, -0.06565304992739192, '1985'),
  Text(1.099584008556975, -0.030249101239122327, '1984'),
  Text(1.0998899696944653, -0.015558102889122453, '1983'),
  Text(1.0999510973205553, -0.010372246781978293, '1995'),
  Text(1.099978265474005, -0.00691487417091984, '1982'),
  Text(1.0999915099953896, -0.004321798012706778, '1977'),
  Text(1.0999969436352761, -0.0025930663415245046, '1997'),
  Text(1.0999996604256972, -0.0008643282658054333, '1981')],
 [Text(0.43104307810291703, 0.4173749690860276, '24.5%'),
  Text(-0.35814107279185087, 0.4813885872966892, '21.4%'),
  Text(-0.5638513914245724, -0.20511364749468416, '19.4%'),
  Text(-0.12127066961341618, -0.5876167328212444, '13.1%'),
  Text(0.26026958049211246, -0.5406105302992716, '7.7%'),
  Text(0.4794113677620424, -0.360783509130514, '7.5%'),
  Text(0.5697238474971134, -0.18818803785864263, '2.9%'),
  Text(0.590396203703239, -0.10692204007034037, '1.6%'),
  Text(0.5966271631387002, -0.0635297426806277, '0.8%'),
  Text(0.5989303714637635, -0.03581075450585013, '0.7%'),
  Text(0.5997730955765317, -0.016499509766793995, '0.3%'),
  Text(0.5999399834697082, -0.008486237939521336, '0.1%'),
  Text(0.599973325811212, -0.005657589153806341, '0.1%'),
  Text(0.5999881448040026, -0.003771749547774457З, '0.1%'),
  Text(0.5999953690883943, -0.0023573443705673333, '0.0%'),
  Text(0.5999983328919687, -0.0014143998226497298, '0.0%'),
  Text(0.599999814777653, -0.0004714517813484181, '0.0%')])
```



Observation

- 65.3 % of candidates were born between 1990 and 1991

10Percentage

In [ ]:
```python
df['10percentage']
```

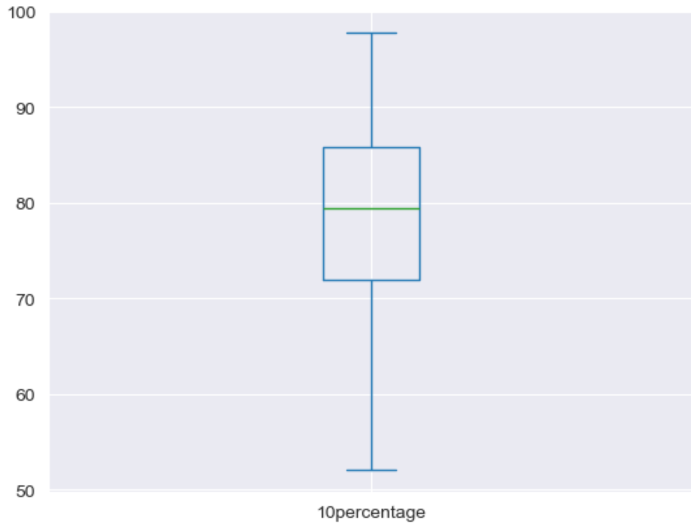
Out[ ]:
```
0       84.30
1       85.40
2       85.00
3       85.60
4       78.00
        ...
3993    52.09
3994    90.00
3995    81.86
3996    78.72
3997    70.60
Name: 10percentage, Length: 3998, dtype: float64
```

In [ ]:
```python
df['10percentage'].describe()
```

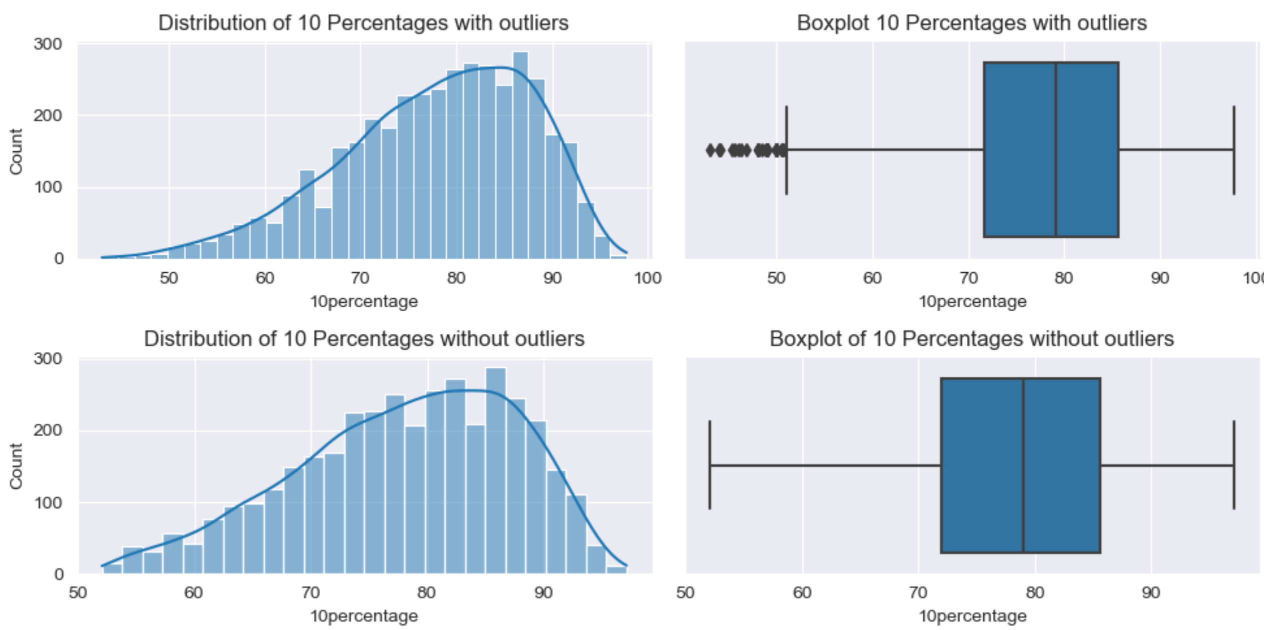
Out[ ]:
```
count    3998.000000
mean       77.925443
std         9.850162
min        43.000000
25%        71.680000
50%        79.150000
75%        85.670000
max        97.760000
Name: 10percentage, dtype: float64
```

In [ ]:
```python
clean_df['10percentage']=df['10percentage'][df['10percentage']>52]
df['10percentage'][df['10percentage']>52].plot(kind='box')
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
plt.figure(figsize=(10,5))
plt.subplot(221)
plt.title('Distribution of 10 Percentages with outliers')
sns.histplot(x=df['10percentage'],kde=True)
plt.subplot(222)
plt.title('Boxplot 10 Percentages with outliers')
sns.boxplot(x=df['10percentage'])
plt.subplot(223)
plt.title('Distribution of 10 Percentages without outliers')
sns.histplot(x=clean_df['10percentage'],kde=True)
plt.subplot(224)
plt.title('Boxplot of 10 Percentages without outliers')
sns.boxplot(x=clean_df['10percentage'])
plt.tight_layout()
```



Observations

- Minimum percentage is 43% and max is 97.7%
- % are left skewed, means most candidates scored more than 60

10board

In [ ]:
```
df['10board'].unique()[:10]
```

Out[ ]:
```
array(['board ofsecondary education,ap', 'cbse', 'state board',
       'mp board bhopal', 'icse',
       'karnataka secondary school of examination', 'up',
       'karnataka state education examination board', 'ssc',
       'kerala state technical education'], dtype=object)
```

In [ ]:
```
def filterBoards(x):
    try:
```

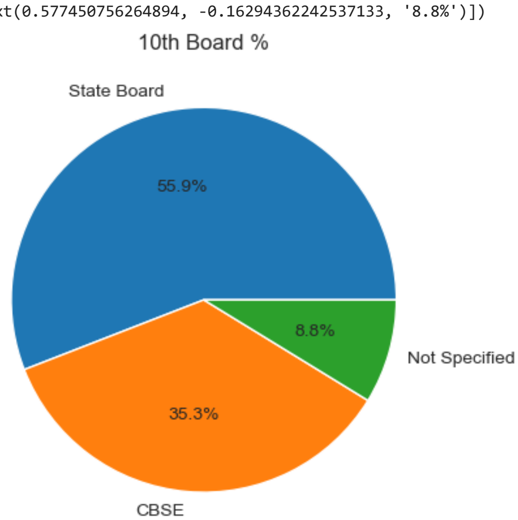
```
        if x==0:
            return 'Not Specified'
        x=x.lower().strip()
        if 'cbse' in x or 'central' in x:
            return 'CBSE'
        return 'State Board'
    except Exception as e:
        print(x)
```

In [ ]:
```
df['10board']=df['10board'].apply(filterBoards)
clean_df['10board']=df['10board']
df['10board'].value_counts()
```

Out[ ]:
```
10board
State Board      2235
CBSE             1413
Not Specified     350
Name: count, dtype: int64
```

In [ ]:
```
plt.title('10th Board %')
plt.pie(x=df['10board'].value_counts().to_list(),labels=df['10board'].value_counts().index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138ba71880>,
  <matplotlib.patches.Wedge at 0x2138baa4550>,
  <matplotlib.patches.Wedge at 0x2138ba60a60>],
 [Text(-0.20282415745552423, 1.0811393810015695, 'State Board'),
  Text(-0.09840639124763757, -1.0955894222570868, 'CBSE'),
  Text(1.0586597198189724, -0.29872997444651417, 'Not Specified')],
 [Text(-0.11063135861210412, 0.5897123896372197, '55.9%'),
  Text(-0.05367621340780231, -0.5975942303220473, '35.3%'),
  Text(0.577450756264894, -0.16294362242537133, '8.8%')])
```



Observations

- 35.3% of candidates belong to CBSE and 55.9% belong to State Board
- 8.8% candidates have not shared their 10th board

12percentage

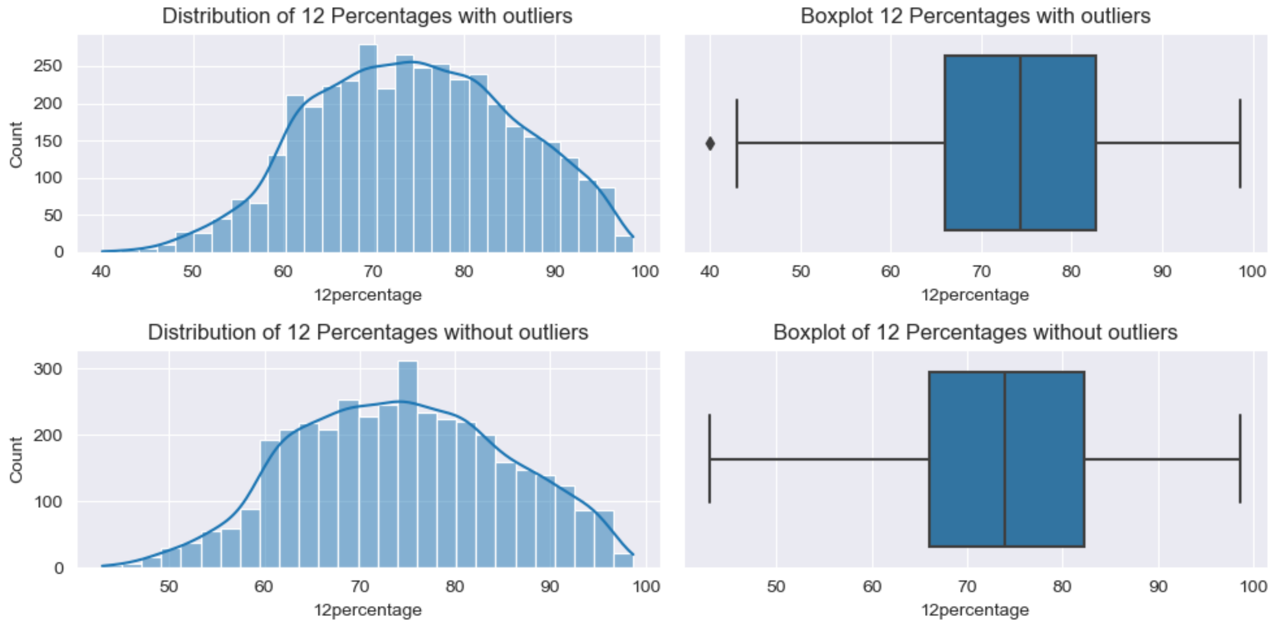
In [ ]:
```
df['12percentage'].describe()
```

Out[ ]:
```
count    3998.000000
mean       74.466366
std        10.999933
min        40.000000
25%        66.000000
50%        74.400000
75%        82.600000
max        98.700000
Name: 12percentage, dtype: float64
```

In [ ]:
```
clean_df['12percentage']=df['12percentage'][df['12percentage']>41]
```

In [ ]:
```
plt.figure(figsize=(10,5))
plt.subplot(221)
plt.title('Distribution of 12 Percentages with outliers')
sns.histplot(x=df['12percentage'],kde=True)
plt.subplot(222)
plt.title('Boxplot 12 Percentages with outliers')
sns.boxplot(x=df['12percentage'])
plt.subplot(223)
plt.title('Distribution of 12 Percentages without outliers')
sns.histplot(x=clean_df['12percentage'],kde=True)
plt.subplot(224)
```

```
plt.title('Boxplot of 12 Percentages without outliers')
sns.boxplot(x=clean_df['12percentage'])
plt.tight_layout()
```



Observation

- Minimum is 40% and max is 98.7%
- Distribution is little leftskewed which is good

In [ ]:

12board

In [ ]:
```
df['12board'].unique()[:10]
```

Out[ ]:
```
array(['board of intermediate education,ap', 'cbse', 'state board',
       'mp board', 'isc', 'icse', 'karnataka pre university board', 'up',
       'p u board, karnataka', 'dept of pre-university education'],
      dtype=object)
```

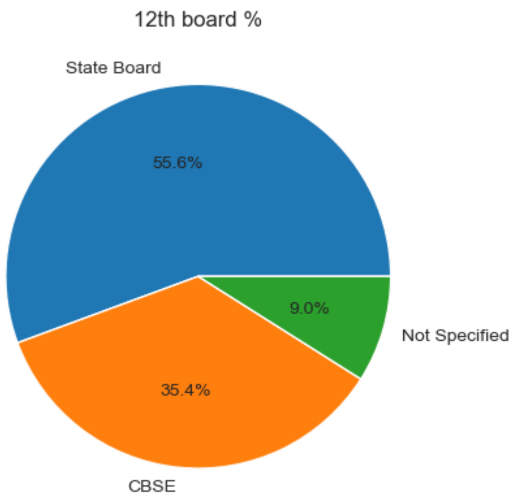
In [ ]:
```
def filterBoards(x):
    try:
        if x==0:
            return 'Not Specified'
        x=x.lower().strip()
        if 'cbse' in x or 'central' in x:
            return 'CBSE'
        return 'State Board'
    except Exception as e:
        print(x)
```

In [ ]:
```
df['12board']=df['12board'].apply(filterBoards)
clean_df['12board']=df['12board']
df['12board'].value_counts()
```

Out[ ]:
```
12board
State Board      2224
CBSE             1415
Not Specified     359
Name: count, dtype: int64
```

In [ ]:
```
plt.title('12th board %')
plt.pie(x=df['12board'].value_counts().to_list(),labels=df['12board'].value_counts().index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138cc5eee0>,
  <matplotlib.patches.Wedge at 0x2138c956e20>,
  <matplotlib.patches.Wedge at 0x2138cc6c910>],
 [Text(-0.19347153400918887, 1.0828521438904442, 'State Board'),
  Text(-0.11561191586601878, -1.0939076217440797, 'CBSE'),
  Text(1.0565205084462548, -0.3062097569191216, 'Not Specified')],
 [Text(-0.10552992764137574, 0.5906466239402423, '55.6%'),
  Text(-0.06306104501782842, -0.5966768845876799, '35.4%'),
  Text(0.5762839136979571, -0.1670235037740663, '9.0%')])
```



CollegeID

In [ ]:
```
df['CollegeID']
# No need
```

Out[ ]:
```
0        1141
1        5807
2          64
3        6920
4       11368
        ...
3993     6268
3994     4883
3995     9786
3996      979
3997     6609
Name: CollegeID, Length: 3998, dtype: int64
```

College Tier

In [ ]:
```
df['CollegeTier'].unique()
```

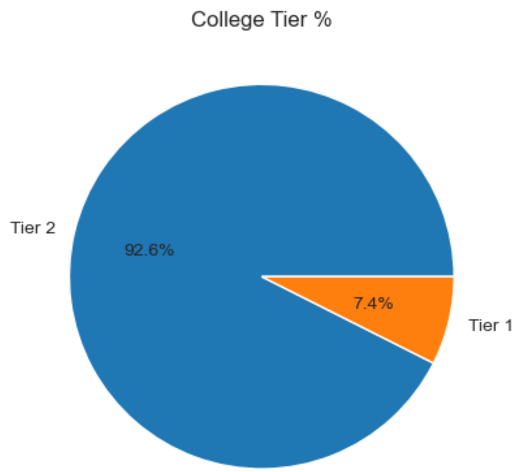
Out[ ]:
```
array([2, 1], dtype=int64)
```

In [ ]:
```
df['CollegeTier']=df['CollegeTier'].map({1:'Tier 1',2:'Tier 2'})
clean_df['CollegeTier']=df['CollegeTier']
clean_df['CollegeTier'].value_counts()
```

Out[ ]:
```
CollegeTier
Tier 2    3561
Tier 1     246
Name: count, dtype: int64
```

In [ ]:
```
plt.title('College Tier %')
plt.pie(x=df['CollegeTier'].value_counts().to_list(),labels=df['CollegeTier'].value_counts().index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138ccb6ca0>,
  <matplotlib.patches.Wedge at 0x2138cc9d640>],
 [Text(-1.070179333470586, 0.2543937778571886, 'Tier 2'),
  Text(1.0701793572886458, -0.25439367765976717, 'Tier 1')],
 [Text(-0.5837341818930468, 0.1387602424675574, '92.6%'),
  Text(0.5837341948847158, -0.13876018781441843, '7.4%')])
```



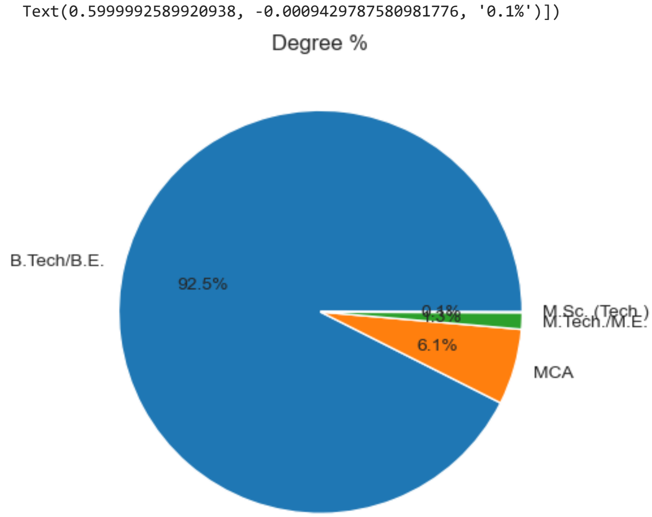
Degree

```
In [ ]: clean_df['Degree']=df['Degree']
        df['Degree'].value_counts()
```

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

```
In [ ]: plt.figure(figsize=(5,15))
        plt.title('Degree %')
        plt.pie(x=df['Degree'].value_counts().to_list(),labels=df['Degree'].value_counts().index,autopct='%1.1f%%')
```

```
Out[ ]: ([<matplotlib.patches.Wedge at 0x2138cd05c10>,
          <matplotlib.patches.Wedge at 0x2138ccd38b0>,
          <matplotlib.patches.Wedge at 0x2138cd16730>,
          <matplotlib.patches.Wedge at 0x2138cd16dc0>],
         [Text(-1.0699790741636042, 0.2552347563557841, 'B.Tech/B.E.'),
          Text(1.0579525160955623, -0.30122495528602755, 'MCA'),
          Text(1.0988967983270421, -0.0492526814150918, 'M.Tech./M.E.'),
          Text(1.0999986414855056, -0.0017287943898466592, 'M.Sc. (Tech.)')],
         [Text(-0.5836249495437841, 0.13921895801224585, '92.5%'),
          Text(0.5770650087793976, -0.1643045210651059, '6.1%'),
          Text(0.599398253632932, -0.026865098953686432, '1.3%'),
          Text(0.5999992589920938, -0.0009429787580981776, '0.1%')])
```



Specialization

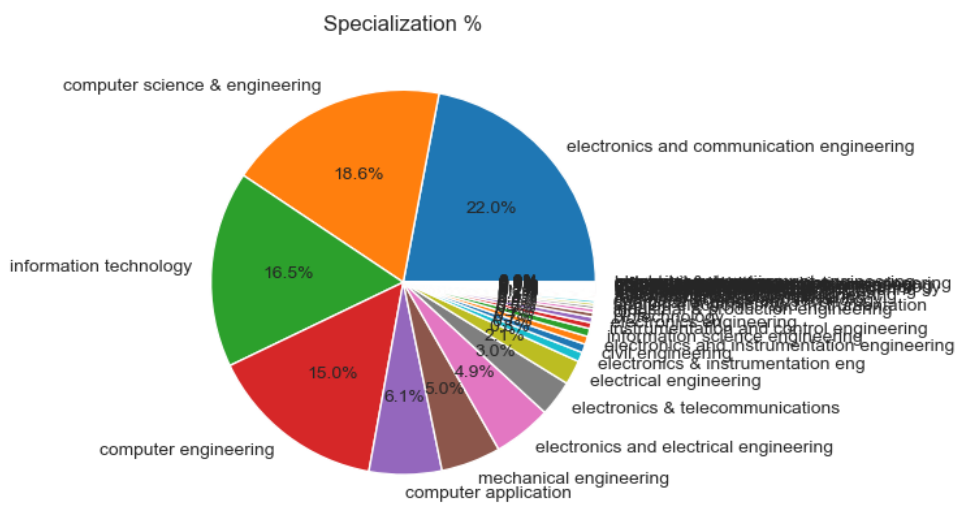
```
In [ ]: clean_df['Specialization']=df['Specialization']
        df['Specialization'].value_counts()
```

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

```
In [ ]: plt.title('Specialization %')
plt.pie(x=df['Specialization'].value_counts().to_list(),labels=df['Specialization'].value_counts().index,autopct='%1.1f%%')
```

```
Out[ ]: ([<matplotlib.patches.Wedge at 0x2138cd347c0>,
  <matplotlib.patches.Wedge at 0x2138ccc6d30>,
  <matplotlib.patches.Wedge at 0x2138d1e4970>,
  <matplotlib.patches.Wedge at 0x2138d1f2040>,
  <matplotlib.patches.Wedge at 0x2138d1f26d0>,
  <matplotlib.patches.Wedge at 0x2138d1f2d60>,
  <matplotlib.patches.Wedge at 0x2138d1fe430>,
  <matplotlib.patches.Wedge at 0x2138d1fea60>,
  <matplotlib.patches.Wedge at 0x2138d20f130>,
  <matplotlib.patches.Wedge at 0x2138d20f7c0>,
  <matplotlib.patches.Wedge at 0x2138d20fe50>,
  <matplotlib.patches.Wedge at 0x2138d21d520>,
  <matplotlib.patches.Wedge at 0x2138d21dbb0>,
  <matplotlib.patches.Wedge at 0x2138d22d0d0>,
  <matplotlib.patches.Wedge at 0x2138d22d790>,
  <matplotlib.patches.Wedge at 0x2138d22de20>,
  <matplotlib.patches.Wedge at 0x2138d2394f0>,
  <matplotlib.patches.Wedge at 0x2138d239b80>,
  <matplotlib.patches.Wedge at 0x2138d247250>,
  <matplotlib.patches.Wedge at 0x2138d2478e0>,
  <matplotlib.patches.Wedge at 0x2138d247f70>,
  <matplotlib.patches.Wedge at 0x2138d256640>,
  <matplotlib.patches.Wedge at 0x2138d256cd0>,
  <matplotlib.patches.Wedge at 0x2138d2653a0>,
  <matplotlib.patches.Wedge at 0x2138d265a30>,
  <matplotlib.patches.Wedge at 0x2138d275100>,
  <matplotlib.patches.Wedge at 0x2138d275790>,
  <matplotlib.patches.Wedge at 0x2138d275e20>,
  <matplotlib.patches.Wedge at 0x2138d2804f0>,
  <matplotlib.patches.Wedge at 0x2138d280b80>,
  <matplotlib.patches.Wedge at 0x2138d28d250>,
  <matplotlib.patches.Wedge at 0x2138d28d8e0>,
  <matplotlib.patches.Wedge at 0x2138d28df70>,
  <matplotlib.patches.Wedge at 0x2138d29d640>,
  <matplotlib.patches.Wedge at 0x2138d29dcd0>,
  <matplotlib.patches.Wedge at 0x2138d2ad3a0>,
  <matplotlib.patches.Wedge at 0x2138d2ada30>,
  <matplotlib.patches.Wedge at 0x2138d2bc100>,
  <matplotlib.patches.Wedge at 0x2138d2bc790>,
  <matplotlib.patches.Wedge at 0x2138d2bce20>,
  <matplotlib.patches.Wedge at 0x2138d2c84f0>,
  <matplotlib.patches.Wedge at 0x2138d2c8b80>,
  <matplotlib.patches.Wedge at 0x2138d2d6250>,
  <matplotlib.patches.Wedge at 0x2138d2d68e0>,
  <matplotlib.patches.Wedge at 0x2138d2d6f70>,
  <matplotlib.patches.Wedge at 0x2138d2e5640>],
 [Text(0.8473220811965028, 0.7014594006190432, 'electronics and communication engineering'),
  Text(-0.42514072093150806, 1.0145222360332167, 'computer science & engineering'),
  Text(-1.0972503311015518, 0.07772844329803029, 'information technology'),
  Text(-0.6669485181724147, -0.8747454910473219, 'computer engineering'),
  Text(0.012965271201902929, -1.0999235890472852, 'computer application'),
  Text(0.3890083025176943, -1.0289181408510117, 'mechanical engineering'),
  Text(0.6860318483177507, -0.8598606300405497, 'electronics and electrical engineering'),
  Text(0.8768374701202781, -0.664195792664385, 'electronics & telecommunications'),
  Text(0.9712062741089824, -0.5164865662641657, 'electrical engineering'),
  Text(1.0135172092080929, -0.427531129438593, 'electronics & instrumentation eng'),
  Text(1.0328382174130462, -0.3784774982096574, 'civil engineering'),
  Text(1.048487658673893, -0.3326764638632232, 'electronics and instrumentation engineering'),
  Text(1.0616560111865974, -0.28790018046427795, 'information science engineering'),
  Text(1.0715624159201755, -0.2485034985414024, 'instrumentation and control engineering'),
  Text(1.0786736759580176, -0.21555301156610668, 'electronics engineering'),
  Text(1.0840469464529037, -0.186660702575918, 'biotechnology'),
  Text(1.0878911717721726, -0.1627660848585145, 'other'),
  Text(1.0906550450597163, -0.14307890370627138, 'industrial & production engineering'),
  Text(1.0926695836965203, -0.12678004915787516, 'applied electronics and instrumentation'),
  Text(1.0943534336403526, -0.11131290257454593, 'chemical engineering'),
  Text(1.0955894150590593, -0.0984064713855156, 'computer science and technology'),
  Text(1.0964686174527383, -0.08807139683961385, 'telecommunication engineering'),
  Text(1.0971889105825794, -0.0785906768937201, 'mechanical and automation'),
  Text(1.0977725890028707, -0.06996672662011966, 'automobile/automotive engineering'),
  Text(1.0982399455821716, -0.06220146242387487, 'instrumentation engineering'),
  Text(1.0986092618320205, -0.05529638158960603, 'mechatronics'),
  Text(1.0988968004525939, -0.049252633991010146, 'aeronautical engineering'),
  Text(1.0991167997016595, -0.04407108591335167, 'electronics and computer engineering'),
  Text(1.0992814692362416, -0.03975237595176454, 'electrical and power engineering'),
  Text(1.0994009871198078, -0.03629696295824744, 'biomedical engineering'),
  Text(1.0995096434999605, -0.03284119136983128, 'information & communication technology'),
  Text(1.0996074373032318, -0.029385095327719497, 'industrial engineering'),
  Text(1.0996943675634705, -0.025928708976323888, 'computer science'),
  Text(1.099770433421851, -0.022472066462924347, 'metallurgical engineering'),
  Text(1.0998203525958128, -0.01987943701471724, 'power systems and automation'),
  Text(1.0998502365450853, -0.018150955118668877, 'control and instrumentation engineering'),
  Text(1.0998774040072403, -0.016422428392119787, 'mechanical & production engineering'),
  Text(1.0999018549151771, -0.01469386110430825, 'embedded systems technology'),
  Text(1.0999235892085057, -0.012965257524569803, 'polymer technology'),
  Text(1.0999426068335452, -0.011236621922329612, 'computer and communication engineering'),
  Text(1.099958907743324, -0.009507958567094866, 'information science'),
  Text(1.0999724918975815, -0.007779271728438373, 'internal combustion engine'),
  Text(1.0999833592627666, -0.00605056567599094, 'computer networking'),
  Text(1.099991509812038, -0.004321844679433762, 'ceramic engineering'),
  Text(1.0999969435252652, -0.00259311300848201, 'electronics'),
  Text(1.0999996603890274, -0.0008643749328772217, 'industrial & management engineering')],
```

```
[Text(0.4621756806526378, 0.3826142185194781, '22.0%'),
 Text(-0.23189493868991345, 0.5533757651090272, '18.6%'),
 Text(-0.5985001806008463, 0.042397332708016516, '16.5%'),
 Text(-0.36379010082131713, -0.47713390420763, '15.0%'),
 Text(0.0070719661101288685, -0.5999583212985191, '6.1%'),
 Text(0.21218634682783324, -0.5612280768278245, '5.0%'),
 Text(0.3741991899915003, -0.469014889113027, '4.9%'),
 Text(0.47827498370196986, -0.36228861418057356, '3.0%'),
 Text(0.5297488767867176, -0.2817199452349994, '2.1%'),
 Text(0.5528275686589597, -0.23319879787559616, '0.8%'),
 Text(0.563366300407116, -0.20644227175072222, '0.7%'),
 Text(0.5719023592766688, -0.18145988937993993, '0.7%'),
 Text(0.5790850970108712, -0.15703646207142433, '0.7%'),
 Text(0.5844885905019139, -0.13554736284076493, '0.5%'),
 Text(0.588367459613464, -0.11757436994514908, '0.5%'),
 Text(0.5912983344288565, -0.10181492867777343, '0.4%'),
 Text(0.5933951846030032, -0.08878150083191699, '0.3%'),
 Text(0.5949027518507543, -0.07804303838523892, '0.3%'),
 Text(0.5960015911071929, -0.06915275408611371, '0.2%'),
 Text(0.5969200547129195, -0.06071612867702505, '0.2%'),
 Text(0.5975942263958505, -0.05367625711937214, '0.2%'),
 Text(0.5980737913378572, -0.048038943730698454, '0.2%'),
 Text(0.5984666784995887, -0.04286764194202914, '0.1%'),
 Text(0.5987850485470203, -0.03816366906551981, '0.1%'),
 Text(0.5990399703175481, -0.033928070413022655, '0.1%'),
 Text(0.5992414155447383, -0.03016166268523965, '0.1%'),
 Text(0.5993982547923238, -0.026865073086005532, '0.1%'),
 Text(0.5995182543827233, -0.024038774134555452, '0.1%'),
 Text(0.599608074128859, -0.02168311415550793, '0.1%'),
 Text(0.5996732657017133, -0.01979834343177133, '0.1%'),
 Text(0.5997325328181602, -0.01791337711081706, '0.1%'),
 Text(0.5997858748926719, -0.016028233815119725, '0.1%'),
 Text(0.5998332913982566, -0.014142932168903936, '0.1%'),
 Text(0.5998747818664641, -0.012257490797958735, '0.1%'),
 Text(0.5999020105068069, -0.010843329280754857, '0.0%'),
 Text(0.5999183108427738, -0.009900520973819387, '0.0%'),
 Text(0.5999331294584946, -0.008957688213883519, '0.0%'),
 Text(0.5999464663173693, -0.00801483332962268, '0.0%'),
 Text(0.5999583213864577, -0.007071958649765346, '0.0%'),
 Text(0.5999686946364791, -0.006129066503088879, '0.0%'),
 Text(0.5999775860418131, -0.0051861592184153805, '0.0%'),
 Text(0.599984995580499, -0.0042432391246027485, '0.0%'),
 Text(0.5999909232342362, -0.0033003085505405125, '0.0%'),
 Text(0.5999953689883843, -0.002357369825145688, '0.0%'),
 Text(0.5999983328319628, -0.0014144252773538235, '0.0%'),
 Text(0.5999998147576513, -0.00047147723611484814, '0.0%')])
```



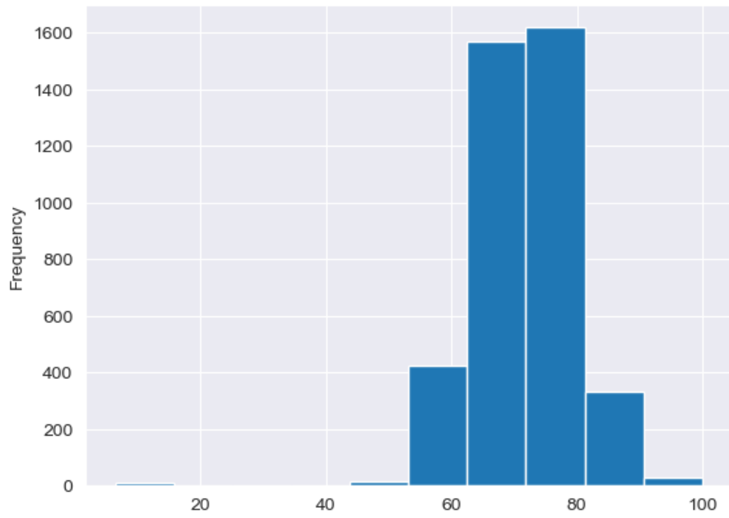
CollegeGPA

In [ ]:
```
df.rename(columns={'collegeGPA':'CollegeGPA'},inplace=True)
df['CollegeGPA']
```

Out[ ]:
```
0       78.00
1       70.06
2       70.00
3       74.64
4       73.90
        ...  
3993    61.50
3994    77.30
3995    70.00
3996    70.42
3997    68.00
Name: CollegeGPA, Length: 3998, dtype: float64
```

```
In [ ]: df['CollegeGPA'].plot(kind='hist')
        # Need to convert entries in 100 point scale to 10 point scale
```

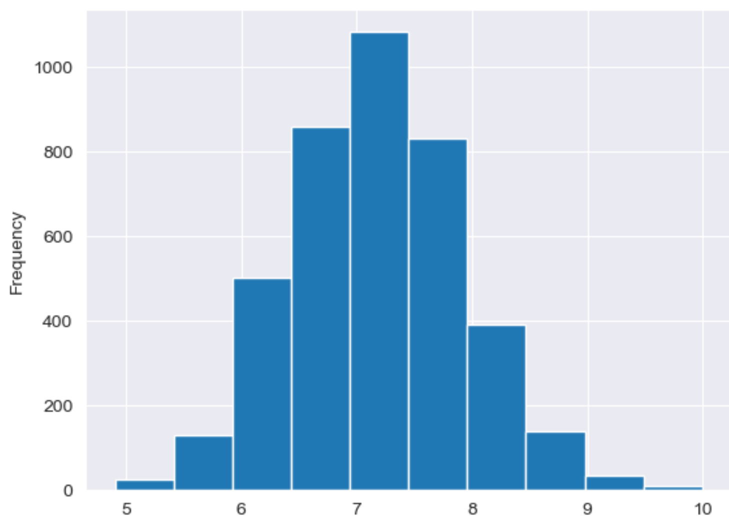
```
Out[ ]: <Axes: ylabel='Frequency'>
```




```
In [ ]: def toGPA(x):
            if x>11:
                return x/10
            return x
```

```
In [ ]: df['CollegeGPA'].apply(toGPA).plot(kind='hist')
```

```
Out[ ]: <Axes: ylabel='Frequency'>
```

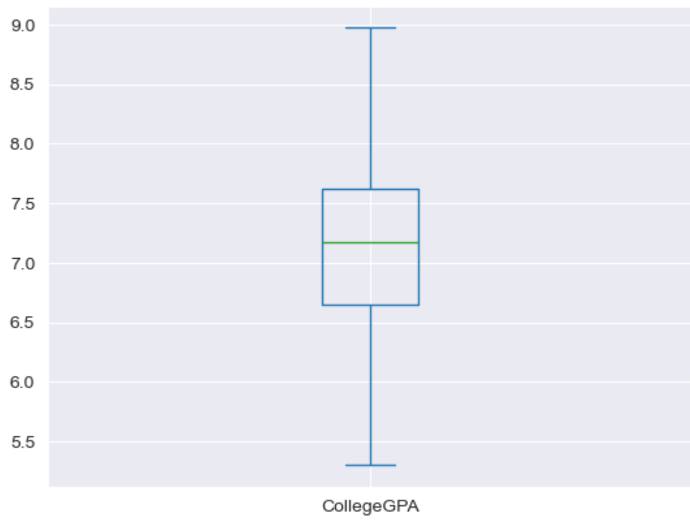



```
In [ ]: df['CollegeGPA'].apply(toGPA).describe()
```

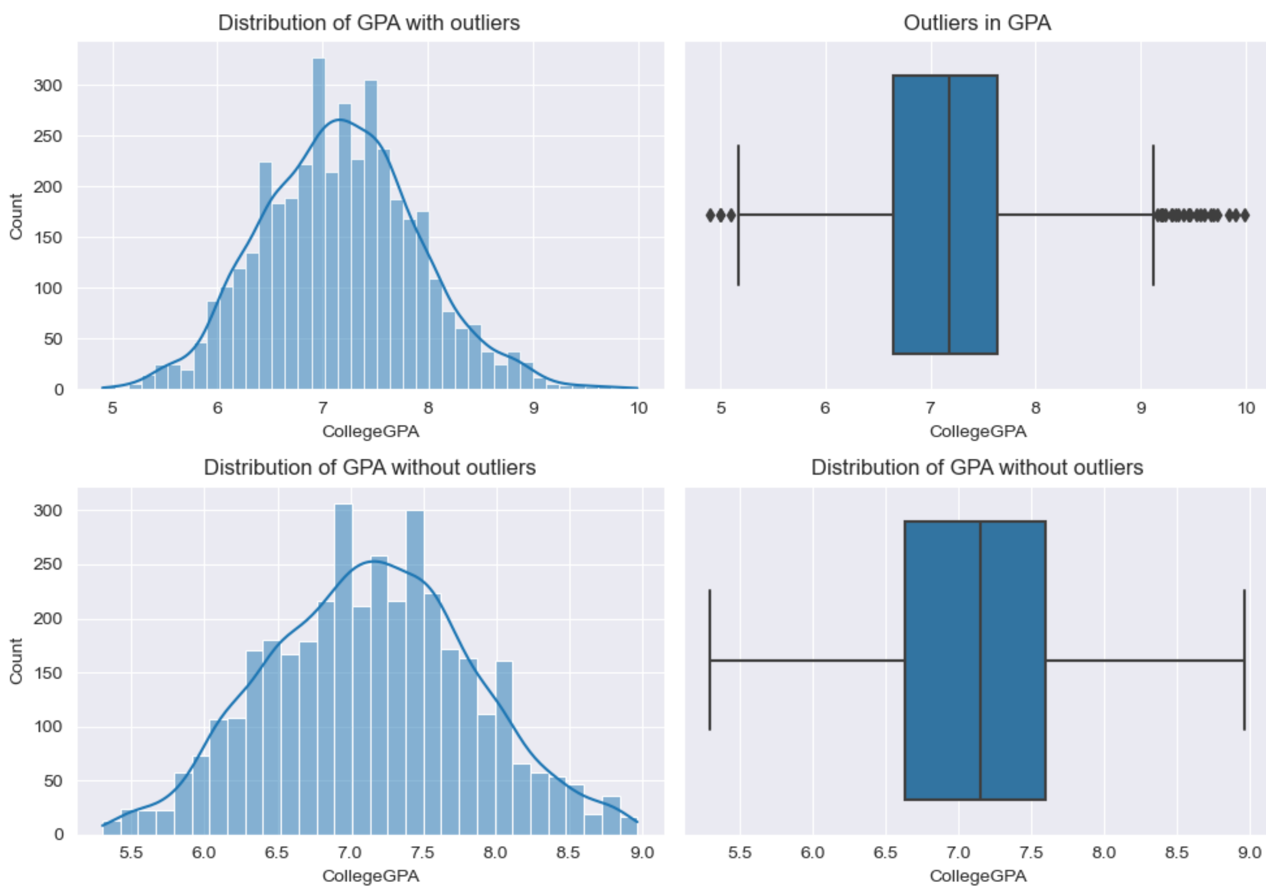
```
Out[ ]: count    3998.000000
        mean        7.169573
        std         0.740663
        min         4.907000
        25%         6.650500
        50%         7.180000
        75%         7.640000
        max         9.993000
        Name: CollegeGPA, dtype: float64
```

```
In [ ]: df['CollegeGPA']=df['CollegeGPA'].apply(toGPA)
        clean_df['CollegeGPA']=df['CollegeGPA'][(df['CollegeGPA']<9)&(df['CollegeGPA']>5.2)]
        df['CollegeGPA'][(df['CollegeGPA']<9)&(df['CollegeGPA']>5.2)].plot(kind='box')
```

```
Out[ ]: <Axes: >
```




```
In [ ]: plt.figure(figsize=(10,7))
plt.subplot(221)
plt.title('Distribution of GPA with outliers')
sns.histplot(x=df['CollegeGPA'],kde=True)
plt.subplot(222)
plt.title('Outliers in GPA')
sns.boxplot(x=df['CollegeGPA'])
plt.subplot(223)
plt.title('Distribution of GPA without outliers')
sns.histplot(x=clean_df['CollegeGPA'],kde=True)
plt.subplot(224)
plt.title('Distribution of GPA without outliers')
sns.boxplot(x=clean_df['CollegeGPA'])
plt.tight_layout()
```



Observation

- Median GPA is 7.16 and 75% candidates have less than 7.62 GPA
- minimum GPA is 4.9 and max GPA is 9.99

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

```
Out[ ]: count    3998.000000
mean        7.169573
std         0.740663
min         4.907000
25%         6.650500
50%         7.180000
75%         7.640000
max         9.993000
Name: CollegeGPA, dtype: float64
```

CollegeCityTier

```
In [ ]: df['CollegeCityTier'].value_counts()
```

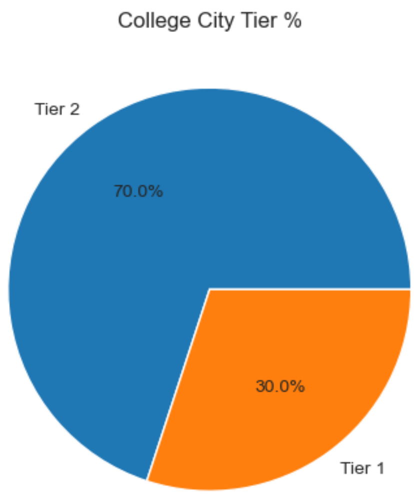
```
Out[ ]: CollegeCityTier
0    2797
1    1201
Name: count, dtype: int64
```

```
In [ ]: df['CollegeCityTier']=df['CollegeCityTier'].map({0:'Tier 2',1:'Tier 1'})
clean_df['CollegeCityTier']=df['CollegeCityTier']
df['CollegeCityTier'].value_counts()
```

```
Out[ ]: CollegeCityTier
Tier 2    2797
Tier 1    1201
Name: count, dtype: int64
```

```
In [ ]: plt.figure(figsize=(5,15))
plt.title('College City Tier %')
plt.pie(x=df['CollegeCityTier'].value_counts().to_list(),labels=df['CollegeCityTier'].value_counts().index,autopct='%1.1f%%')
```

```
Out[ ]: ([<matplotlib.patches.Wedge at 0x2138c8d6940>,
  <matplotlib.patches.Wedge at 0x2138e5d0190>],
 [Text(-0.645444409814041, 0.8907308874400867, 'Tier 2'),
  Text(0.6454444098140409, -0.8907308874400868, 'Tier 1')],
 [Text(-0.3520605871712951, 0.4858532113309563, '70.0%'),
  Text(0.352060587171295, -0.4858532113309564, '30.0%')])
```



CollegeState

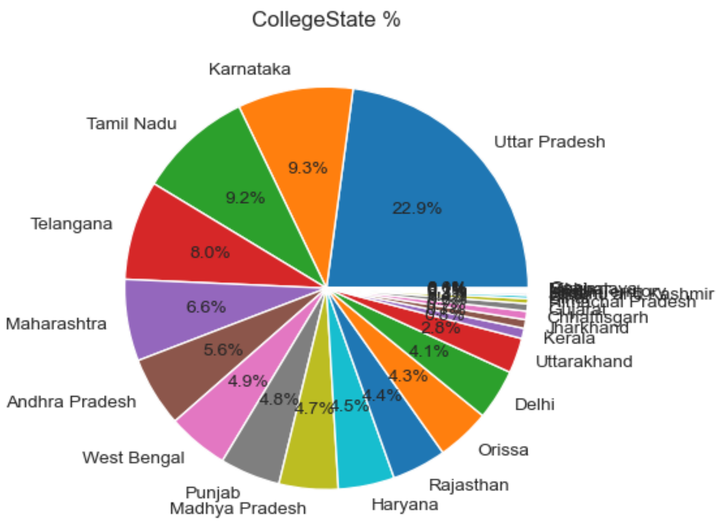
```
In [ ]: df['CollegeState'].unique()
```

```
Out[ ]: array(['Andhra Pradesh', 'Madhya Pradesh', 'Uttar Pradesh', 'Delhi',
       'Karnataka', 'Tamil Nadu', 'West Bengal', 'Maharashtra', 'Haryana',
       'Telangana', 'Orissa', 'Punjab', 'Kerala', 'Gujarat', 'Rajasthan',
       'Chhattisgarh', 'Uttarakhand', 'Jammu and Kashmir', 'Jharkhand',
       'Himachal Pradesh', 'Bihar', 'Assam', 'Goa', 'Sikkim',
       'Union Territory', 'Meghalaya'], dtype=object)
```

```
In [ ]: clean_df['CollegeState']=df[['CollegeState']]
plt.figure(figsize=(5,15))
plt.title('CollegeState %')
plt.pie(x=df['CollegeState'].value_counts().to_list(),labels=df['CollegeState'].value_counts().index,autopct='%1.1f%%')
```

Out[ ]:

```
([<matplotlib.patches.Wedge at 0x2138d368e50>,
  <matplotlib.patches.Wedge at 0x2138c90ffd0>,
  <matplotlib.patches.Wedge at 0x2138c8a9b80>,
  <matplotlib.patches.Wedge at 0x2138c8a9a90>,
  <matplotlib.patches.Wedge at 0x2138c8a93d0>,
  <matplotlib.patches.Wedge at 0x2138c908640>,
  <matplotlib.patches.Wedge at 0x2138c908430>,
  <matplotlib.patches.Wedge at 0x2138c9029d0>,
  <matplotlib.patches.Wedge at 0x2138c902160>,
  <matplotlib.patches.Wedge at 0x2138c902520>,
  <matplotlib.patches.Wedge at 0x2138be5bb20>,
  <matplotlib.patches.Wedge at 0x2138b982940>,
  <matplotlib.patches.Wedge at 0x2138b982be0>,
  <matplotlib.patches.Wedge at 0x2138b8f3df0>,
  <matplotlib.patches.Wedge at 0x2138b8f3820>,
  <matplotlib.patches.Wedge at 0x2138b93e520>,
  <matplotlib.patches.Wedge at 0x2138b93eaf0>,
  <matplotlib.patches.Wedge at 0x2138b89ca60>,
  <matplotlib.patches.Wedge at 0x2138ba27dc0>,
  <matplotlib.patches.Wedge at 0x2138ba27700>,
  <matplotlib.patches.Wedge at 0x2138ba27fd0>,
  <matplotlib.patches.Wedge at 0x2138c93e940>,
  <matplotlib.patches.Wedge at 0x2138ba8cb20>,
  <matplotlib.patches.Wedge at 0x2138ba8ce20>,
  <matplotlib.patches.Wedge at 0x2138ba603d0>,
  <matplotlib.patches.Wedge at 0x2138c91ca60>],
 [Text(0.8277120691686681, 0.7244948105766682, 'Uttar Pradesh'),
  Text(-0.17301695493867608, 1.0863080287394309, 'Karnataka'),
  Text(-0.7393350948646061, 0.8144836508497539, 'Tamil Nadu'),
  Text(-1.0525873161265074, 0.31946821740134984, 'Telangana'),
  Text(-1.0856198704833289, -0.17728366199895698, 'Maharashtra'),
  Text(-0.940894713268117, -0.5698395726378679, 'Andhra Pradesh'),
  Text(-0.7047830569079585, -0.8445595554461944, 'West Bengal'),
  Text(-0.41795530177903256, -1.0175034966597403, 'Punjab'),
  Text(-0.09840646818000008, -1.0955894153469805, 'Madhya Pradesh'),
  Text(0.2189423677656325, -1.0779908346536062, 'Haryana'),
  Text(0.5065386221005431, -0.9764315768759648, 'Rajasthan'),
  Text(0.7501490206843727, -0.8045349257591473, 'Orissa'),
  Text(0.9331988208752585, -0.5823572449253356, 'Delhi'),
  Text(1.0363611030711029, -0.3687216620179065, 'Uttarakhand'),
  Text(1.0717573592614522, -0.24766138752118552, 'Kerala'),
  Text(1.082393032379904, -0.19602378288217068, 'Jharkhand'),
  Text(1.0898515397792319, -0.1490758908772224, 'Chhattisgarh'),
  Text(1.0949491566850065, -0.10529171038022501, 'Gujarat'),
  Text(1.097717271273427, -0.07082931842126815, 'Himachal Pradesh'),
  Text(1.0989351637459182, -0.04838910913348122, 'Bihar'),
  Text(1.099483497916311, -0.033705159986463326, 'Jammu and Kashmir'),
  Text(1.0997524356573483, -0.023336243607967194, 'Assam'),
  Text(1.099901854990413, -0.014693855472575831, 'Union Territory'),
  Text(1.0999724919374132, -0.007779266096344275, 'Sikkim'),
  Text(1.0999945662918562, -0.0034574742791916714, 'Meghalaya'),
  Text(1.0999996603934532, -0.0008643693006449924, 'Goa')],
 [Text(0.45147931045563705, 0.3951789875872735, '22.9%'),
  Text(-0.09437288451200514, 0.5925316520396895, '9.3%'),
  Text(-0.403273688107967, 0.4442638095544112, '9.2%'),
  Text(-0.574138536069004, 0.17425539130982717, '8.0%'),
  Text(-0.5921562929909067, -0.09670017927215833, '6.6%'),
  Text(-0.5132152981462456, -0.31082158507520063, '5.6%'),
  Text(-0.3844271219497955, -0.4606688484251969, '4.9%'),
  Text(-0.22797561915219955, -0.5550019072689492, '4.8%'),
  Text(-0.053676255370909136, -0.5975942265528984, '4.7%'),
  Text(0.11942310969034499, -0.5879950007201488, '4.5%'),
  Text(0.2762937938730235, -0.5325990419323444, '4.4%'),
  Text(0.4091721931005669, -0.4388372322322622, '4.3%'),
  Text(0.5090175386592318, -0.3176494063229103, '4.1%'),
  Text(0.5652878744024197, -0.20112090655522172, '2.8%'),
  Text(0.5845949232335192, -0.13508802955701027, '0.8%'),
  Text(0.5903961994799476, -0.10692206339027491, '0.7%'),
  Text(0.5944644762432173, -0.08131412229666675, '0.7%'),
  Text(0.597244994555458, -0.05743184202557727, '0.6%'),
  Text(0.5987548752400511, -0.03863417368432808, '0.4%'),
  Text(0.5994191802250463, -0.026394059527353388, '0.3%'),
  Text(0.5997182715907151, -0.018384632719889084, '0.2%'),
  Text(0.5998649649040081, -0.012728860149800287, '0.1%'),
  Text(0.599946466358407, -0.008014830257768633, '0.1%'),
  Text(0.5999849956022253, -0.0042432360525514225, '0.1%'),
  Text(0.5999970361591943, -0.001885895061377275, '0.1%'),
  Text(0.5999998147600654, -0.00047147416398817763, '0.0%')])
```



GraduationYear

In [ ]:
```
df['GraduationYear'].value_counts()
```

Out[ ]:
```
GraduationYear
2013    1181
2014    1036
2012     847
2011     507
2010     292
2015      94
2009      24
2017       8
2016       7
0          1
2007       1
Name: count, dtype: int64
```

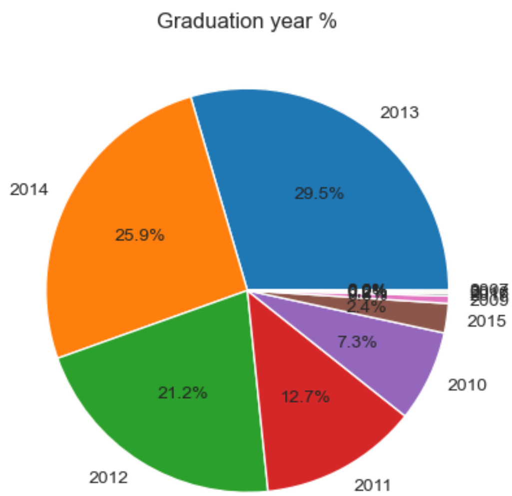
In [ ]:
```
clean_df['GraduationYear']=(df['GraduationYear'][df['GraduationYear']!=0]).astype(int)
clean_df['GraduationYear'].value_counts()
```

Out[ ]:
```
GraduationYear
2013.0    1146
2014.0    1015
2012.0     819
2011.0     461
2010.0     242
2015.0      89
2009.0      18
2017.0       8
2016.0       7
2007.0       1
Name: count, dtype: int64
```

In [ ]:
```
plt.figure(figsize=(5,15))
plt.title('Graduation year %')
plt.pie(x=df['GraduationYear'].value_counts().to_list(),labels=df['GraduationYear'].value_counts().index,autopct='%1.1f%%')
```

Out[ ]:

```
([<matplotlib.patches.Wedge at 0x2138b960eb0>,
  <matplotlib.patches.Wedge at 0x2138b875850>,
  <matplotlib.patches.Wedge at 0x2138b960280>,
  <matplotlib.patches.Wedge at 0x2138b9076a0>,
  <matplotlib.patches.Wedge at 0x2138b907d90>,
  <matplotlib.patches.Wedge at 0x2138ba7b790>,
  <matplotlib.patches.Wedge at 0x2138ba7b280>,
  <matplotlib.patches.Wedge at 0x2138ba7bf70>,
  <matplotlib.patches.Wedge at 0x2138c8f1970>,
  <matplotlib.patches.Wedge at 0x2138b9e2310>,
  <matplotlib.patches.Wedge at 0x2138b9e2d60>],
 [Text(0.6593626871041788, 0.8804776242782988, '2013'),
  Text(-0.9799894078438199, 0.49962061658214163, '2014'),
  Text(-0.5867505996035782, -0.9304427622722644, '2012'),
  Text(0.5286560396498096, -0.9646360929085015, '2011'),
  Text(0.99446905353715, -0.47013966175693517, '2010'),
  Text(1.0894970875382415, -0.15164463803804476, '2015'),
  Text(1.0988577602289418, -0.050116093070325315, '2009'),
  Text(1.099770434013597, -0.022472037503191367, '2017'),
  Text(1.099958907990215, -0.009507930004686489, '2016'),
  Text(1.0999969435916512, -0.0025930848473865093, '0'),
  Text(1.099999660411156, -0.0008643467717131442, '2007')],
 [Text(0.35965237478409745, 0.4802605223336175, '29.5%'),
  Text(-0.5345396770057199, 0.27252033631753175, '25.9%'),
  Text(-0.3200457816019517, -0.5075142339666896, '21.2%'),
  Text(0.288357839808987, -0.5261651415864553, '12.7%'),
  Text(0.5424376655657182, -0.2564398155037828, '7.3%'),
  Text(0.5942711386572225, -0.08271525711166078, '2.4%'),
  Text(0.5993769601248773, -0.027336050765631986, '0.6%'),
  Text(0.5998747821892346, -0.012257475001740745, '0.2%'),
  Text(0.5999775861764809, -0.005186143638919903, '0.2%'),
  Text(0.5999983328681733, -0.0014144099167562778, '0.0%'),
  Text(0.5999998147697213, -0.0004714618754798968, '0.0%')])
```



English

In [ ]:

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

Out[ ]:

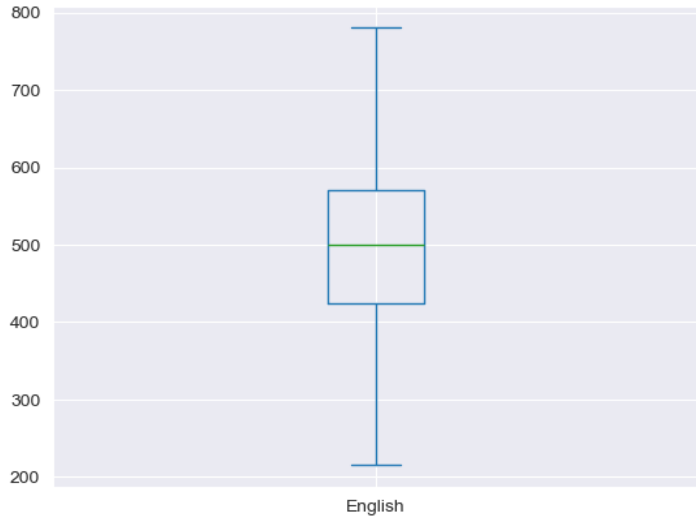
```
count    3998.000000
mean      501.649075
std       104.940021
min       180.000000
25%       425.000000
50%       500.000000
75%       570.000000
max       875.000000
Name: English, dtype: float64
```

In [ ]:

```
df['English'][(df['English']>210)&(df['English']<790)].plot(kind='box')
```

Out[ ]:

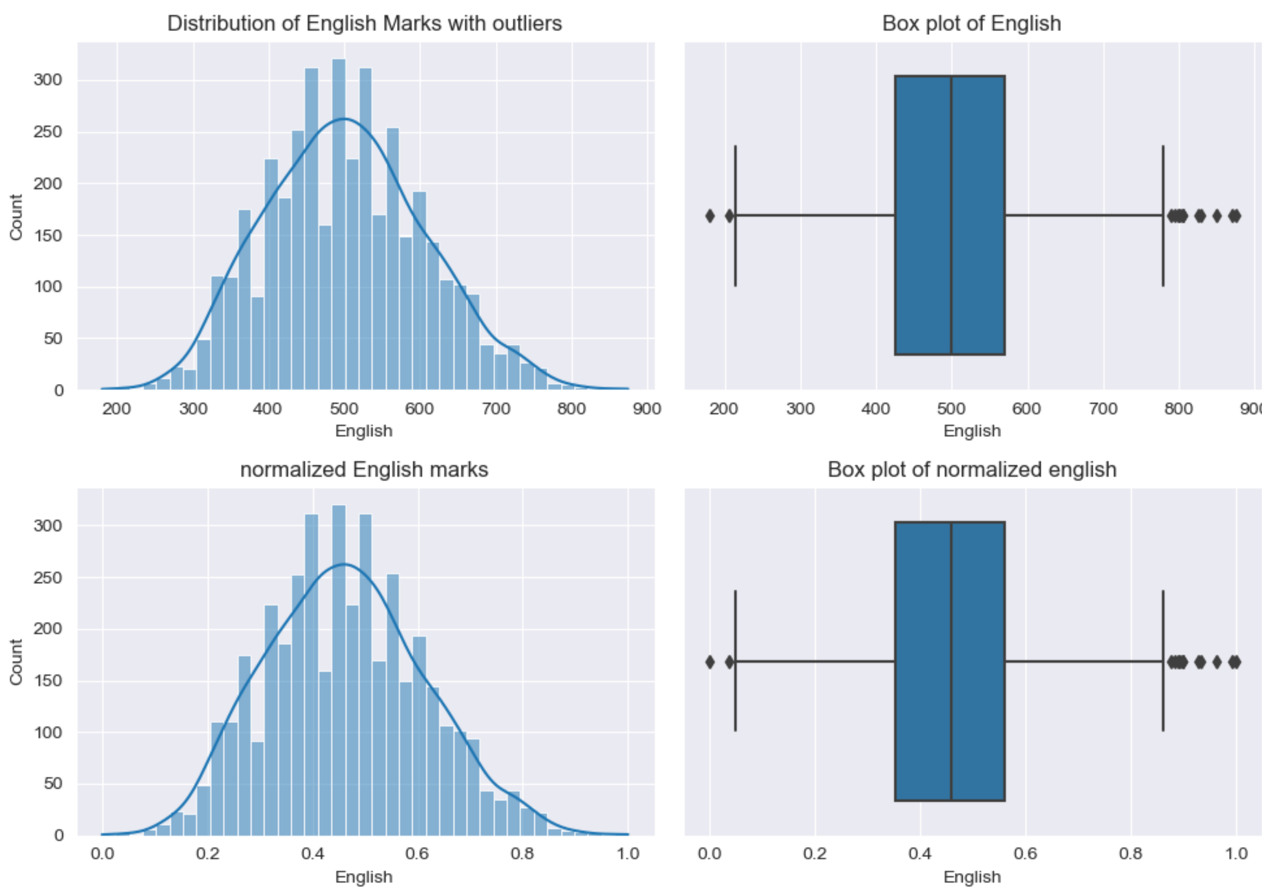
```
<Axes: >
```




```
In [ ]: scaler = MinMaxScaler()
        stand_marks = pd.DataFrame(index=df.index)  # Defining the index

        # Standarize the 'English' column and assign it to 'stand_marks'
        stand_marks['English'] = scaler.fit_transform(df[['English']])
```

```
In [ ]: plt.figure(figsize=(10,7))
        plt.subplot(221)
        plt.title('Distribution of English Marks with outliers')
        sns.histplot(x=df['English'],kde=True)
        plt.subplot(222)
        plt.title('Box plot of English')
        sns.boxplot(x=df['English'])
        plt.subplot(223)
        plt.title('normalized English marks')
        sns.histplot(x=stand_marks['English'],kde=True)
        plt.subplot(224)
        plt.title('Box plot of normalized english')
        sns.boxplot(x=stand_marks['English'])
        plt.tight_layout()
```



Observation

- Distribution of English marks resemble normal distribution
- 75% of students scored less than 570 marks
- max marks scored is 875 and minimum marks scored=180

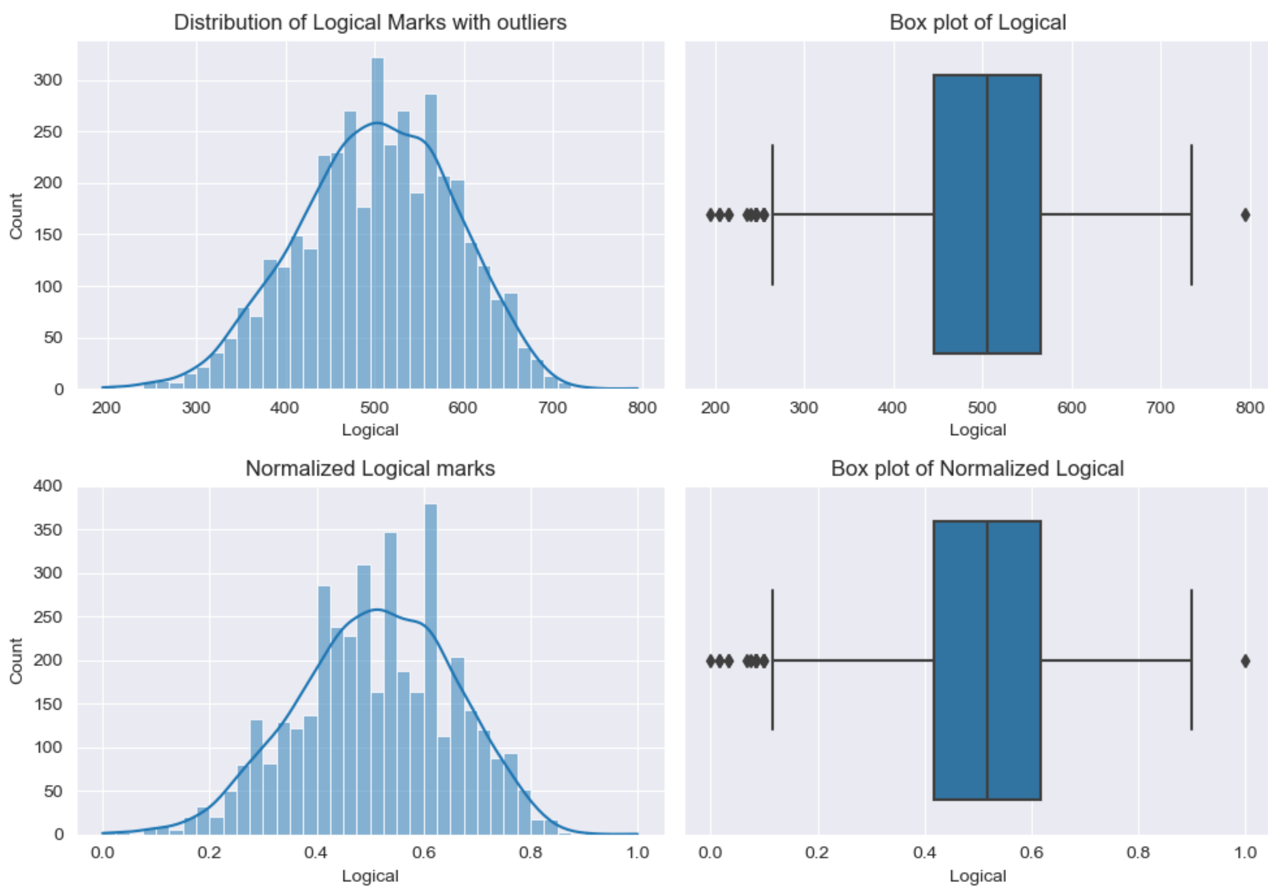
Logical

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

Out[ ]:
```
count    3998.000000
mean      501.598799
std        86.783297
min       195.000000
25%       445.000000
50%       505.000000
75%       565.000000
max       795.000000
Name: Logical, dtype: float64
```

In [ ]:
```
stand_marks['Logical']=scaler.fit_transform(df[['Logical']])
```

In [ ]:
```
plt.figure(figsize=(10,7))
plt.subplot(221)
plt.title('Distribution of Logical Marks with outliers')
sns.histplot(x=df['Logical'],kde=True)
plt.subplot(222)
plt.title('Box plot of Logical')
sns.boxplot(x=df['Logical'])
plt.subplot(223)
plt.title('Normalized Logical marks')
sns.histplot(x=stand_marks['Logical'],kde=True)
plt.subplot(224)
plt.title('Box plot of Normalized Logical')
sns.boxplot(x=stand_marks['Logical'])
plt.tight_layout()
```



Observation

- Logical Marks are left skwed and 75% candidates scored above 445

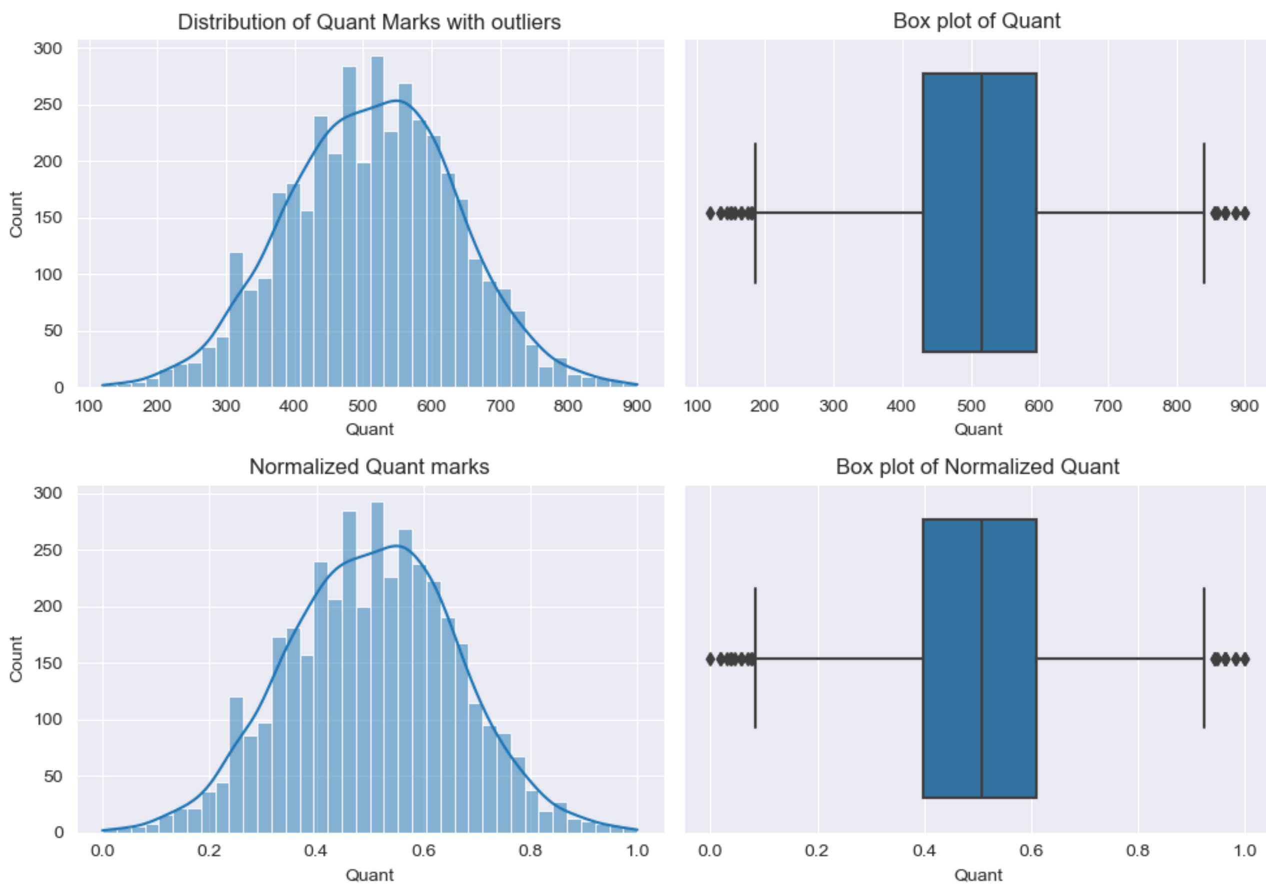
Quant

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

Out[ ]:
```
count    3998.000000
mean      513.378189
std       122.302332
min       120.000000
25%       430.000000
50%       515.000000
75%       595.000000
max       900.000000
Name: Quant, dtype: float64
```

In [ ]:
```
stand_marks['Quant']=scaler.fit_transform(df[['Quant']])
```

In [ ]:
```
plt.figure(figsize=(10,7))
plt.subplot(221)
plt.title('Distribution of Quant Marks with outliers')
sns.histplot(x=df['Quant'],kde=True)
plt.subplot(222)
plt.title('Box plot of Quant')
sns.boxplot(x=df['Quant'])
plt.subplot(223)
plt.title('Normalized Quant marks')
sns.histplot(x=stand_marks['Quant'],kde=True)
plt.subplot(224)
plt.title('Box plot of Normalized Quant')
sns.boxplot(x=stand_marks['Quant'])
plt.tight_layout()
```



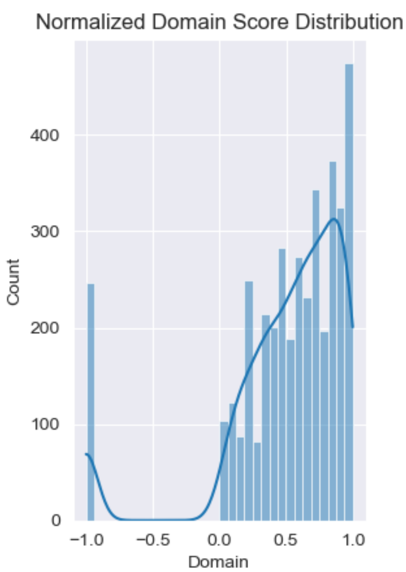
Observation

- Quant marks are normally distributed
- 75% candidates scored less than 595 and max score is 900

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

Out[ ]:
```
count    3998.000000
mean      513.378189
std       122.302332
min       120.000000
25%       430.000000
50%       515.000000
75%       595.000000
max       900.000000
Name: Quant, dtype: float64
```

Domain

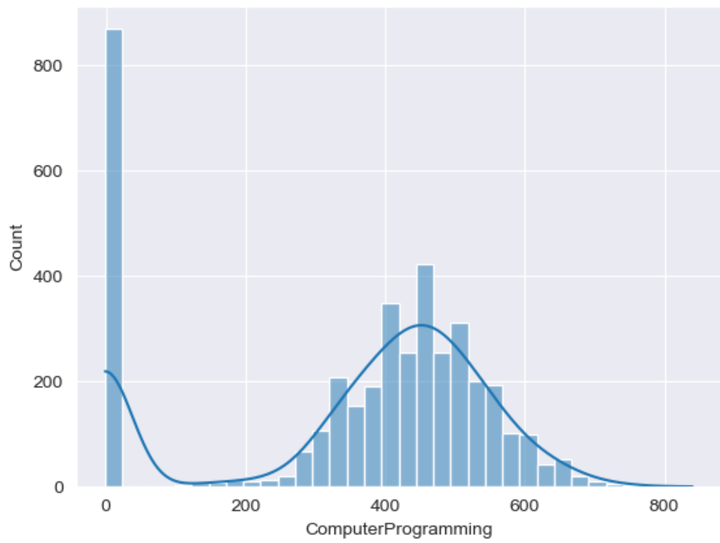
In [ ]:
```
df['Domain'].describe()
```
Out[ ]:
```
count    3998.000000
mean        0.510490
std         0.468671
min        -1.000000
25%         0.342315
50%         0.622643
75%         0.842248
max         0.999910
Name: Domain, dtype: float64
```
In [ ]:
```
(df['Domain'][df['Domain']==-1].count()/len(df))*100
```
Out[ ]:
```
6.153076538269135
```
In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['Domain'],kde=True)
plt.title('Normalized Domain Score Distribution')
```
Out[ ]:
```
Text(0.5, 1.0, 'Normalized Domain Score Distribution')
```


Observation

- Normalized Domain Score is left skewed. If we ignore -1.
- 6.15 % of candidates didn't filled their domain scores.

Computer Programming

In [ ]:
```
df['ComputerProgramming'].describe()
```
Out[ ]:
```
count    3998.000000
mean      353.102801
std       205.355519
min        -1.000000
25%       295.000000
50%       415.000000
75%       495.000000
max       840.000000
Name: ComputerProgramming, dtype: float64
```
In [ ]:
```
(df['ComputerProgramming'][df['ComputerProgramming']!=-1].count()/len(df))*100
```
Out[ ]:
```
78.28914457228613
```
In [ ]:
```
sns.histplot(x=df['ComputerProgramming'],kde=True)
```
Out[ ]:
```
<Axes: xlabel='ComputerProgramming', ylabel='Count'>
```



Observation

- 78.3 % of candidates attempted computer programming section
- Computer Programming scores are normally distributed

Electronics and Semicon

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

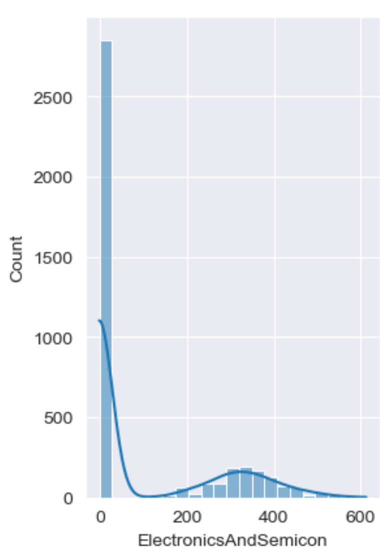
Out[ ]:
```
count    3998.000000
mean       95.328414
std       158.241218
min        -1.000000
25%        -1.000000
50%        -1.000000
75%       233.000000
max       612.000000
Name: ElectronicsAndSemicon, dtype: float64
```

In [ ]:
```
(df['ElectronicsAndSemicon'][df['ElectronicsAndSemicon']!=-1].count()/len(df))*100
```

Out[ ]:
```
28.61430715357679
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['ElectronicsAndSemicon'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='ElectronicsAndSemicon', ylabel='Count'>
```



Observation

- 28.6% of candidates gave ElectronicsAndSemicon exam

Computer Science

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

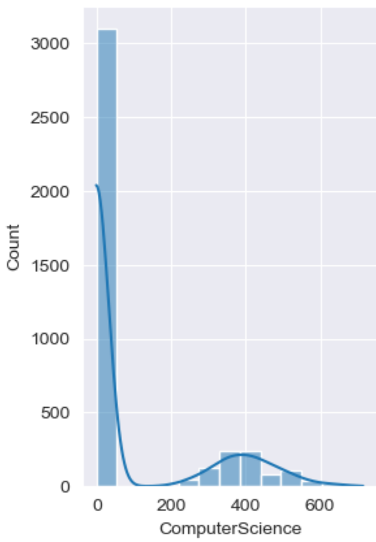
Out[ ]:
```
count    3998.000000
mean       90.742371
std       175.273083
min        -1.000000
25%        -1.000000
50%        -1.000000
75%        -1.000000
max       715.000000
Name: ComputerScience, dtype: float64
```

In [ ]:
```
(df['ComputerScience'][df['ComputerScience']!=-1].count()/len(df))*100
```

Out[ ]:
```
22.56128064032016
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['ComputerScience'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='ComputerScience', ylabel='Count'>
```



Observations

- 22.561% of candidates have attempted the Computer Science exam

Mechanical Engineering

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

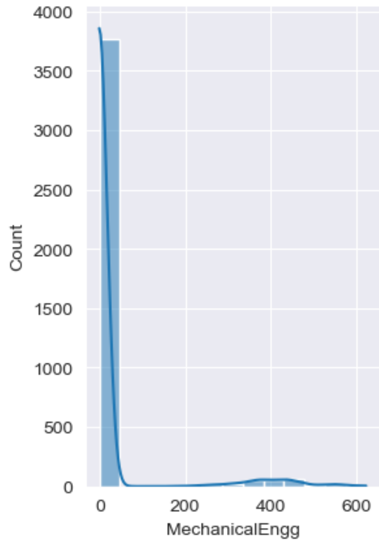
Out[ ]:
```
count    3998.000000
mean       22.974737
std        98.123311
min        -1.000000
25%        -1.000000
50%        -1.000000
75%        -1.000000
max       623.000000
Name: MechanicalEngg, dtype: float64
```

In [ ]:
```
(df['MechanicalEngg'][df['MechanicalEngg']!=-1].count()/len(df))*100
```

Out[ ]:
```
5.877938969484743
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['MechanicalEngg'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='MechanicalEngg', ylabel='Count'>
```



Observations

- 5.9 % candidates have attempted Mechanical Enggineering section

Electrical Engineering

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

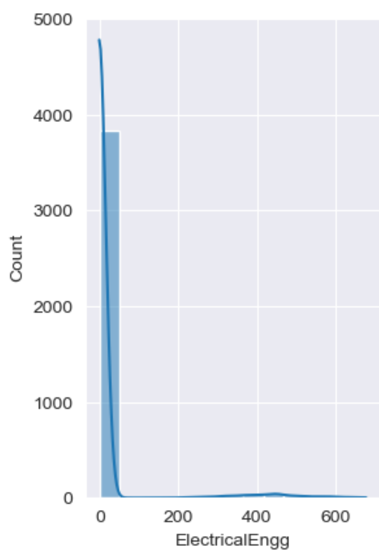
Out[ ]:
```
count    3998.000000
mean       16.478739
std        87.585634
min        -1.000000
25%        -1.000000
50%        -1.000000
75%        -1.000000
max       676.000000
Name: ElectricalEngg, dtype: float64
```

In [ ]:
```
(df['ElectricalEngg'][df['ElectricalEngg']!=-1].count()/len(df))*100
```

Out[ ]:
```
4.027013506753377
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['ElectricalEngg'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='ElectricalEngg', ylabel='Count'>
```



Observation

- Only 4% of candidates attempted Elctrical Engineering

Telecom Engg

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

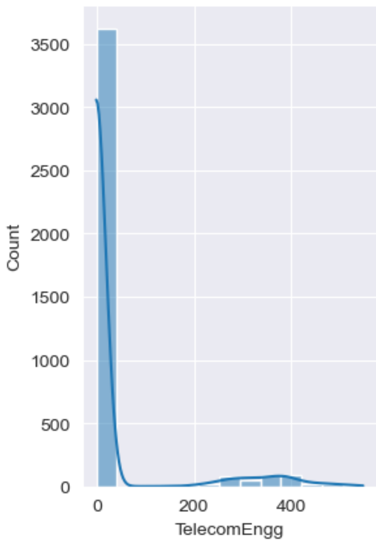
Out[ ]:
```
count    3998.000000
mean       31.851176
std       104.852845
min        -1.000000
25%        -1.000000
50%        -1.000000
75%        -1.000000
max       548.000000
Name: TelecomEngg, dtype: float64
```

In [ ]:
```
(df['TelecomEngg'][df['TelecomEngg']!=-1].count()/len(df))*100
```

Out[ ]:
```
9.354677338669335
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['TelecomEngg'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='TelecomEngg', ylabel='Count'>
```



Observation

- Only 9% candidates attempted telecom engg

Civil Engg

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

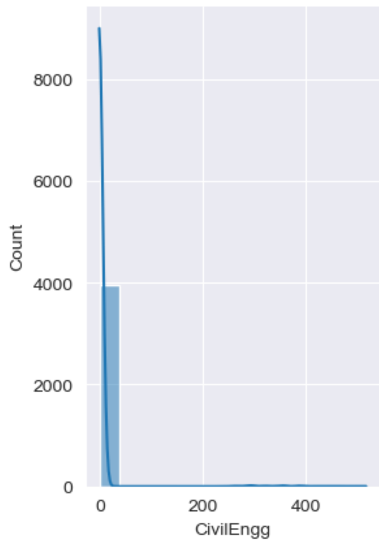
Out[ ]:
```
count    3998.000000
mean        2.683842
std        36.658505
min        -1.000000
25%        -1.000000
50%        -1.000000
75%        -1.000000
max       516.000000
Name: CivilEngg, dtype: float64
```

In [ ]:
```
(df['CivilEngg'][df['CivilEngg']!=-1].count()/len(df))*100
```

Out[ ]:
```
1.0505252626313157
```

In [ ]:
```
plt.subplot(121)
sns.histplot(x=df['CivilEngg'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='CivilEngg', ylabel='Count'>
```



Observation

- Only 1% of candidates attempted the CivilEngg

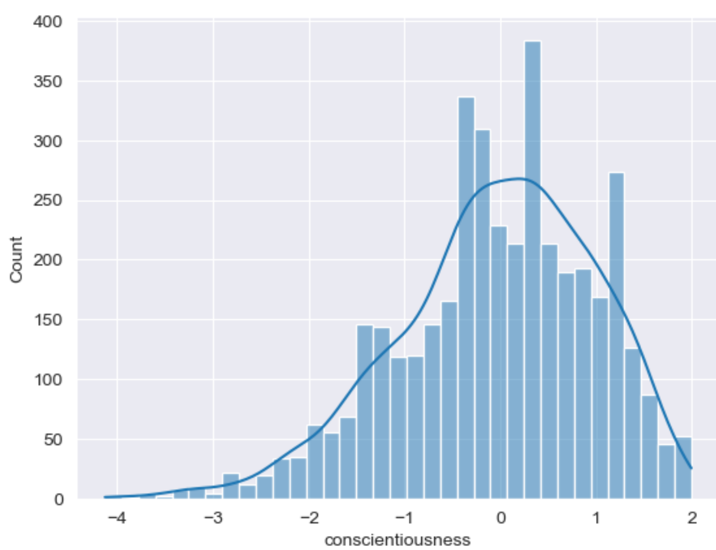
Personality Tests

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

Out[ ]:
```
count    3998.000000
mean       -0.037831
std         1.028666
min        -4.126700
25%        -0.713525
50%         0.046400
75%         0.702700
max         1.995300
Name: conscientiousness, dtype: float64
```

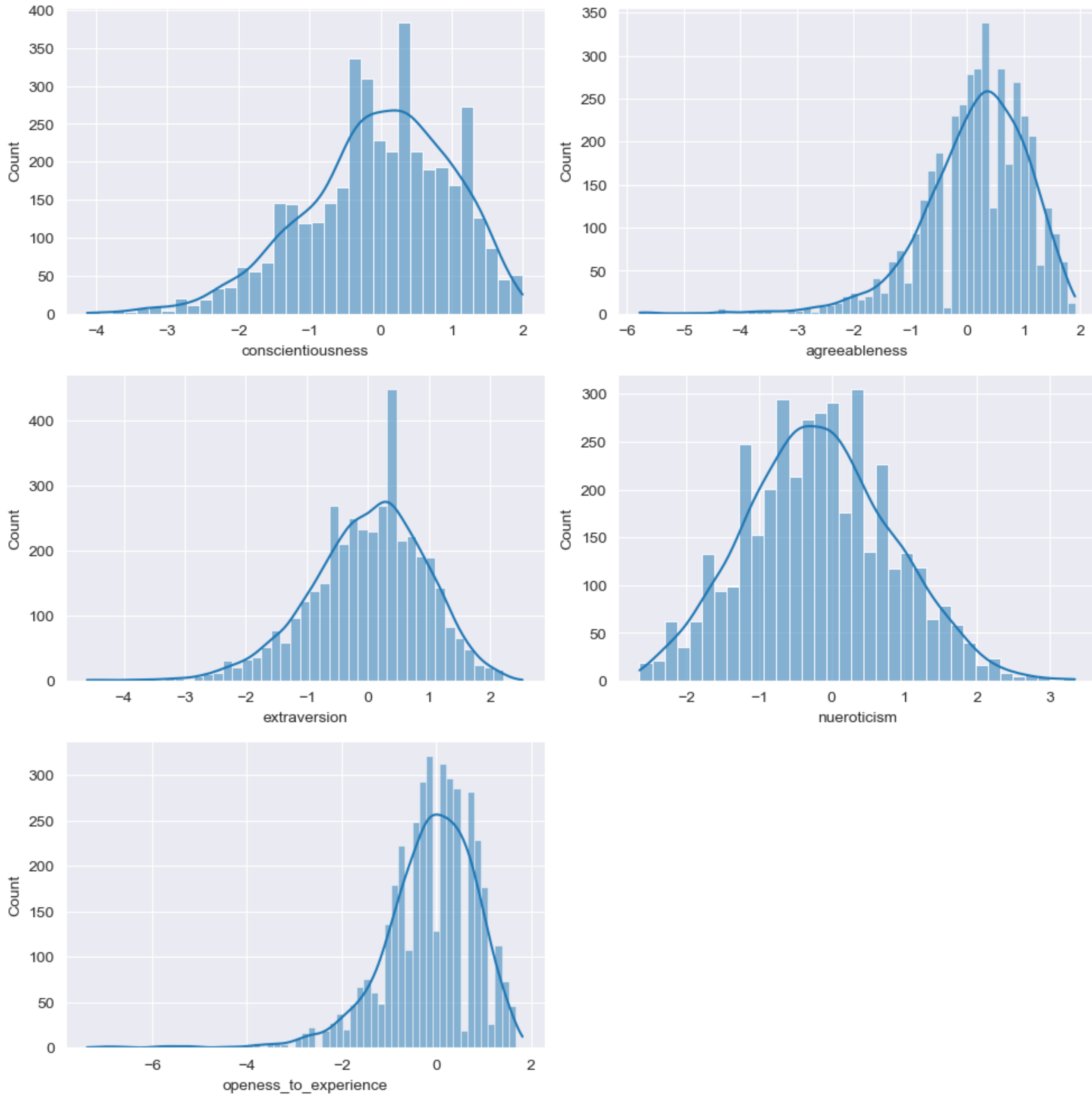
In [ ]:
```
sns.histplot(x=df['conscientiousness'],kde=True)
```

Out[ ]:
```
<Axes: xlabel='conscientiousness', ylabel='Count'>
```



In [ ]:
```
personality_test_scores=df.iloc[:,33:]
count=1
plt.figure(figsize=(10,10))
for col in (personality_test_scores.columns):

    plt.subplot(3,2,count)
    sns.histplot(x=df[col],kde=True)
    count+=1
plt.tight_layout()
```



Observation

- Except nueroticism, other personality traits are left skewed

Handling outliers

In [ ]:
```
def remove_outliers_iqr(df,column_name):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile)
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    # Calculate IQR (Interquartile Range)
    IQR = Q3 - Q1

    # Define the lower and upper bounds for outlier detection
    lower_bound = Q1 -1.5 * IQR
    upper_bound = Q3 +1.5* IQR

    def checkoutlier(x):
        if x<lower_bound or x>upper_bound:
            return None
        return x
    # Replace outliers with None
    return df[column_name].apply(checkoutlier)
```

In [ ]:
```
remove_outliers_iqr(df,'Salary')
```

Out[ ]:
```
0       4.20
1       5.00
2       3.25
3        NaN
4       2.00
        ...
3993    2.80
3994    1.00
3995    3.20
3996    2.00
3997    4.00
Name: Salary, Length: 3998, dtype: float64
```

In [ ]:
```
numerical_cols=[col for col in df.columns if df[col].dtype!='O'][1:]
categorical_cols=[col for col in df.columns if df[col].dtype=='O'][1:]
clean_df=df[numerical_cols]
clean_df
```

Out[ ]:

	Salary	DOJ	DOL	DOB	10percentage	12graduation	12percentage	CollegeID	CollegeGPA	CollegeCityID	...	ComputerScience	MechanicalEngg	ElectricalEn
0	4.20	2012-06-01	2050-12-31	1990-02-19	84.30	2007	95.80	1141	7.800	1141	...	-1	-1	
1	5.00	2013-09-01	2050-12-31	1989-10-04	85.40	2007	85.00	5807	7.006	5807	...	-1	-1	
2	3.25	2014-06-01	2050-12-31	1992-08-03	85.00	2010	68.20	64	7.000	64	...	-1	-1	
3	11.00	2011-07-01	2050-12-31	1989-12-05	85.60	2007	83.60	6920	7.464	6920	...	-1	-1	
4	2.00	2014-03-01	2015-03-01	1991-02-27	78.00	2008	76.80	11368	7.390	11368	...	-1	-1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3993	2.80	2011-10-01	2012-10-01	1987-04-15	52.09	2006	55.50	6268	6.150	6268	...	-1	-1	
3994	1.00	2013-07-01	2013-07-01	1992-08-27	90.00	2009	93.00	4883	7.730	4883	...	-1	-1	
3995	3.20	2013-07-01	2050-12-31	1991-07-03	81.86	2008	65.50	9786	7.000	9786	...	-1	-1	
3996	2.00	2014-07-01	2015-01-01	1992-03-20	78.72	2010	69.88	979	7.042	979	...	438	-1	
3997	4.00	2013-02-01	2050-12-31	1991-02-26	70.60	2008	68.00	6609	6.800	6609	...	-1	-1	

3998 rows × 27 columns

In [ ]:
```
for col in numerical_cols:
    clean_df.loc[:,col]=remove_outliers_iqr(df,col)
clean_df
```

Out[ ]:

	Salary	DOJ	DOL	DOB	10percentage	12graduation	12percentage	CollegeID	CollegeGPA	CollegeCityID	...	ComputerScience	MechanicalEngg	ElectricalEn
0	4.20	2012-06-01	2050-12-31	1990-02-19	84.30	2007.0	95.80	1141	7.800	1141	...	-1.0	-1.0	-
1	5.00	2013-09-01	2050-12-31	1989-10-04	85.40	2007.0	85.00	5807	7.006	5807	...	-1.0	-1.0	-
2	3.25	2014-06-01	2050-12-31	1992-08-03	85.00	2010.0	68.20	64	7.000	64	...	-1.0	-1.0	-
3	NaN	2011-07-01	2050-12-31	1989-12-05	85.60	2007.0	83.60	6920	7.464	6920	...	-1.0	-1.0	-
4	2.00	2014-03-01	2015-03-01	1991-02-27	78.00	2008.0	76.80	11368	7.390	11368	...	-1.0	-1.0	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3993	2.80	2011-10-01	2012-10-01	1987-04-15	52.09	2006.0	55.50	6268	6.150	6268	...	-1.0	-1.0	-
3994	1.00	2013-07-01	2013-07-01	1992-08-27	90.00	2009.0	93.00	4883	7.730	4883	...	-1.0	-1.0	-
3995	3.20	2013-07-01	2050-12-31	1991-07-03	81.86	2008.0	65.50	9786	7.000	9786	...	-1.0	-1.0	-
3996	2.00	2014-07-01	2015-01-01	1992-03-20	78.72	2010.0	69.88	979	7.042	979	...	NaN	-1.0	-
3997	4.00	2013-02-01	2050-12-31	1991-02-26	70.60	2008.0	68.00	6609	6.800	6609	...	-1.0	-1.0	-

3998 rows × 27 columns

In [ ]:
```
clean_df=pd.concat([clean_df,df[categorical_cols]],axis=1)
```

In [ ]:
```
clean_df.isnull().sum()
```

Out[ ]:
```
Salary                    109
DOJ                        19
DOL                         0
DOB                        68
10percentage               30
12graduation               45
12percentage                1
CollegeID                   0
CollegeGPA                 27
CollegeCityID               0
GraduationYear              2
English                    15
Logical                    18
Quant                      25
Domain                    246
ComputerProgramming         2
ElectronicsAndSemicon       2
ComputerScience           902
MechanicalEngg            235
ElectricalEngg            161
TelecomEngg               374
CivilEngg                  42
conscientiousness          39
agreeableness             123
extraversion               40
nueroticism                15
openess_to_experience      95
JobCity                   461
Gender                      0
10board                     0
12board                     0
CollegeTier                 0
Degree                      0
Specialization              0
CollegeCityTier             0
CollegeState                0
dtype: int64
```

In [ ]:
```
clean_df.dropna(axis=0,inplace=True)
```

In [ ]:
```
clean_df.isna().sum()
```

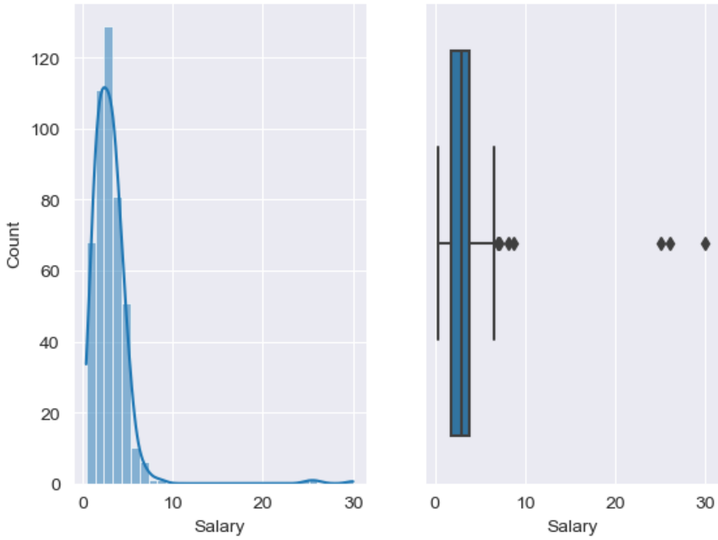
Out[ ]:
```
Salary                   0
DOJ                      0
DOL                      0
DOB                      0
10percentage             0
12graduation             0
12percentage             0
CollegeID                0
CollegeGPA               0
CollegeCityID            0
GraduationYear    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

Bivariate Analysis

Q Salary range of candidates who didn't mentioned their job City

In [ ]:
```
plt.figure9figsize=(10,10)
plt.subplot(121)
sns.histplot(df['Salary'][df['JobCity'].isna()],kde=True,bins=30)
plt.subplot(122)
sns.boxplot(x=df['Salary'][df['JobCity'].isna()])
```

Out[ ]: <Axes: xlabel='Salary'>



Observaton

- Most candidates belong to salary range <10L

In [ ]:

Candidates who didn't mentioned Job City and who didn't mentioned they DOL

In [ ]:
```
df[(df['JobCity'].isna())&(df['JobCity']=='2050-12-31')]
```

Out[ ]:

ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	10board	...	ComputerScience	MechanicalEngg	ElectricalEngg	TelecomEngg	CivilEngg	c

0 rows × 38 columns

Observations

- Candidates who didn't mentioned Job City and who didn't mentioned they DOL are none

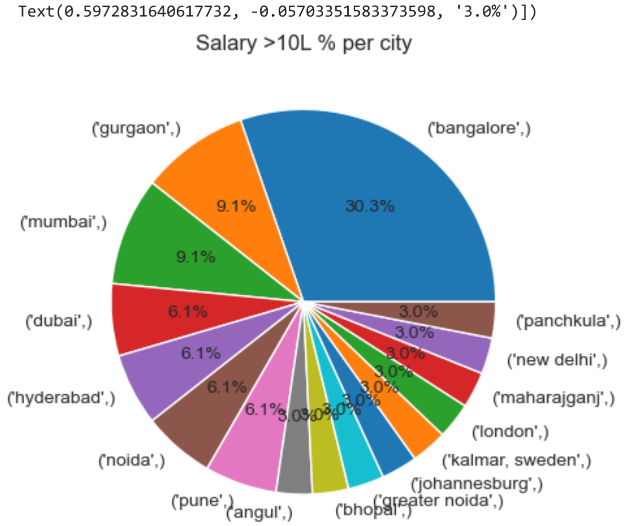
Highest package count in different JobCities

In [ ]:
```
salary_gret_10_percity=df[['JobCity','Salary']][(df['Salary']>10)].value_counts(subset=['JobCity'])
salary_gret_10_percity
```

Out[ ]:
```
JobCity
bangalore         10
gurgaon            3
mumbai             3
dubai              2
hyderabad          2
noida              2
pune               2
angul              1
bhopal             1
greater noida      1
johannesburg       1
kalmar, sweden     1
london             1
maharajganj        1
new delhi          1
panchkula          1
Name: count, dtype: int64
```

In [ ]:
```
plt.title('Salary >10L % per city')
plt.pie(x=salary_gret_10_percity.to_list(),labels=salary_gret_10_percity.index,autopct='%1.1f%%')
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x2138f08b100>,
  <matplotlib.patches.Wedge at 0x2138bd70fa0>,
  <matplotlib.patches.Wedge at 0x2138f08bd30>,
  <matplotlib.patches.Wedge at 0x2138f099400>,
  <matplotlib.patches.Wedge at 0x2138f099a90>,
  <matplotlib.patches.Wedge at 0x2138f0a5160>,
  <matplotlib.patches.Wedge at 0x2138f0a57f0>,
  <matplotlib.patches.Wedge at 0x2138f0a5e20>,
  <matplotlib.patches.Wedge at 0x2138f0b34f0>,
  <matplotlib.patches.Wedge at 0x2138f0b3b80>,
  <matplotlib.patches.Wedge at 0x2138f0c0280>,
  <matplotlib.patches.Wedge at 0x2138f02fb20>,
  <matplotlib.patches.Wedge at 0x2138f0c0a00>,
  <matplotlib.patches.Wedge at 0x2138f0d20d0>,
  <matplotlib.patches.Wedge at 0x2138f0d2760>,
  <matplotlib.patches.Wedge at 0x2138f0d2df0>],
 [Text(0.6380625751062828, 0.8960335653583181, "('bangalore',)"),
  Text(-0.6380626589989968, 0.8960335056185845, "('gurgaon',)"),
  Text(-1.0212047599553085, 0.4088286172036162, "('mumbai',)"),
  Text(-1.0950191047439604, -0.10456175326444932, "('dubai',)"),
  Text(-0.9777189391775435, -0.5040492793106041, "('hyderabad',)"),
  Text(-0.7203467082780743, -0.8313246176271597, "('noida',)"),
  Text(-0.3597746239838725, -1.0395009475403394, "('pune',)"),
  Text(-0.052339954655623376, -1.098754080377701, "('angul',)"),
  Text(0.15654647964602333, -1.0888035634174043, "('bhopal',)"),
  Text(0.3597749159591659, -1.0395008464867046, "('greater noida',)"),
  Text(0.5500001486524352, -0.9526278583383436, "('johannesburg',)"),
  Text(0.7203469417807291, -0.8313244152959488, "('kalmar, sweden',)"),
  Text(0.8646585180402889, -0.679974740104639, "('london',)"),
  Text(0.9777190807550031, -0.5040490046886235, "('maharajganj',)"),
  Text(1.0554423263691122, -0.30990562387371495, "('new delhi',)"),
  Text(1.095019134113251, -0.10456144569518265, "('panchkula',)")],
 [Text(0.34803413187615423, 0.4887455811045371, '30.3%'),
  Text(-0.34803417763581646, 0.48874554851922786, '9.1%'),
  Text(-0.5570207781574409, 0.2229974275656088, '9.1%'),
  Text(-0.5972831480421602, -0.05703368359879053, '6.1%'),
  Text(-0.5333012395513873, -0.2749359705330567, '6.1%'),
  Text(-0.39291638633349507, -0.45344979143299613, '6.1%'),
  Text(-0.19624070399120316, -0.567000516840185, '6.1%'),
  Text(-0.028549066175794564, -0.5993204074787459, '3.0%'),
  Text(0.08538898889783089, -0.5938928527731295, '3.0%'),
  Text(0.1962408632504541, -0.5670004617200206, '3.0%'),
  Text(0.30000008108314646, -0.5196151954572783, '3.0%'),
  Text(0.39291651369857944, -0.4534496810705174, '3.0%'),
  Text(0.47163191893106665, -0.3708953127843485, '3.0%'),
  Text(0.5333013167754562, -0.2749358207392491, '3.0%'),
  Text(0.5756958143831521, -0.1690394312038445, '3.0%'),
  Text(0.5972831640617732, -0.05703351583373598, '3.0%')])
```

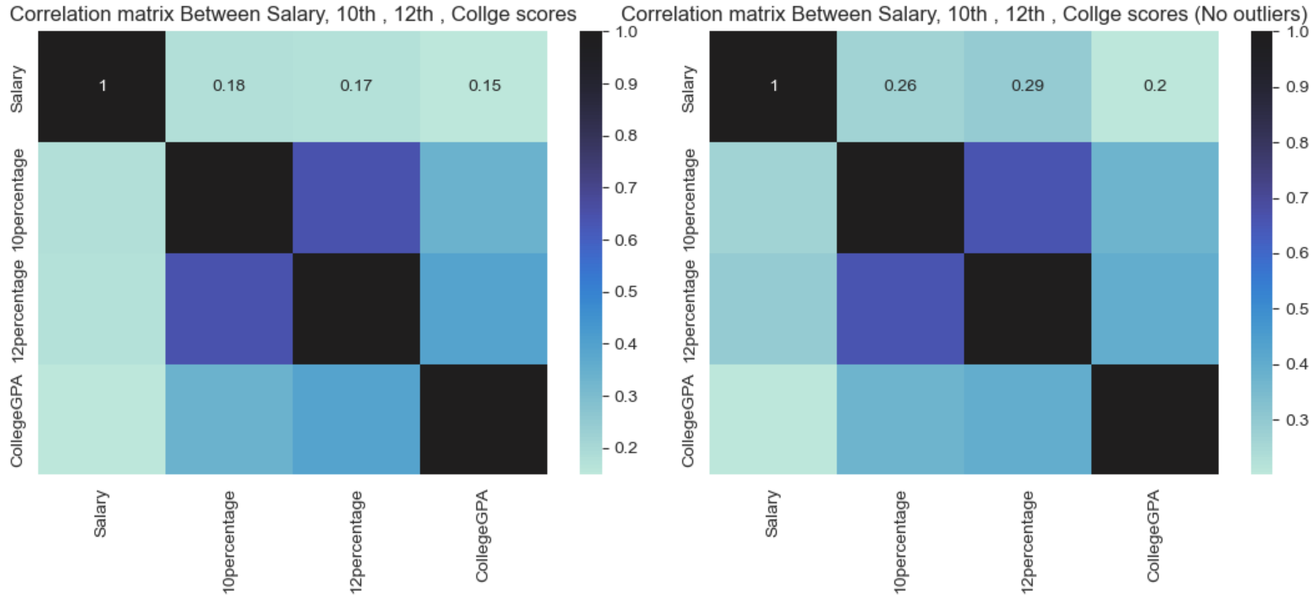


Observation

- 30.3% of salary packages greater than 10LPA were given in banglore based companies

Relation between Salary and 10, 12 th, GPA of the candidates

In [ ]:
```
plt.figure(figsize=(11,5))
plt.subplot(121)
plt.title('Correlation matrix Between Salary, 10th , 12th , Collge scores')
sns.heatmap(df[['Salary','10percentage','12percentage','CollegeGPA']].corr(),annot=True,center=1)
plt.subplot(122)
plt.title('Correlation matrix Between Salary, 10th , 12th , Collge scores (No outliers)')
sns.heatmap(clean_df[['Salary','10percentage','12percentage','CollegeGPA']].corr(),annot=True,center=1)
plt.tight_layout()
```



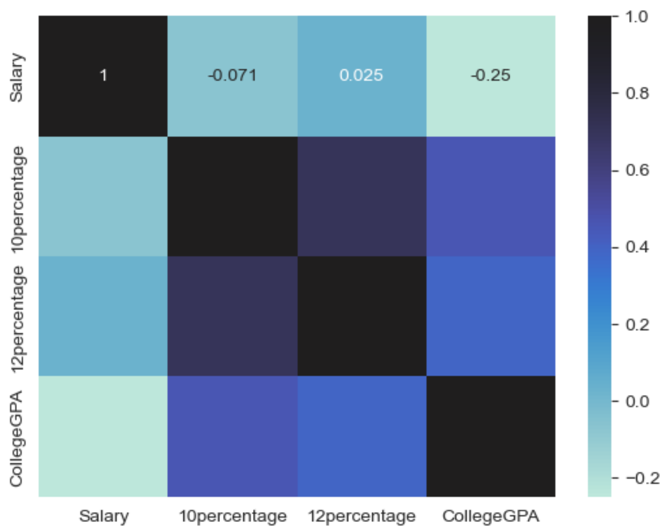
Observation

- There is no significant relationship between salary and other scores

Relation between Salary>10Lakhs and the score(10,12, college)

In [ ]:
```
sns.heatmap(df[df['Salary']>10][['Salary','10percentage','12percentage','CollegeGPA']].corr(),annot=True,center=1)
```

Out[ ]: <Axes: >



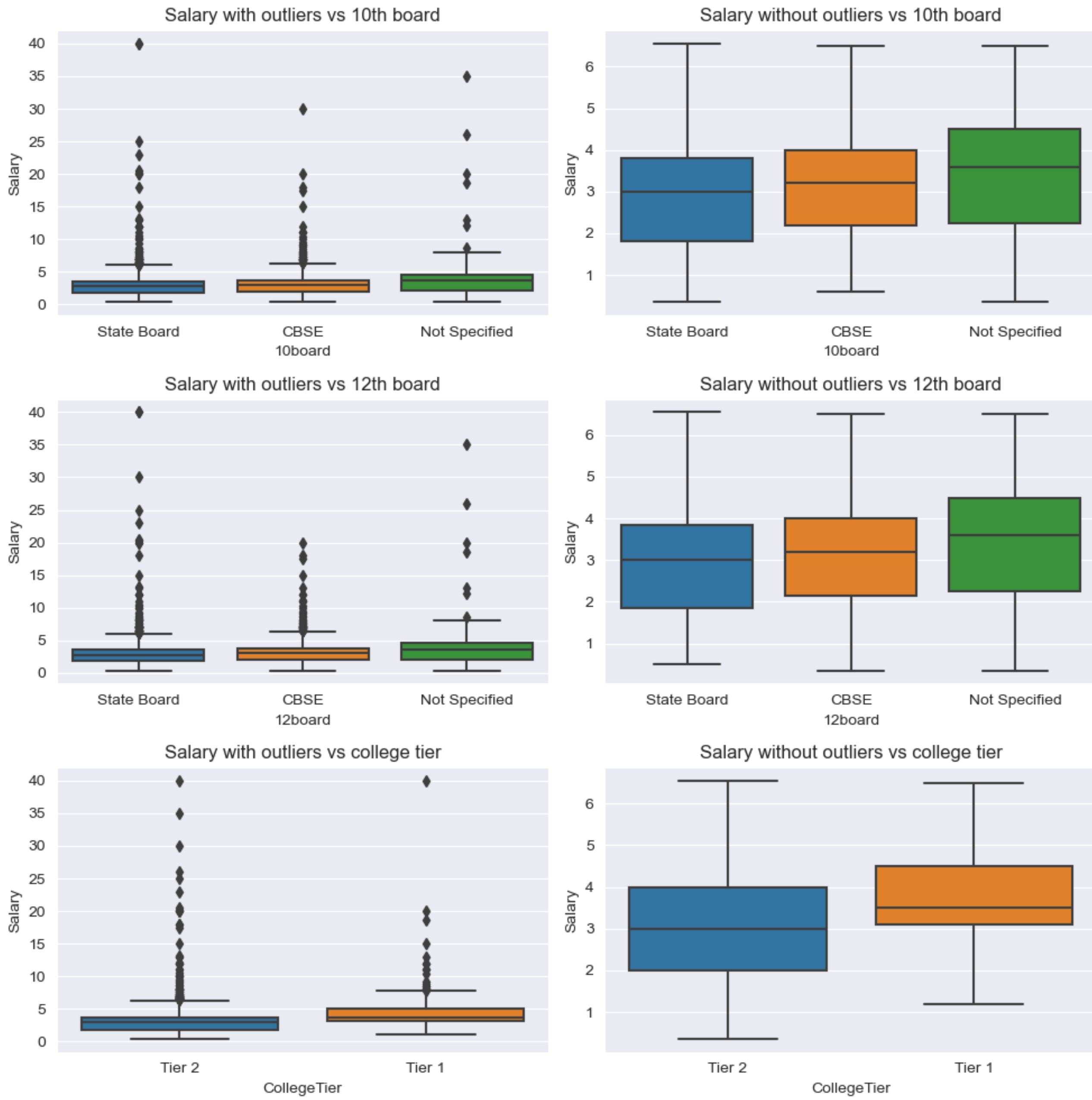
Observation

- There is no significant relationship between candidates with salary >10 lakhs and their 10,12,college scores

Salary vs 10th, 12th board, Collge Tier

In [ ]:
```
plt.figure(figsize=(10,10))
plt.subplot(321)
plt.title('Salary with outliers vs 10th board')
sns.boxplot(data=df[['Salary','10board']],x='10board',y='Salary')
plt.subplot(322)
plt.title('Salary without outliers vs 10th board')
sns.boxplot(data=clean_df[['Salary','10board']],x='10board',y='Salary')
plt.subplot(323)
plt.title('Salary with outliers vs 12th board')
sns.boxplot(data=df[['Salary','12board']],x='12board',y='Salary')
plt.subplot(324)
plt.title('Salary without outliers vs 12th board')
sns.boxplot(data=clean_df[['Salary','12board']],x='12board',y='Salary')
plt.subplot(325)
plt.title('Salary with outliers vs college tier')
sns.boxplot(data=df[['Salary','CollegeTier']],x='CollegeTier',y='Salary')
```

```
plt.subplot(326)
plt.title('Salary without outliers vs college tier')
sns.boxplot(data=clean_df[['Salary','CollegeTier']],x='CollegeTier',y='Salary')
plt.tight_layout()
```



Observation

- Candidates who studied in CBSE board and also in Tier 1 college tend to get a better median salary

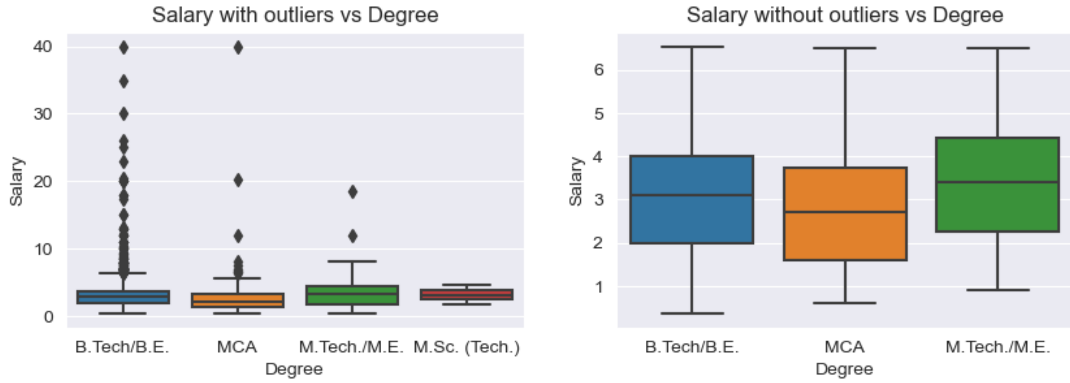
Degree and Specialization vs Salary

In [ ]:

```
plt.figure(figsize=(10,10))
plt.subplot(321)
plt.title('Salary with outliers vs Degree')
sns.boxplot(data=df[['Salary','Degree']],x='Degree',y='Salary')
plt.subplot(322)
plt.title('Salary without outliers vs Degree')
sns.boxplot(data=clean_df[['Salary','Degree']],x='Degree',y='Salary')
```

Out[ ]:

```
<Axes: title={'center': 'Salary without outliers vs Degree'}, xlabel='Degree', ylabel='Salary'>
```



Observations

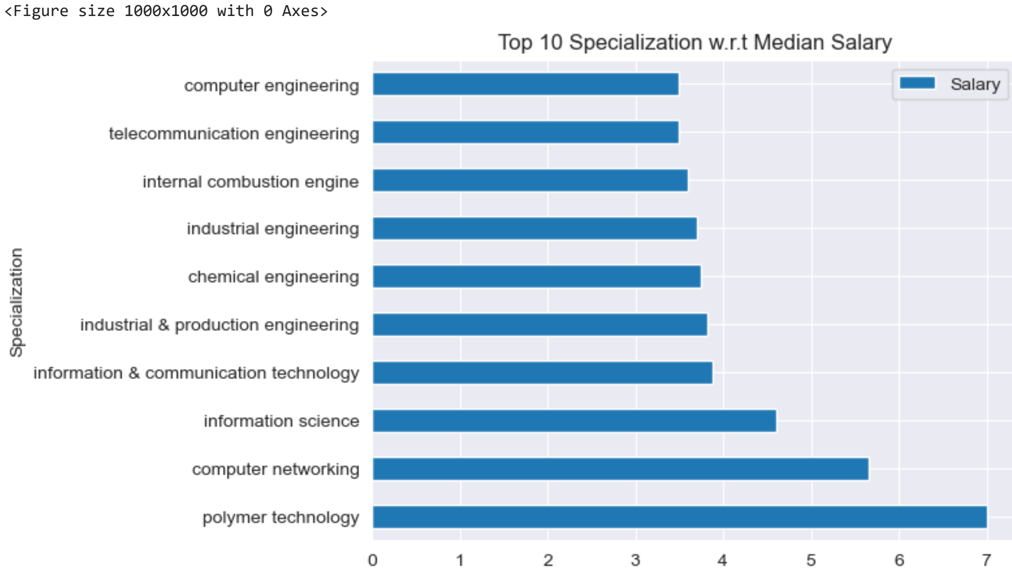
- The density of outliers is more in B.Tech/BE degree than M.Tech/ME. Hence, higher packages are more likely be secured by studying B.Tech.
- But when outliers are removed, M.Tech offers better median package than even MCA.

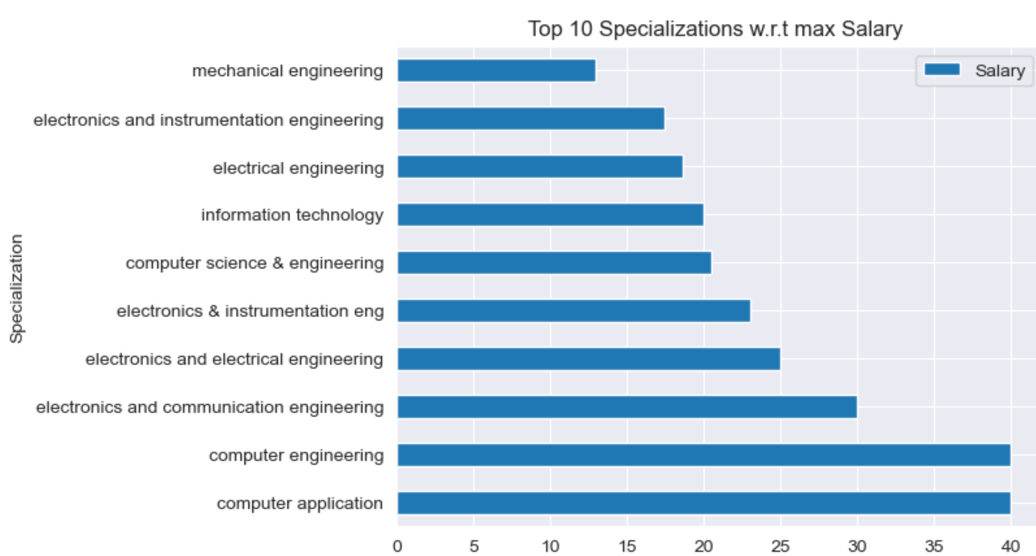
In [ ]:
```
salary_special=df[['Salary','Specialization']].groupby(by='Specialization')
```

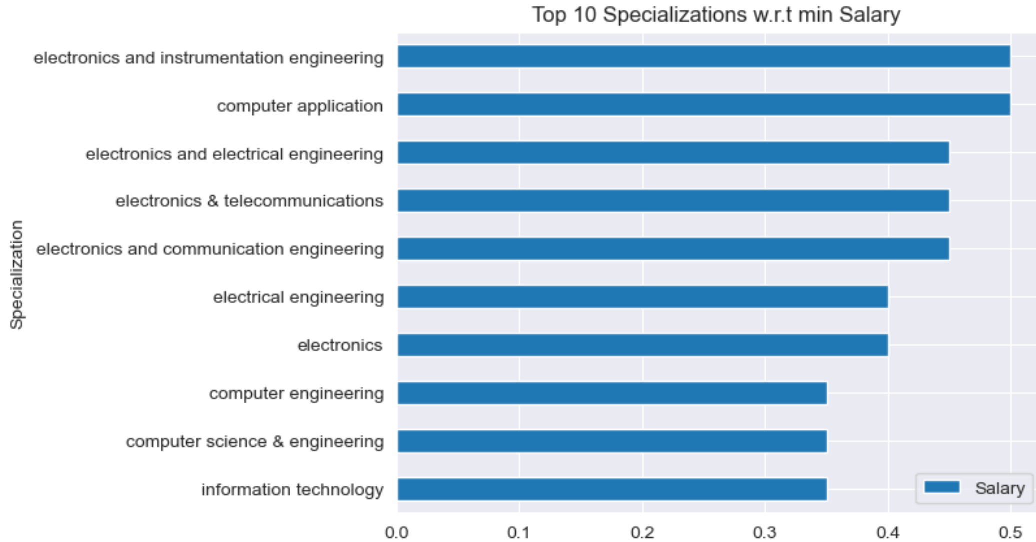
In [ ]:
```
plt.figure(figsize=(10,10))
salary_special.median().sort_values(by='Salary',ascending=False)[:10].plot(kind='barh')
plt.title('Top 10 Specialization w.r.t Median Salary')
salary_special.max().sort_values(by='Salary',ascending=False)[:10].plot(kind='barh')
plt.title('Top 10 Specializations w.r.t max Salary ')
salary_special.min().sort_values(by='Salary')[:10].plot(kind='barh')
plt.title('Top 10 Specializations w.r.t min Salary ')
```

Out[ ]:
```
Text(0.5, 1.0, 'Top 10 Specializations w.r.t min Salary ')
```

```
<Figure size 1000x1000 with 0 Axes>
```







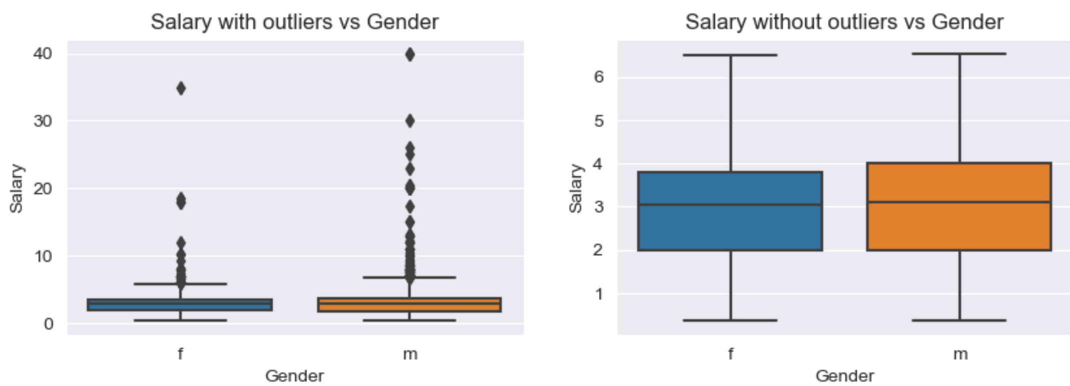
Observation

- Max Median salary is offered by Polymer Technology Specialization
- Max Salary is offered by Computer Application Specialization
- Min Salary is offered by Information Technology Specialization

Gender vs Salary

In [ ]:
```
plt.figure(figsize=(10,10))
plt.subplot(321)
plt.title('Salary with outliers vs Gender')
sns.boxplot(data=df[['Salary','Gender']],x='Gender',y='Salary')
plt.subplot(322)
plt.title('Salary without outliers vs Gender')
sns.boxplot(data=clean_df[['Salary','Gender']],x='Gender',y='Salary')
```

Out[ ]:
```
<Axes: title={'center': 'Salary without outliers vs Gender'}, xlabel='Gender', ylabel='Salary'>
```



Observation

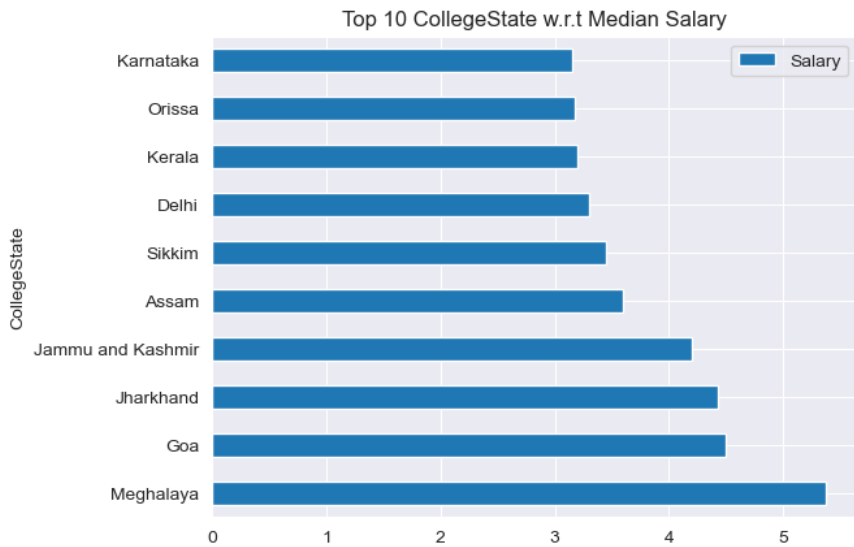
- Median compensation between male and female candidates is similar.
- But the density of outliers is much more in male candidates than female candidates

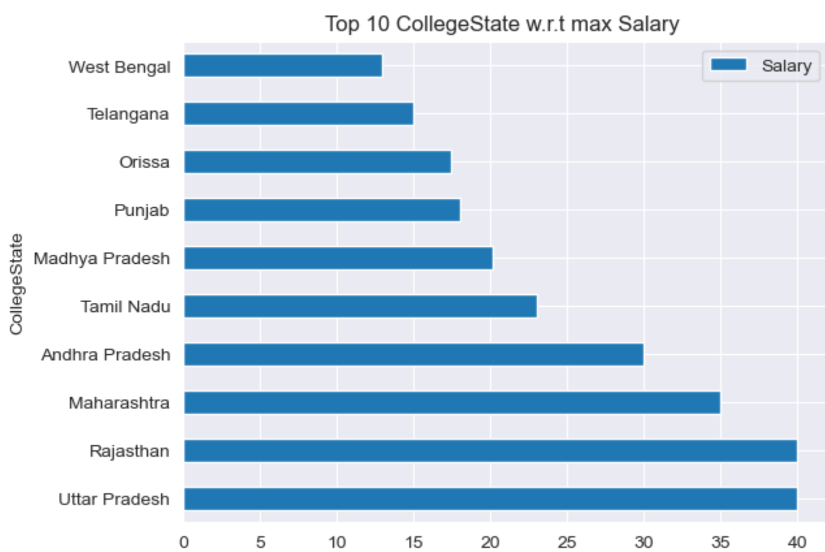
Relationship between College State and Salary

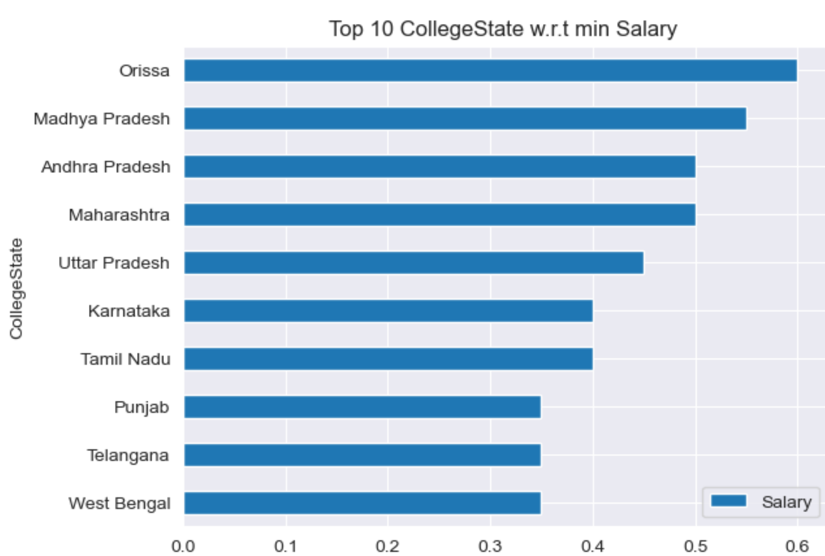
In [ ]:
```
salary_collegestate=df[['Salary','CollegeState']].groupby(by='CollegeState')
```

In [ ]:
```
plt.figure(figsize=(10,10))
salary_collegestate.median().sort_values(by='Salary',ascending=False)[:10].plot(kind='barh')
plt.title('Top 10 CollegeState w.r.t Median Salary')
salary_collegestate.max().sort_values(by='Salary',ascending=False)[:10].plot(kind='barh')
plt.title('Top 10 CollegeState w.r.t max Salary ')
salary_collegestate.min().sort_values(by='Salary')[:10].plot(kind='barh')
plt.title('Top 10 CollegeState w.r.t min Salary ')
```

Out[ ]:
```
Text(0.5, 1.0, 'Top 10 CollegeState w.r.t min Salary ')
<Figure size 1000x1000 with 0 Axes>
```







Observation

- Max Median packages offered to candidates studied at Meghalaya
- Maximum package were offered to candidates who studied in a UP college
- Minimum package was offered to candidate who studied in a WB college

Maximum Salary secured by a candidate who didn't gave any Computer related domain during AMCAT

In [ ]:
```
df[(df['ComputerProgramming']==-1) & (df['ComputerScience']==-1)]['Salary'].sort_values(ascending=False)
```

Out[ ]:
```
166     18.60
2493    17.45
2152    12.00
2230    12.00
123     12.00
        ...
3002     0.60
1385     0.50
1617     0.50
1957     0.45
3231     0.40
Name: Salary, Length: 824, dtype: float64
```

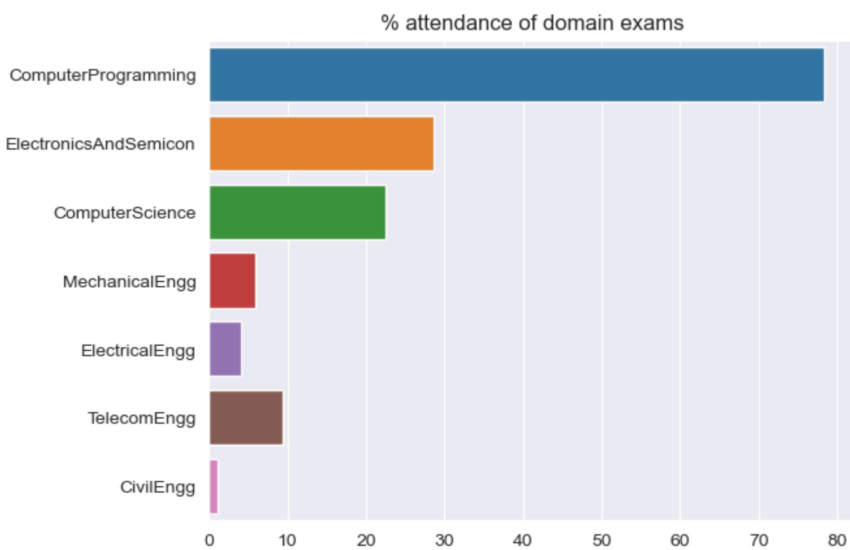
% of attendance to domain exams

In [ ]:
```
domains=df[['ComputerProgramming','ElectronicsAndSemicon', 'ComputerScience', 'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg']]
attendance=[((domains[col]!=-1).sum()/len(df))*100 for col in domains.columns]
attendance
```

Out[ ]:
```
[78.28914457228613,
 28.61430715357679,
 22.56128064032016,
 5.877938969484743,
 4.027013506753377,
 9.354677338669335,
 1.0505252626313157]
```

In [ ]:
```
plt.title('% attendance of domain exams')
sns.barplot(x=attendance,y=domains.columns)
```

Out[ ]:
```
<Axes: title={'center': '% attendance of domain exams'}>
```

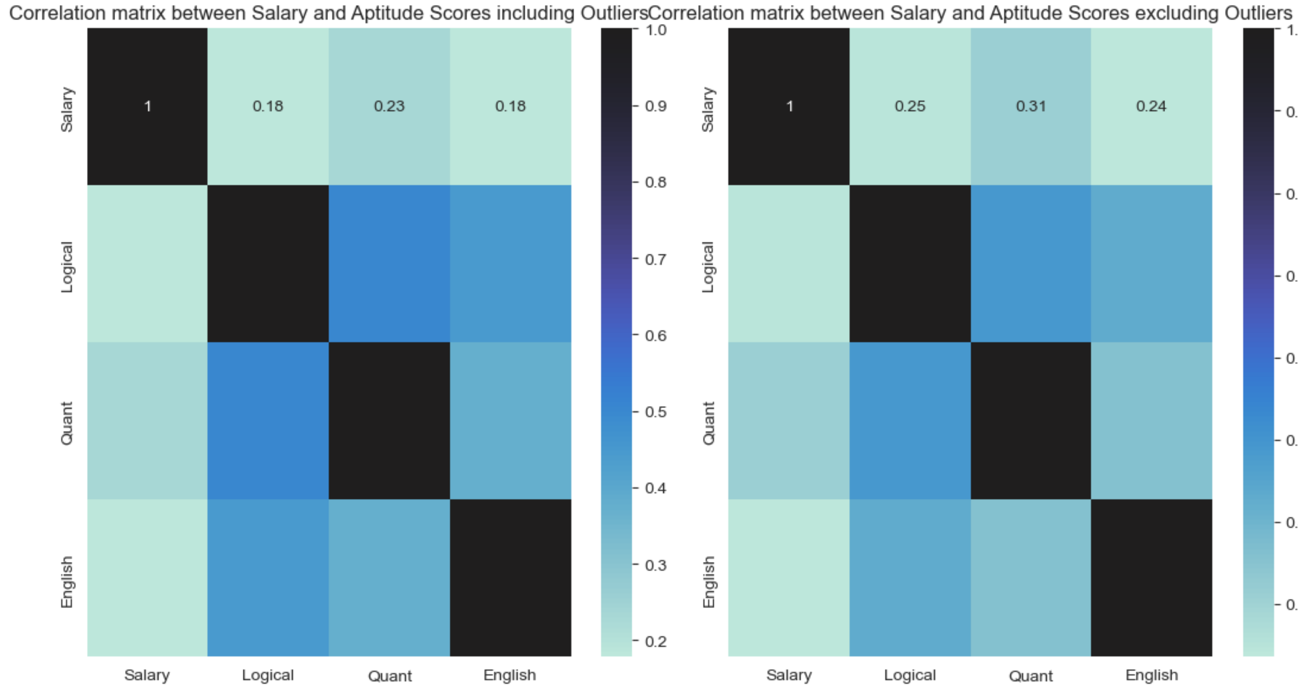


Observation

- 78% of Candidates gave Computer Programming domain exam in Amcat

Relationship between Salary and Logical,English,Quant Scores

In [ ]:
```
plt.figure(figsize=(11,6))
plt.subplot(121)
plt.title('Correlation matrix between Salary and Aptitude Scores including Outliers')
sns.heatmap(df[['Salary','Logical','Quant','English']].corr(),annot=True,center=1)
plt.subplot(122)
plt.title('Correlation matrix between Salary and Aptitude Scores excluding Outliers')
sns.heatmap(clean_df[['Salary','Logical','Quant','English']].corr(),annot=True,center=1)
plt.tight_layout()
```

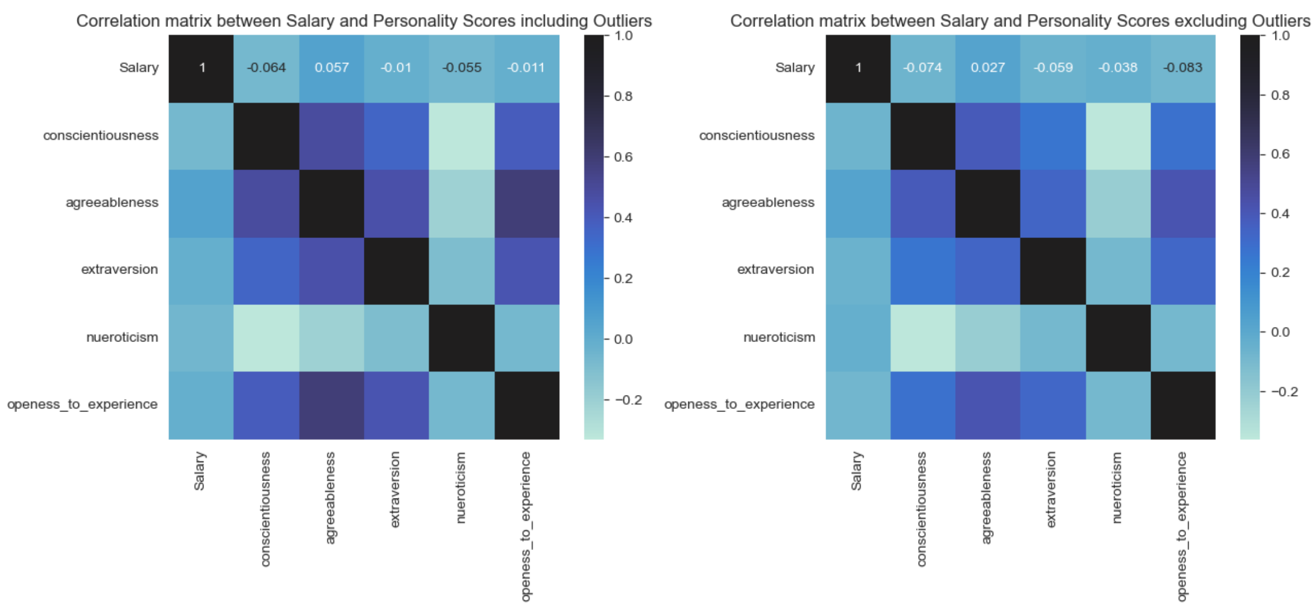


Observation

- Quantitative Reasoning has highest correlaiton with the salary

Relationship between Salary and Big 5 Personality Checks

In [ ]:
```
plt.figure(figsize=(13,6))
plt.subplot(121)
plt.title('Correlation matrix between Salary and Personality Scores including Outliers')
sns.heatmap(df[['Salary','conscientiousness', 'agreeableness', 'extraversion', 'nueroticism','openess_to_experience']].corr(),annot=True,cente
plt.subplot(122)
plt.title('Correlation matrix between Salary and Personality Scores excluding Outliers')
sns.heatmap(clean_df[['Salary','conscientiousness', 'agreeableness', 'extraversion', 'nueroticism','openess_to_experience']].corr(),annot=True
plt.tight_layout()
```



Observation

- Personality scores and Salary are not highly correlated

Research Question

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

H0- Mean salary is 2.75L

H1- Mean salary is not 2.75L

In [ ]:
```python
# Consider all specializations which have computer word in it
computer_jobs=df[['Specialization','Designation','Salary']][df['Specialization'].str.contains('computer',regex=False)]
t_stats,p_val,result=[],[],[]
def t_test(designation):
    t_statistic, p_value =  stats.ttest_1samp(computer_jobs['Salary'][computer_jobs['Designation']==designation],2.75)

    print("T-statistic:", t_statistic)
    print("P-value:", p_value)
    t_stats.append(t_statistic)
    p_val.append(p_value)

    if t_statistic<0 or t_statistic==None:
        print('Not enough data')
    elif p_value<0.05 and t_statistic>0:
        print('Null Hypothesis is rejected---Mean Salary for {} is not equal to 2.75'.format(designation))
    else:
        print('Null Hypothesis is not rejected---Mean Salary for {} is  equal to 2.75'.format(designation))
```

In [ ]:
```python
for designation in ['programmer analyst','software engineer','hardware engineer','associate engineer']:
    t_test(designation)
```

```
T-statistic: 9.662583462303736
P-value: 1.5624620958175607e-13
Null Hypothesis is rejected---Mean Salary for programmer analyst is not equal to 2.75
T-statistic: 7.816693023627976
P-value: 1.1365392982872651e-13
Null Hypothesis is rejected---Mean Salary for software engineer is not equal to 2.75
T-statistic: nan
P-value: nan
Null Hypothesis is not rejected---Mean Salary for hardware engineer is  equal to 2.75
T-statistic: -0.7019212139086011
P-value: 0.53328424401178
Not enough data
```

In [ ]:
```python
result=['Null Hypothesis is rejected','Null Hypothesis is rejected','Not Enough Data','Not Enough Data']
result_df=pd.DataFrame(data={'Designation':['programmer analyst','software engineer','hardware engineer','associate engineer'],
                             'T-statistic':t_stats,'p-value':p_val,'Result':result})
result_df
```

Out[ ]:

	Designation	T-statistic	p-value	Result
0	programmer analyst	9.662583	0.000001	Null Hypothesis is rejected
1	software engineer	7.816693	0.000001	Null Hypothesis is rejected
2	hardware engineer	NaN	0.000001	Not Enough Data
3	associate engineer	-0.701921	0.000001	Not Enough Data

Observation

- For Programmer analyst and Software Engineer Specializations, the mean salary is not 2.75 Lakhs as mentioned in the news article
- For Hardware Engineer and Associate Engineer, there is not enough data to perform a accurate Hypothesis testing

Specialization vs Gender

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

Ho- Null Hypothesis: There does not exist a significant relationship

H1- Alternate Hypthesis: There does exist a significant relationship.

In [ ]:
```python
from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(df['Specialization'],df['Gender'])
chi2_stat, p_val, dof, expected = chi2_contingency(contingency_table)
# Print the results
print("Chi-square statistic:", chi2_stat)
print("P-value:", p_val)
print(p_val<0.05)
```

```
Chi-square statistic: 104.46891913608454
P-value: 1.2453868176977011e-06
True
```

Chi2 statistic	104.46891913608454
p-value	1.2453868176977011e-06
result	Null Hypothesis is rejected

As p-val is less than 0.05. We reject null hypothesis, thus there exist significant relation ship between specialization and gender

```
In [ ]:
```