amcat-eda-1

October 6, 2024

EDA Project - AMCAT Data Analysis

0.1 1. INTRODUCTION

0.1.1 DATASET DESCRIPTION

- The dataset under consideration contains information collected through the AMCAT (Aspiring Minds Computer Adaptive Test), a widely used employability assessment tool. This dataset comprises approximately 40 variables and 4000 data points, capturing various aspects of candidates' profiles, educational backgrounds, skill sets, and personality traits. The data includes attributes such as ID, salary, date of joining (DOJ), date of leaving (DOL), designation, job city, gender, date of birth (DOB), educational qualifications, college details, domain-specific scores, and personality traits

0.2 1.1 OBJECTIVE

- The primary objective of this exploratory data analysis (EDA) project is to gain insights into the relationships between different variables and to uncover patterns or trends within the dataset. By conducting a thorough analysis, we aim to extract valuable information that can aid in understanding the factors influencing salary, employment outcomes, and overall employability of individuals who have taken the AMCAT assessment.
- Through visualizations, statistical summaries, and correlation analyses, we seek to answer pertinent questions such as
- What is the distribution of salaries among the candida es? Are there any significant differences in salaries based on gender, educational qualifications, or specializa ion? How do personality traits correlate with employability metrics such as domain-specific scores and overall perfor ance? Are there any discernible patterns in the data regarding job cities, college tiers, or graduation years that could impact employability?

0.2.1 IMPORTING ALL NECESSSARY LIBRARIES

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

```python
import warnings
warnings.filterwarnings('ignore')

# choose a matplotlib style option
plt.style.use('seaborn-v0_8-bright')

# choose seaborn style option
sns.set_style('darkgrid')
```

[3]:
```python
df = pd.read_excel("data.xlsx")
```

[4]:
```python
head = df.head()
tail = df.tail()
shape = df.shape
description = df.describe()
```

[5]:
```python
head
```

[5]:
```
  Unnamed: 0      ID   Salary        DOJ                  DOL  \
0      train  203097   420000 2012-06-01              present
1      train  579905   500000 2013-09-01              present
2      train  810601   325000 2014-06-01              present
3      train  267447  1100000 2011-07-01              present
4      train  343523   200000 2014-03-01  2015-03-01 00:00:00

               Designation    JobCity Gender        DOB  10percentage  …  \
0  senior quality engineer  Bangalore      f 1990-02-19          84.3  …
1        assistant manager     Indore      m 1989-10-04          85.4  …
2         systems engineer    Chennai      f 1992-08-03          85.0  …
3  senior software engineer    Gurgaon      m 1989-12-05          85.6  …
4                      get    Manesar      m 1991-02-27          78.0  …

  ComputerScience  MechanicalEngg  ElectricalEngg TelecomEngg  CivilEngg  \
0              -1              -1              -1          -1         -1
1              -1              -1              -1          -1         -1
2              -1              -1              -1          -1         -1
3              -1              -1              -1          -1         -1
4              -1              -1              -1          -1         -1

   conscientiousness agreeableness extraversion  nueroticism  \
0             0.9737        0.8128       0.5269      1.35490
1            -0.7335        0.3789       1.2396     -0.10760
2             0.2718        1.7109       0.1637     -0.86820
3             0.0464        0.3448      -0.3440     -0.40780
4            -0.8810       -0.2793      -1.0697      0.09163
```

```
   openess_to_experience
0                -0.4455
1                 0.8637
2                 0.6721
3                -0.9194
4                -0.1295

[5 rows x 39 columns]
```

[6]:
```
tail
```

[6]:
```
     Unnamed: 0      ID    Salary               DOJ               DOL  \
3993      train   47916  280000.0  01-10-2011 00:00  01-10-2012 00:00
3994      train  752781  100000.0  01-07-2013 00:00  01-07-2013 00:00
3995      train  355888  320000.0  01-07-2013 00:00           present
3996      train  947111  200000.0  01-07-2014 00:00  01-01-2015 00:00
3997      train  324966  400000.0  01-02-2013 00:00           present

                      Designation          JobCity Gender              DOB  \
3993            software engineer        New Delhi      m  15-04-1987 00:00
3994             technical writer        Hyderabad      f  27-08-1992 00:00
3995  associate software engineer        Bangalore      m  03-07-1991 00:00
3996           software developer  Asifabadbanglore      f  20-03-1992 00:00
3997      senior systems engineer          Chennai      f  26-02-1991 00:00

      10percentage  … ComputerScience  MechanicalEngg  ElectricalEngg  \
3993         52.09  …              -1              -1              -1
3994         90.00  …              -1              -1              -1
3995         81.86  …              -1              -1              -1
3996         78.72  …             438              -1              -1
3997         70.60  …              -1              -1              -1

     TelecomEngg  CivilEngg  conscientiousness agreeableness extraversion  \
3993          -1         -1            -0.1082        0.3448       0.2366
3994          -1         -1            -0.3027        0.8784       0.9322
3995          -1         -1            -1.5765       -1.5273      -1.5051
3996          -1         -1            -0.1590        0.0459      -0.4511
3997          -1         -1            -1.1128       -0.2793      -0.6343

      nueroticism  openess_to_experience
3993      0.64980                -0.9194
3994      0.77980                -0.0943
3995     -1.31840                -0.7615
3996     -0.36120                -0.0943
3997      1.32553                -0.6035

[5 rows x 39 columns]
```

[7]:
```
shape
```

[7]: (3998, 39)

[8]:
```
description
```

[8]:
```
                 ID        Salary  10percentage  12graduation  12percentage  \
count  3.998000e+03  3.998000e+03   3998.000000   3998.000000   3998.000000
mean   6.637945e+05  3.076998e+05     77.925443   2008.087544     74.466366
std    3.632182e+05  2.127375e+05      9.850162      1.653599     10.999933
min    1.124400e+04  3.500000e+04     43.000000   1995.000000     40.000000
25%    3.342842e+05  1.800000e+05     71.680000   2007.000000     66.000000
50%    6.396000e+05  3.000000e+05     79.150000   2008.000000     74.400000
75%    9.904800e+05  3.700000e+05     85.670000   2009.000000     82.600000
max    1.298275e+06  4.000000e+06     97.760000   2013.000000     98.700000

          CollegeID  CollegeTier   collegeGPA  CollegeCityID  CollegeCityTier  \
count   3998.000000  3998.000000  3998.000000    3998.000000      3998.000000
mean    5156.851426     1.925713    71.486171    5156.851426         0.300400
std     4802.261482     0.262270     8.167338    4802.261482         0.458489
min        2.000000     1.000000     6.450000       2.000000         0.000000
25%      494.000000     2.000000    66.407500     494.000000         0.000000
50%     3879.000000     2.000000    71.720000    3879.000000         0.000000
75%     8818.000000     2.000000    76.327500    8818.000000         1.000000
max    18409.000000     2.000000    99.930000   18409.000000         1.000000

       …  ComputerScience  MechanicalEngg  ElectricalEngg  TelecomEngg  \
count  …      3998.000000     3998.000000     3998.000000  3998.000000
mean   …        90.742371       22.974737       16.478739    31.851176
std    …       175.273083       98.123311       87.585634   104.852845
min    …        -1.000000       -1.000000       -1.000000    -1.000000
25%    …        -1.000000       -1.000000       -1.000000    -1.000000
50%    …        -1.000000       -1.000000       -1.000000    -1.000000
75%    …        -1.000000       -1.000000       -1.000000    -1.000000
max    …       715.000000      623.000000      676.000000   548.000000

         CivilEngg  conscientiousness  agreeableness  extraversion  \
count  3998.000000        3998.000000    3998.000000   3998.000000
mean      2.683842          -0.037831       0.146496      0.002763
std      36.658505           1.028666       0.941782      0.951471
min      -1.000000          -4.126700      -5.781600     -4.600900
25%      -1.000000          -0.713525      -0.287100     -0.604800
50%      -1.000000           0.046400       0.212400      0.091400
75%      -1.000000           0.702700       0.812800      0.672000
max     516.000000           1.995300       1.904800      2.535400

       nueroticism  openess_to_experience
```

```
count  3998.000000            3998.000000
mean     -0.169033              -0.138110
std       1.007580               1.008075
min      -2.643000              -7.375700
25%      -0.868200              -0.669200
50%      -0.234400              -0.094300
75%       0.526200               0.502400
max       3.352500               1.822400

[8 rows x 27 columns]
```

0.3 Print all the existing column names, one below another

[12]:
```python
for col in df.columns:
    print(col)
```

```
Unnamed: 0
ID
Salary
DOJ
DOL
Designation
JobCity
Gender
DOB
10percentage
10board
12graduation
12percentage
12board
CollegeID
CollegeTier
Degree
Specialization
collegeGPA
CollegeCityID
CollegeCityTier
CollegeState
GraduationYear
English
Logical
Quant
Domain
ComputerProgramming
ElectronicsAndSemicon
ComputerScience
MechanicalEngg
```

```
ElectricalEngg
TelecomEngg
CivilEngg
conscientiousness
agreeableness
extraversion
nueroticism
openess_to_experience
```

0.4 Rename the columns

[13]:
```python
df.columns = df.columns.str.replace('nueroticism','neuroticism')
df.columns = df.columns.str.replace('collegeGPA','CollegeGPA')
```

[15]:
```python
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
print(df.columns)
```

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

[16]:
```python
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   float64
 2   DOJ                    3998 non-null   object
 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   object
 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  neuroticism            3998 non-null   float64
 37  openess_to_experience  3998 non-null   float64
dtypes: float64(10), int64(17), object(11)
memory usage: 1.2+ MB
```

[17]:
```
df.isnull().sum()
```

[17]:
```
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
neuroticism              0
openess_to_experience    0
dtype: int64
```

[18]:
```
df.duplicated().sum()
```

[18]:
```
0
```

[19]:
```
df.nunique()
```

[19]:
```
ID                       3998
Salary                    177
DOJ                        81
DOL                        67
Designation               419
JobCity                   339
Gender                      2
DOB                      1872
10percentage              851
10board                   275
12graduation               16
12percentage              801
12board                   340
CollegeID                1350
CollegeTier                 2
Degree                      4
Specialization             46
CollegeGPA               1282
```

```
CollegeCityID            1350
CollegeCityTier             2
CollegeState               26
GraduationYear             11
English                   111
Logical                   107
Quant                     138
Domain                    243
ComputerProgramming        79
ElectronicsAndSemicon      29
ComputerScience            20
MechanicalEngg             42
ElectricalEngg             31
TelecomEngg                26
CivilEngg                  23
conscientiousness         141
agreeableness             149
extraversion              154
neuroticism               217
openess_to_experience     142
dtype: int64
```

[22]:
```
# Replace 'present' with a specific date
df['DOL'].replace('present', '2015-12-31', inplace=True)

# Convert DOL and DOJ columns to datetime format, handling inconsistent formats
df['DOL'] = pd.to_datetime(df['DOL'], errors='coerce', dayfirst=True)
df['DOJ'] = pd.to_datetime(df['DOJ'], errors='coerce', dayfirst=True)

# Display the first few rows to verify
df.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\4020330569.py:5: UserWarning:
Parsing dates in %Y-%m-%d format when dayfirst=True was specified. Pass
`dayfirst=False` or specify a format to silence this warning.
  df['DOL'] = pd.to_datetime(df['DOL'], errors='coerce', dayfirst=True)
```

[22]:
```
       ID     Salary        DOJ        DOL               Designation  \
0  203097   420000.0 2012-06-01 2015-12-31   senior quality engineer
1  579905   500000.0 2013-09-01 2015-12-31         assistant manager
2  810601   325000.0 2014-06-01 2015-12-31          systems engineer
3  267447  1100000.0 2011-07-01 2015-12-31  senior software engineer
4  343523   200000.0 2014-03-01        NaT                       get

     JobCity Gender               DOB  10percentage  \
0  Bangalore      f  19-02-1990 00:00          84.3
1     Indore      m  04-10-1989 00:00          85.4
```

```
2    Chennai      f  03-08-1992 00:00          85.0
3    Gurgaon      m  05-12-1989 00:00          85.6
4    Manesar      m  27-02-1991 00:00          78.0

                         10board  …  ComputerScience  MechanicalEngg  \
0  board ofsecondary education,ap  …               -1              -1
1                            cbse  …               -1              -1
2                            cbse  …               -1              -1
3                            cbse  …               -1              -1
4                            cbse  …               -1              -1

  ElectricalEngg  TelecomEngg  CivilEngg conscientiousness agreeableness  \
0             -1           -1         -1            0.9737        0.8128
1             -1           -1         -1           -0.7335        0.3789
2             -1           -1         -1            0.2718        1.7109
3             -1           -1         -1            0.0464        0.3448
4             -1           -1         -1           -0.8810       -0.2793

   extraversion  neuroticism  openess_to_experience
0        0.5269      1.35490                -0.4455
1        1.2396     -0.10760                 0.8637
2        0.1637     -0.86820                 0.6721
3       -0.3440     -0.40780                -0.9194
4       -1.0697      0.09163                -0.1295

[5 rows x 38 columns]
```

[23]:
```
categorical = ['Designation',␣
↪'JobCity','Gender','10board','12board','CollegeTier','Degree','Specialization','CollegeCity'
for cat in categorical:
    df[cat] = df[cat].astype('category')
```

[24]:
```
df.dtypes
```

[24]:
```
ID                                int64
Salary                          float64
DOJ                      datetime64[ns]
DOL                      datetime64[ns]
Designation                    category
JobCity                        category
Gender                         category
DOB                              object
10percentage                    float64
10board                        category
12graduation                      int64
12percentage                    float64
12board                        category
```

```
CollegeID                          int64
CollegeTier                     category
Degree                          category
Specialization                  category
CollegeGPA                       float64
CollegeCityID                      int64
CollegeCityTier                 category
CollegeState                    category
GraduationYear                     int64
English                            int64
Logical                            int64
Quant                              int64
Domain                           float64
ComputerProgramming                int64
ElectronicsAndSemicon              int64
ComputerScience                    int64
MechanicalEngg                     int64
ElectricalEngg                     int64
TelecomEngg                        int64
CivilEngg                          int64
conscientiousness                float64
agreeableness                    float64
extraversion                     float64
neuroticism                      float64
openess_to_experience            float64
dtype: object
```

[26]:
```
dates = df[(df['DOL'] < df['DOJ'])].shape[0]
print(f'DOL is earlier than DOJ for {dates} observations.')
print(df.shape)
```

```
DOL is earlier than DOJ for 0 observations.
(3998, 38)
```

[28]:
```
df = df.drop(df[~(df['DOL'] > df['DOJ'])].index)
print(df.shape)
```

```
(1875, 38)
```

[29]:
```
df['Gender'].replace({'f':'Female','m':'Male'}, inplace = True)
df.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2573337485.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Gender'].replace({'f':'Female','m':'Male'}, inplace = True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2573337485.py:1: FutureWarning:
The behavior of Series.replace (and DataFrame.replace) with CategoricalDtype is
deprecated. In a future version, replace will only be used for cases that
preserve the categories. To change the categories, use ser.cat.rename_categories
instead.
  df['Gender'].replace({'f':'Female','m':'Male'}, inplace = True)
```

[29]:

```
        ID     Salary        DOJ        DOL              Designation  \
0   203097   420000.0 2012-06-01 2015-12-31   senior quality engineer
1   579905   500000.0 2013-09-01 2015-12-31         assistant manager
2   810601   325000.0 2014-06-01 2015-12-31          systems engineer
3   267447  1100000.0 2011-07-01 2015-12-31  senior software engineer
5  1027655   300000.0 2014-06-01 2015-12-31           system engineer

     JobCity  Gender               DOB  10percentage  \
0  Bangalore  Female  19-02-1990 00:00         84.30
1     Indore    Male  04-10-1989 00:00         85.40
2    Chennai  Female  03-08-1992 00:00         85.00
3    Gurgaon    Male  05-12-1989 00:00         85.60
5  Hyderabad    Male  02-07-1992 00:00         89.92

                         10board  …  ComputerScience  MechanicalEngg  \
0  board ofsecondary education,ap  …               -1              -1
1                            cbse  …               -1              -1
2                            cbse  …               -1              -1
3                            cbse  …               -1              -1
5                     state board  …              407              -1

  ElectricalEngg  TelecomEngg CivilEngg conscientiousness agreeableness  \
0             -1           -1        -1            0.9737        0.8128
1             -1           -1        -1           -0.7335        0.3789
2             -1           -1        -1            0.2718        1.7109
3             -1           -1        -1            0.0464        0.3448
5             -1           -1        -1           -0.3027       -0.6201

   extraversion  neuroticism openess_to_experience
0        0.5269       1.3549               -0.4455
1        1.2396      -0.1076                0.8637
2        0.1637      -0.8682                0.6721
3       -0.3440      -0.4078               -0.9194
```

```
5      -2.2954      -0.7415                -0.8608

[5 rows x 38 columns]
```

[31]:
```python
print((df['10percentage'] <=10).sum())
print((df['12percentage'] <=10).sum())
print((df['CollegeGPA'] <=10).sum())
```

```
0
0
1
```

[34]:
```python
df.loc[df['CollegeGPA']<=10,'CollegeGPA'].index
```

[34]:
```
Index([1439], dtype='int64')
```

[38]:
```python
df.loc[df['CollegeGPA']<=10,'CollegeGPA'] = (df.
 ↪loc[df['CollegeGPA']<=10,'CollegeGPA']/10)*100
df.head()
```

[38]:
```
        ID     Salary        DOJ        DOL              Designation  \
0   203097   420000.0 2012-06-01 2015-12-31  senior quality engineer
1   579905   500000.0 2013-09-01 2015-12-31        assistant manager
2   810601   325000.0 2014-06-01 2015-12-31         systems engineer
3   267447  1100000.0 2011-07-01 2015-12-31 senior software engineer
5  1027655   300000.0 2014-06-01 2015-12-31          system engineer

     JobCity  Gender              DOB  10percentage  \
0  Bangalore  Female  19-02-1990 00:00         84.30
1     Indore    Male  04-10-1989 00:00         85.40
2    Chennai  Female  03-08-1992 00:00         85.00
3    Gurgaon    Male  05-12-1989 00:00         85.60
5  Hyderabad    Male  02-07-1992 00:00         89.92

                      10board  …  ComputerScience  MechanicalEngg  \
0  board ofsecondary education,ap  …              -1              -1
1                        cbse  …              -1              -1
2                        cbse  …              -1              -1
3                        cbse  …              -1              -1
5                 state board  …             407              -1

  ElectricalEngg  TelecomEngg CivilEngg conscientiousness agreeableness  \
0             -1           -1        -1            0.9737        0.8128
1             -1           -1        -1           -0.7335        0.3789
2             -1           -1        -1            0.2718        1.7109
3             -1           -1        -1            0.0464        0.3448
5             -1           -1        -1           -0.3027       -0.6201
```

```
   extraversion  neuroticism openess_to_experience
0        0.5269       1.3549                -0.4455
1        1.2396      -0.1076                 0.8637
2        0.1637      -0.8682                 0.6721
3       -0.3440      -0.4078                -0.9194
5       -2.2954      -0.7415                -0.8608

[5 rows x 38 columns]
```

[39]:
```
print((df==0).sum()[(df==0).sum() > 0])
```

```
CollegeCityTier    1308
dtype: int64
```

[40]:
```
(df==-1).sum()[(df==-1).sum()>0]/len(df)*100
```

[40]:
```
Domain                    5.813333
ComputerProgramming      22.560000
ElectronicsAndSemicon    71.413333
ComputerScience          75.626667
MechanicalEngg           94.293333
ElectricalEngg           96.106667
TelecomEngg              89.973333
CivilEngg                98.826667
dtype: float64
```

[41]:
```
df = df.drop(columns = ['MechanicalEngg', 'ElectricalEngg',␣
 ↪'TelecomEngg','CivilEngg'])
df.head()
```

[41]:
```
        ID     Salary         DOJ         DOL               Designation  \
0   203097   420000.0 2012-06-01 2015-12-31   senior quality engineer
1   579905   500000.0 2013-09-01 2015-12-31         assistant manager
2   810601   325000.0 2014-06-01 2015-12-31          systems engineer
3   267447  1100000.0 2011-07-01 2015-12-31  senior software engineer
5  1027655   300000.0 2014-06-01 2015-12-31           system engineer

     JobCity  Gender               DOB  10percentage  \
0  Bangalore  Female  19-02-1990 00:00         84.30
1     Indore    Male  04-10-1989 00:00         85.40
2    Chennai  Female  03-08-1992 00:00         85.00
3    Gurgaon    Male  05-12-1989 00:00         85.60
5  Hyderabad    Male  02-07-1992 00:00         89.92

                          10board  …  Quant    Domain ComputerProgramming  \
0  board ofsecondary education,ap  …    525  0.635979                 445
```

```
1                              cbse  ...    780  0.960603                    -1
2                              cbse  ...    370  0.450877                   395
3                              cbse  ...    625  0.974396                   615
5                       state board  ...    620 -1.000000                   645

   ElectronicsAndSemicon ComputerScience conscientiousness agreeableness  \
0                     -1              -1            0.9737        0.8128
1                    466              -1           -0.7335        0.3789
2                     -1              -1            0.2718        1.7109
3                     -1              -1            0.0464        0.3448
5                     -1             407           -0.3027       -0.6201

   extraversion  neuroticism openess_to_experience
0        0.5269       1.3549               -0.4455
1        1.2396      -0.1076                0.8637
2        0.1637      -0.8682                0.6721
3       -0.3440      -0.4078               -0.9194
5       -2.2954      -0.7415               -0.8608

[5 rows x 34 columns]
```

[42]:
```
df['10board'] = df['10board'].astype(str)
df['12board'] = df['12board'].astype(str)
df['JobCity'] = df['JobCity'].astype(str)
```

[44]:
```
df['10board'] = df['10board'].replace({'0':np.nan})
df['12board'] = df['12board'].replace({'0':np.nan})
df['GraduationYear'] = df['GraduationYear'].replace({0:np.nan})
df['JobCity'] = df['JobCity'].replace({'-1':np.nan})
df['Domain'] = df['Domain'].replace({-1:np.nan})
df['ElectronicsAndSemicon'] = df['ElectronicsAndSemicon'].replace({-1:0})
df['ComputerScience'] = df['ComputerScience'].replace({-1:0})
df['ComputerProgramming'] = df['ComputerProgramming'].replace({-1:np.nan})
```

[45]:
```
df['10board'] = df['10board'].astype('category')
df['12board'] = df['12board'].astype('category')
df['JobCity'] = df['JobCity'].astype('category')
```

[46]:
```
df['10board'].fillna(df['10board'].mode()[0], inplace = True)
df['12board'].fillna(df['12board'].mode()[0], inplace = True)
df['GraduationYear'].fillna(df['GraduationYear'].mode()[0], inplace = True)
df['JobCity'].fillna(df['JobCity'].mode()[0], inplace = True)
df
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2406600982.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['10board'].fillna(df['10board'].mode()[0], inplace = True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2406600982.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['12board'].fillna(df['12board'].mode()[0], inplace = True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2406600982.py:3: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['GraduationYear'].fillna(df['GraduationYear'].mode()[0], inplace = True)
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2406600982.py:4: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

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

[46]:

```
           ID     Salary         DOJ         DOL                  Designation  \
0      203097   420000.0  2012-06-01  2015-12-31      senior quality engineer
1      579905   500000.0  2013-09-01  2015-12-31            assistant manager
2      810601   325000.0  2014-06-01  2015-12-31             systems engineer
3      267447  1100000.0  2011-07-01  2015-12-31     senior software engineer
5     1027655   300000.0  2014-06-01  2015-12-31              system engineer
…         …         …          …          …                          …
3987   439787   280000.0  2012-11-01  2015-12-31             network engineer
3989  1204604   300000.0  2014-09-01  2015-12-31            software engineer
3990   204287   480000.0  2012-02-01  2015-12-31      senior systems engineer
3995   355888   320000.0  2013-07-01  2015-12-31  associate software engineer
3997   324966   400000.0  2013-02-01  2015-12-31      senior systems engineer

        JobCity  Gender               DOB  10percentage  \
0     Bangalore  Female  19-02-1990 00:00         84.30
1        Indore    Male  04-10-1989 00:00         85.40
2       Chennai  Female  03-08-1992 00:00         85.00
3       Gurgaon    Male  05-12-1989 00:00         85.60
5     Hyderabad    Male  02-07-1992 00:00         89.92
…           …      …                …           …
3987  New Delhi  Female  16-01-1990 00:00         86.70
3989  Bangalore    Male  23-11-1991 00:00         74.88
3990  Hyderabad  Female  04-09-1989 00:00         88.00
3995  Bangalore    Male  03-07-1991 00:00         81.86
3997    Chennai  Female  26-02-1991 00:00         70.60

                             10board  …  Quant    Domain  \
0     board ofsecondary education,ap  …    525  0.635979
1                               cbse  …    780  0.960603
2                               cbse  …    370  0.450877
3                               cbse  …    625  0.974396
5                        state board  …    620       NaN
…                                …   …     …         …
3987                      cbse board  …    485  0.376060
3989                     state board  …    500  0.356536
3990                            cbse  …    605  0.824666
3995                      bse,odisha  …    465  0.488348
3997                            cbse  …    464  0.600057

     ComputerProgramming  ElectronicsAndSemicon ComputerScience  \
0                  445.0                      0               0
1                    NaN                    466               0
2                  395.0                      0               0
3                  615.0                      0               0
5                  645.0                      0             407
```

```
…                   …                    …            …
3987                 NaN                 300            0
3989               465.0                   0          346
3990               285.0                 400            0
3995               405.0                   0            0
3997               435.0                   0            0

     conscientiousness agreeableness  extraversion  neuroticism  \
0               0.9737        0.8128        0.5269      1.35490
1              -0.7335        0.3789        1.2396     -0.10760
2               0.2718        1.7109        0.1637     -0.86820
3               0.0464        0.3448       -0.3440     -0.40780
5              -0.3027       -0.6201       -2.2954     -0.74150
…                  …            …            …           …
3987           -1.4992       -1.8393       -0.7794      1.47240
3989            0.1282        0.0459        1.2396      1.03330
3990            0.6646        0.3448        0.3817     -1.34780
3995           -1.5765       -1.5273       -1.5051     -1.31840
3997           -1.1128       -0.2793       -0.6343      1.32553

     openess_to_experience
0                  -0.4455
1                   0.8637
2                   0.6721
3                  -0.9194
5                  -0.8608
…                      …
3987               -2.3017
3989                0.6721
3990                0.8183
3995               -0.7615
3997               -0.6035

[1875 rows x 34 columns]
```

[48]:
```python
def correct_string_data(data):
    '''
    Convert the textual categories to lower case
    and remove the leading or trailing spaces if any.
    '''
    df[data] = df[data].str.lower().str.strip()
```

[49]:
```python
textual_columns =␣
 ↪['Designation','JobCity','10board','12board','Specialization','CollegeState']
```

[51]:
```python
for col in textual_columns:
```

```
    print(f'Number of unique values in {col} with inconsistency : {df[col].
 ↪nunique()}')
```

```
Number of unique values in Designation with inconsistency : 276
Number of unique values in JobCity with inconsistency : 181
Number of unique values in 10board with inconsistency : 141
Number of unique values in 12board with inconsistency : 165
Number of unique values in Specialization with inconsistency : 36
Number of unique values in CollegeState with inconsistency : 25
```

[52]:
```
for col in textual_columns:
    correct_string_data(col)
```

[53]:
```
for col in textual_columns:
    print(f'Number of unique values in {col} without inconsistency : {df[col].
 ↪nunique()}')
```

```
Number of unique values in Designation without inconsistency : 276
Number of unique values in JobCity without inconsistency : 127
Number of unique values in 10board without inconsistency : 141
Number of unique values in 12board without inconsistency : 164
Number of unique values in Specialization without inconsistency : 36
Number of unique values in CollegeState without inconsistency : 25
```

[55]:
```
def collapsing_categories(df, data):
    for Designation in df[data].unique():
        min_count = df[data].value_counts()[:10].min()
        if df[df[data] == Designation][data].value_counts()[0] < min_count:
            df.loc[df[data] == Designation, data] = 'other'
```

[60]:
```
for cols in textual_columns:
    print('')
    print('Top 10 categories in:', cols)
    print('')
    print(df[cols].value_counts())
    print('')
    print('*'*100)
```

```
Top 10 categories in: Designation

Designation
other                         1010
software engineer              299
software developer             113
system engineer                111
programmer analyst              82
systems engineer                66
```

```
software test engineer        58
java software engineer        53
senior software engineer      46
project engineer              37
Name: count, dtype: int64

********************************************************************************
********************

Top 10 categories in: JobCity

JobCity
bangalore    538
other        323
noida        191
hyderabad    188
pune         175
chennai      152
gurgaon      111
new delhi     81
kolkata       61
mumbai        55
Name: count, dtype: int64

********************************************************************************
********************

Top 10 categories in: 10board

10board
cbse                           849
state board                    567
other                          186
icse                           138
ssc                             54
up board                        29
matriculation                   13
sslc                            10
rbse                             8
maharashtra state board          7
upboard                          7
board of secondary education     7
Name: count, dtype: int64

********************************************************************************
********************

Top 10 categories in: 12board
```

```
12board
cbse                              861
state board                       610
other                             240
icse                               65
up board                           31
board of intermediate education    20
isc                                17
board of intermediate              17
maharashtra state board             7
ipe                                 7
Name: count, dtype: int64

********************************************************************************
********************

Top 10 categories in: Specialization

Specialization
electronics and communication engineering    442
computer science & engineering               378
information technology                       298
computer engineering                         271
other                                        125
mechanical engineering                        97
electronics and electrical engineering        90
computer application                          89
electronics & telecommunications              51
electrical engineering                        34
Name: count, dtype: int64

********************************************************************************
********************

Top 10 categories in: CollegeState

CollegeState
other             449
uttar pradesh     413
karnataka         171
telangana         155
tamil nadu        152
andhra pradesh    124
maharashtra       118
west bengal       102
punjab            100
delhi              91
```

```
Name: count, dtype: int64

********************************************************************************
********************
```

[61]:

```
df
```

[61]:

```
           ID     Salary         DOJ         DOL               Designation  \
0      203097   420000.0  2012-06-01  2015-12-31                     other
1      579905   500000.0  2013-09-01  2015-12-31                     other
2      810601   325000.0  2014-06-01  2015-12-31          systems engineer
3      267447  1100000.0  2011-07-01  2015-12-31  senior software engineer
5     1027655   300000.0  2014-06-01  2015-12-31           system engineer
…         …         …          …           …                       …
3987   439787   280000.0  2012-11-01  2015-12-31                     other
3989  1204604   300000.0  2014-09-01  2015-12-31         software engineer
3990   204287   480000.0  2012-02-01  2015-12-31                     other
3995   355888   320000.0  2013-07-01  2015-12-31                     other
3997   324966   400000.0  2013-02-01  2015-12-31                     other

        JobCity  Gender               DOB  10percentage      10board  …  \
0     bangalore  Female  19-02-1990 00:00         84.30        other  …
1         other    Male  04-10-1989 00:00         85.40         cbse  …
2       chennai  Female  03-08-1992 00:00         85.00         cbse  …
3       gurgaon    Male  05-12-1989 00:00         85.60         cbse  …
5     hyderabad    Male  02-07-1992 00:00         89.92  state board  …
…           …       …                …            …            …  …
3987  new delhi  Female  16-01-1990 00:00         86.70        other  …
3989  bangalore    Male  23-11-1991 00:00         74.88  state board  …
3990  hyderabad  Female  04-09-1989 00:00         88.00         cbse  …
3995  bangalore    Male  03-07-1991 00:00         81.86        other  …
3997    chennai  Female  26-02-1991 00:00         70.60         cbse  …

      Quant    Domain ComputerProgramming  ElectronicsAndSemicon  \
0       525  0.635979               445.0                      0
1       780  0.960603                 NaN                    466
2       370  0.450877               395.0                      0
3       625  0.974396               615.0                      0
5       620       NaN               645.0                      0
…       …       …                   …                    …
3987    485  0.376060                 NaN                    300
3989    500  0.356536               465.0                      0
3990    605  0.824666               285.0                    400
3995    465  0.488348               405.0                      0
3997    464  0.600057               435.0                      0

     ComputerScience conscientiousness agreeableness  extraversion  \
```

```
0                 0        0.9737      0.8128      0.5269
1                 0       -0.7335      0.3789      1.2396
2                 0        0.2718      1.7109      0.1637
3                 0        0.0464      0.3448     -0.3440
5               407       -0.3027     -0.6201     -2.2954
…             …             …           …           …
3987              0       -1.4992     -1.8393     -0.7794
3989            346        0.1282      0.0459      1.2396
3990              0        0.6646      0.3448      0.3817
3995              0       -1.5765     -1.5273     -1.5051
3997              0       -1.1128     -0.2793     -0.6343

      neuroticism openess_to_experience
0         1.35490               -0.4455
1        -0.10760                0.8637
2        -0.86820                0.6721
3        -0.40780               -0.9194
5        -0.74150               -0.8608
…             …                   …
3987      1.47240               -2.3017
3989      1.03330                0.6721
3990     -1.34780                0.8183
3995     -1.31840               -0.7615
3997      1.32553               -0.6035

[1875 rows x 34 columns]
```

[62]:

```python
df['DOB'] = pd.to_datetime(df['DOB'])
df['Age'] = 2015 - df['DOB'].dt.year
df.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\378708360.py:1: UserWarning:
Parsing dates in %d-%m-%Y %H:%M format when dayfirst=False (the default) was
specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df['DOB'] = pd.to_datetime(df['DOB'])
```

[62]:

```
        ID     Salary        DOJ        DOL               Designation  \
0   203097   420000.0 2012-06-01 2015-12-31                     other
1   579905   500000.0 2013-09-01 2015-12-31                     other
2   810601   325000.0 2014-06-01 2015-12-31          systems engineer
3   267447  1100000.0 2011-07-01 2015-12-31  senior software engineer
5  1027655   300000.0 2014-06-01 2015-12-31           system engineer

     JobCity Gender        DOB  10percentage      10board  …    Domain  \
0  bangalore Female 1990-02-19         84.30        other  …  0.635979
1      other   Male 1989-10-04         85.40         cbse  …  0.960603
2    chennai Female 1992-08-03         85.00         cbse  …  0.450877
```

```
3    gurgaon    Male 1989-12-05         85.60          cbse  …  0.974396
5  hyderabad    Male 1992-07-02         89.92   state board  …       NaN

   ComputerProgramming ElectronicsAndSemicon  ComputerScience  \
0                445.0                     0                0
1                  NaN                   466                0
2                395.0                     0                0
3                615.0                     0                0
5                645.0                     0              407

  conscientiousness agreeableness extraversion  neuroticism  \
0            0.9737        0.8128       0.5269       1.3549
1           -0.7335        0.3789       1.2396      -0.1076
2            0.2718        1.7109       0.1637      -0.8682
3            0.0464        0.3448      -0.3440      -0.4078
5           -0.3027       -0.6201      -2.2954      -0.7415

   openess_to_experience Age
0                -0.4455  25
1                 0.8637  26
2                 0.6721  23
3                -0.9194  26
5                -0.8608  23

[5 rows x 35 columns]
```

[64]:
```python
delta = (df['DOL'] - df['DOJ'])
tenure = np.zeros(len(df))
for i, date in enumerate(delta):
    tenure[i] = round(date.days/365,2)
df['Tenure'] = tenure
```

[65]:
```python
len(df[(df['GraduationYear'] > df['DOJ'].dt.year)].index)
```

[65]: 23

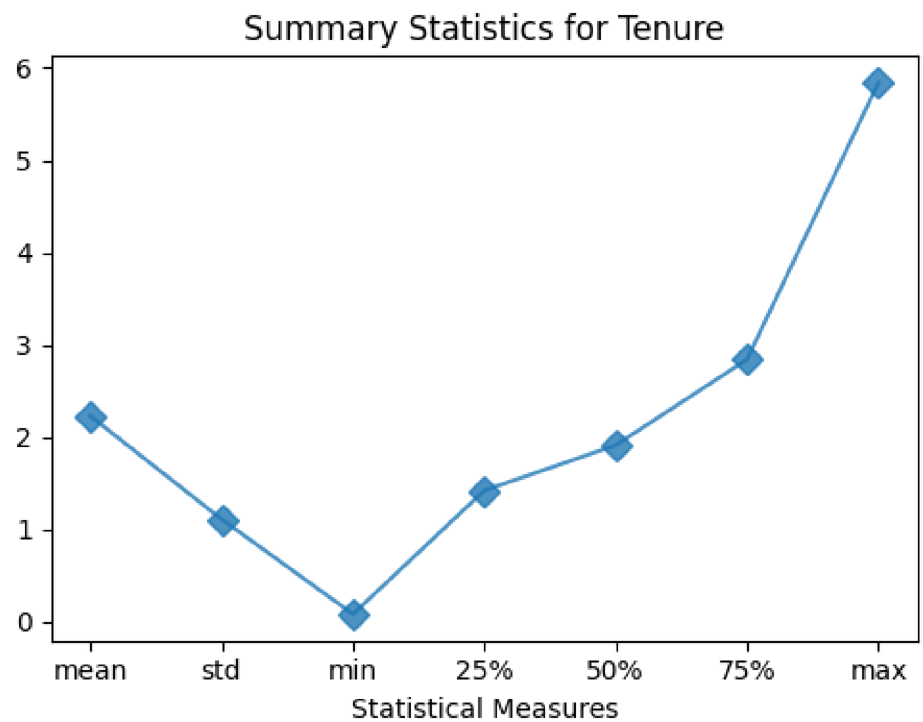
[66]:
```python
df = df.drop(df[(df['GraduationYear'] > df['DOJ'].dt.year)].index)
```

[67]:
```python
def cdf(data):
    x = np.sort(data)
    y = np.arange(1, len(x)+1)/len(x)
    return x, y
```

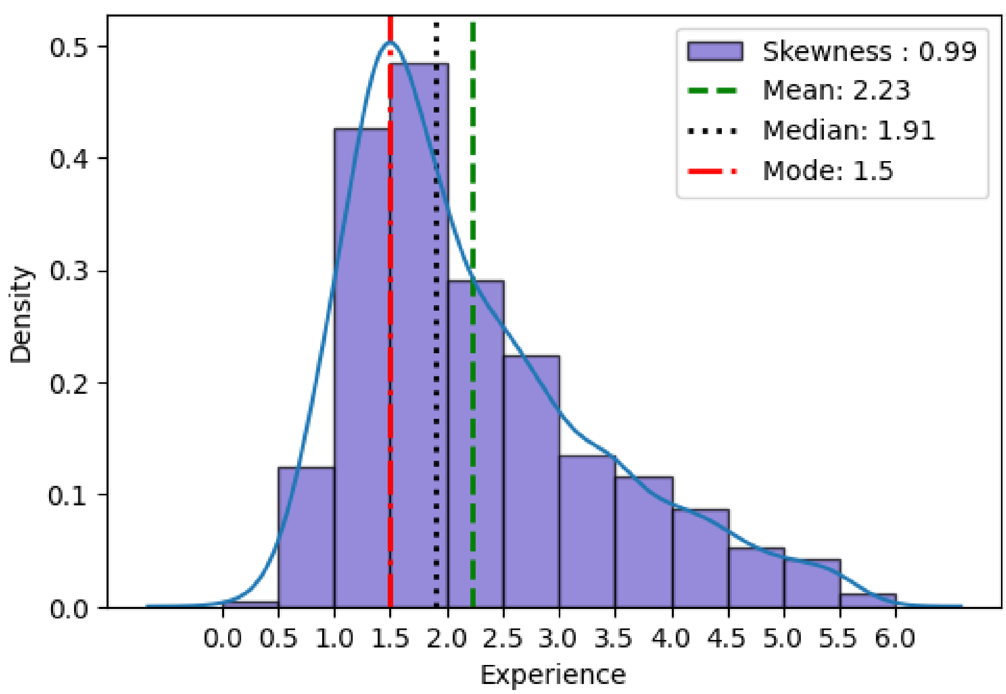
1 Univariate Analysis

[73]:
```
plt.figure(figsize=(5, 4))
df['Tenure'].describe()[1:].plot(alpha = 0.8,marker = 'D', markersize = 8)
plt.title('Summary Statistics for Tenure')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
plt.figure(figsize = (6,4))
plt.hist(df['Tenure'],ec = 'k',bins = np.arange(0, df['Tenure'].max()+0.5, 0.5),color = 'slateblue',alpha = 0.7,label = f"Skewness : {round(df['Tenure'].skew(),2)}",density = True)
plt.xticks(ticks = np.arange(0, df['Tenure'].max()+0.5, 0.5))
plt.xlabel('Experience')
plt.ylabel('Density')
plt.axvline(df['Tenure'].mean(), label = f"Mean: {round(df['Tenure'].mean(),2)}",linestyle = '--',color = 'green', linewidth = 2)
plt.axvline(df['Tenure'].median(), label = f"Median: {round(df['Tenure'].median(),2)}",linestyle = ':',color = 'k', linewidth = 2)
plt.axvline(df['Tenure'].mode()[0], label = f"Mode: {round(df['Tenure'].mode()[0],2)}", linestyle = '-.',color = 'red', linewidth = 2)
sns.kdeplot(df['Tenure'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5, 4))
sns.boxplot(df['Tenure'])
plt.xlabel('Tenure')
plt.tight_layout()
plt.show()

#CDF
plt.figure(figsize=(5, 4))
x_tenure, y_tenure = cdf(df['Tenure'])
x_sample_tenure, y_sample_tenure = cdf(np.random.normal(df['Tenure'].mean(),df['Tenure'].std(), size = len(df['Tenure'])))
plt.plot(x_tenure, y_tenure, linestyle = 'None',marker = '.', color = 'green',alpha = 0.7, label = 'Tenure')
plt.plot(x_sample_tenure, y_sample_tenure, linestyle = 'None',marker ='.', color = 'red',alpha = 0.7, label = 'Normal Distribution')
plt.xlabel('Tenure')
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
```

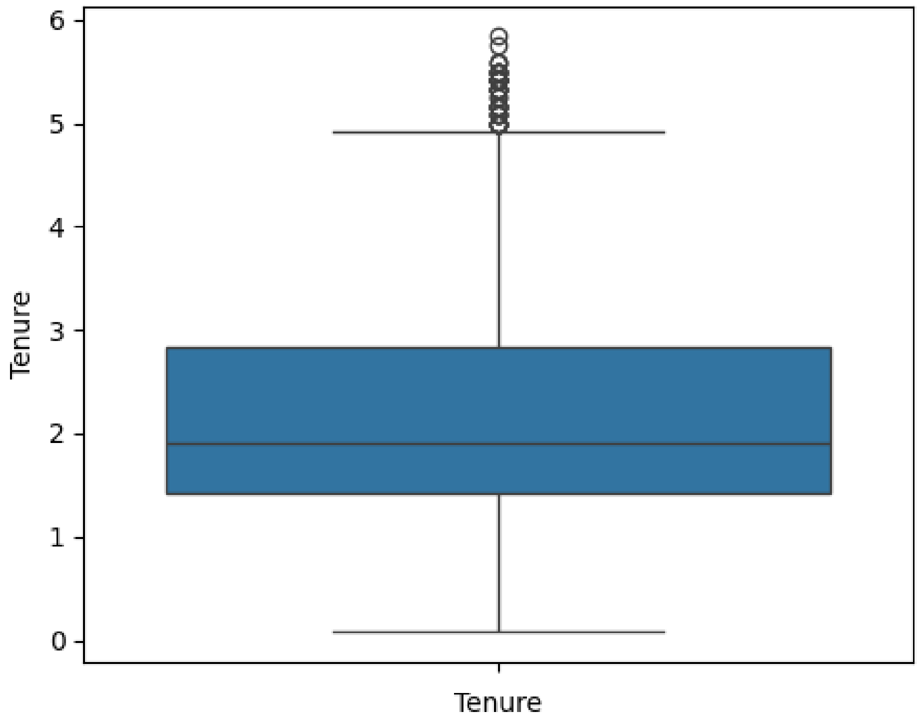
```
plt.show()
```



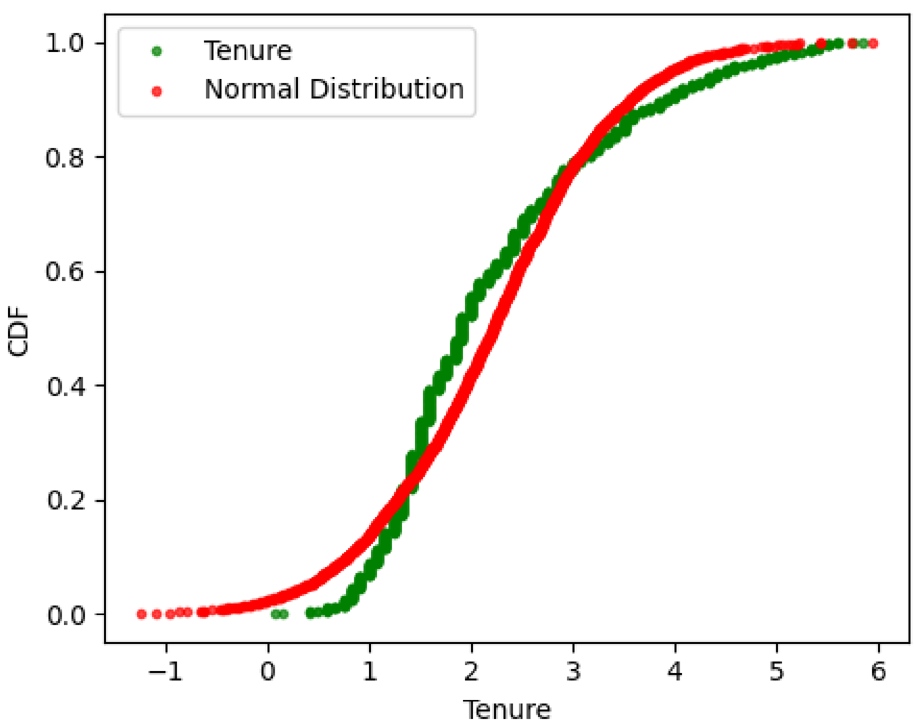
Summary Statistics for Tenure
6
5
4
3
2
1
0
mean
std
min
25%
50%
75%
max
Statistical Measures



Skewness : 0.99
Mean: 2.23
Median: 1.91
Mode: 1.5
Density
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
Experience

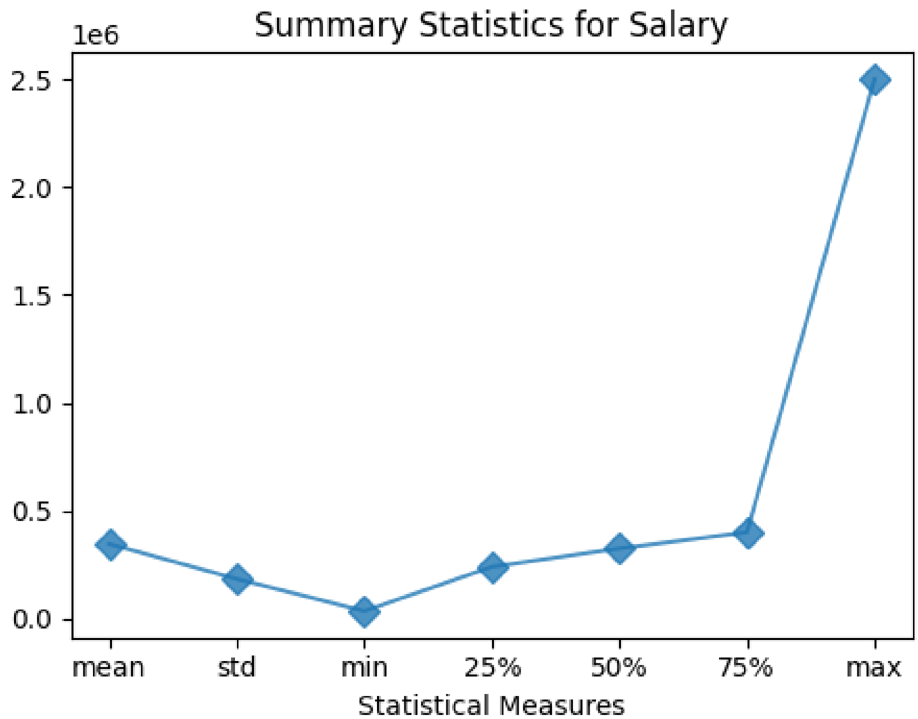


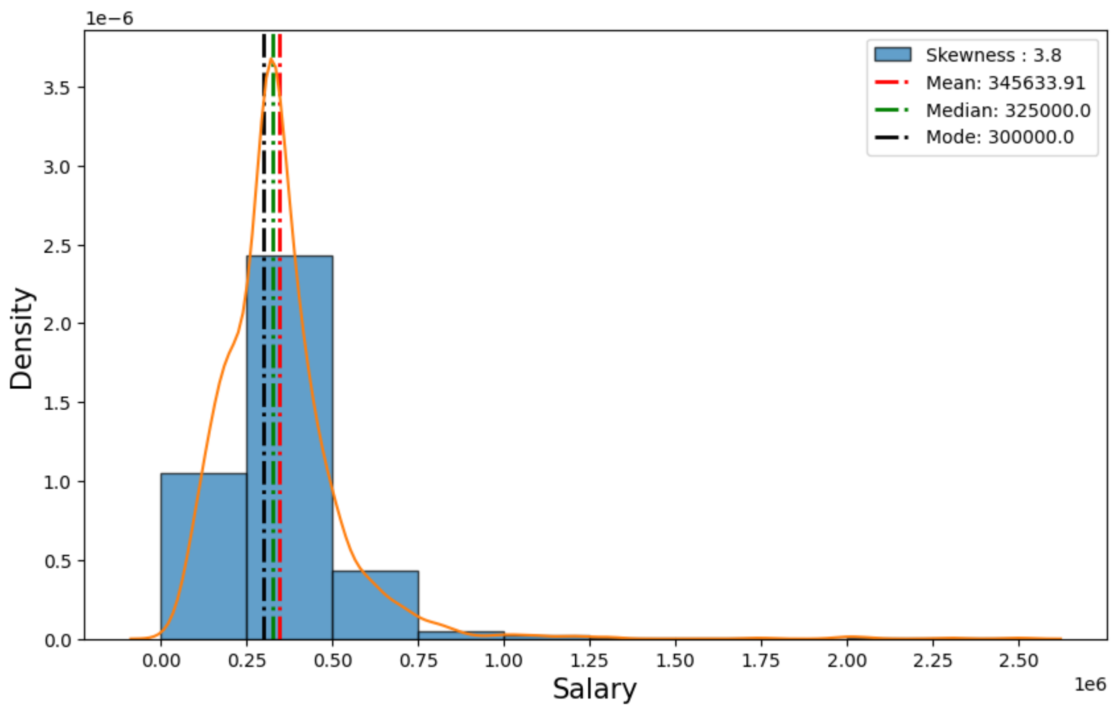
6
5
4
3
2
1
0
Tenure
Tenure

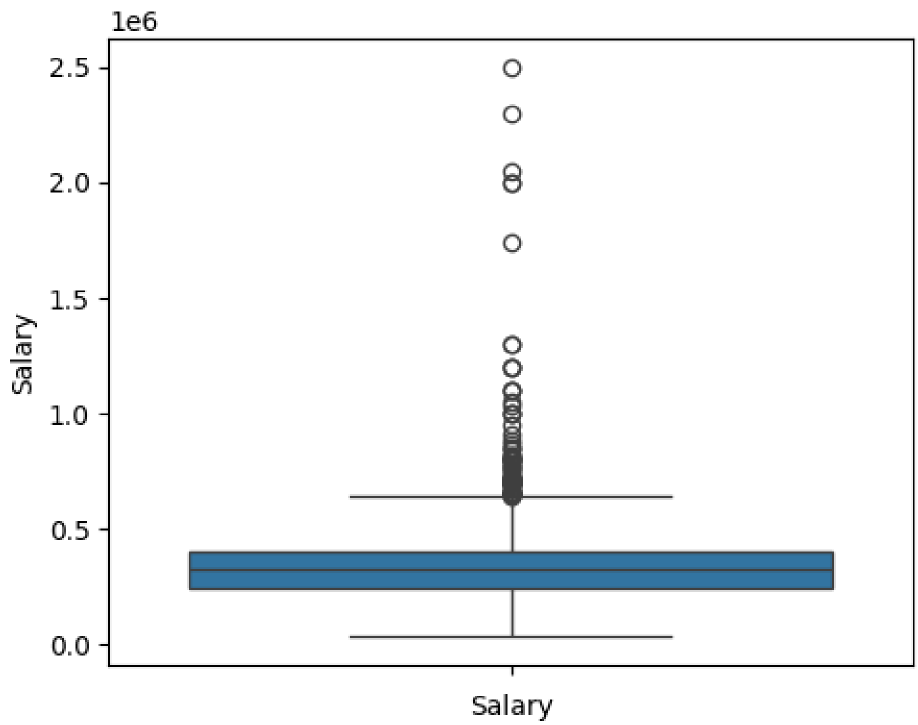


Tenure
Normal Distribution
1.0
0.8
0.6
0.4
0.2
0.0
CDF
−1
0
1
2
3
4
5
6
Tenure

[75]:
```
# Summary Plot
plt.figure(figsize=(5,4))
df['Salary'].describe()[1:].plot(alpha = 0.8,marker = 'D', markersize = 8)
plt.title('Summary Statistics for Salary')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(0, df['Salary'].max()+250000, 250000)
plt.figure(figsize = (10,6))
plt.hist(df['Salary'], ec = 'k',bins = bins,label = f"Skewness :␣
↪{round(df['Salary'].skew(),2)}",alpha = 0.7,density = True)
plt.xticks(bins)
plt.xlabel('Salary', size = 15)
plt.ylabel('Density', size = 15)
plt.axvline(df['Salary'].mean(), label = f"Mean: {round(df['Salary'].
↪mean(),2)}", linestyle = '-.',color = 'red', linewidth = 2)
plt.axvline(df['Salary'].median(), label = f"Median: {round(df['Salary'].
↪median(),2)}", linestyle = '-.',color = 'green', linewidth = 2)
plt.axvline(df['Salary'].mode()[0], label = f"Mode: {round(df['Salary'].
↪mode()[0],2)}", linestyle = '-.',color = 'k', linewidth = 2)
sns.kdeplot(df['Salary'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['Salary'])
plt.xlabel('Salary')
plt.tight_layout()
plt.show()
```

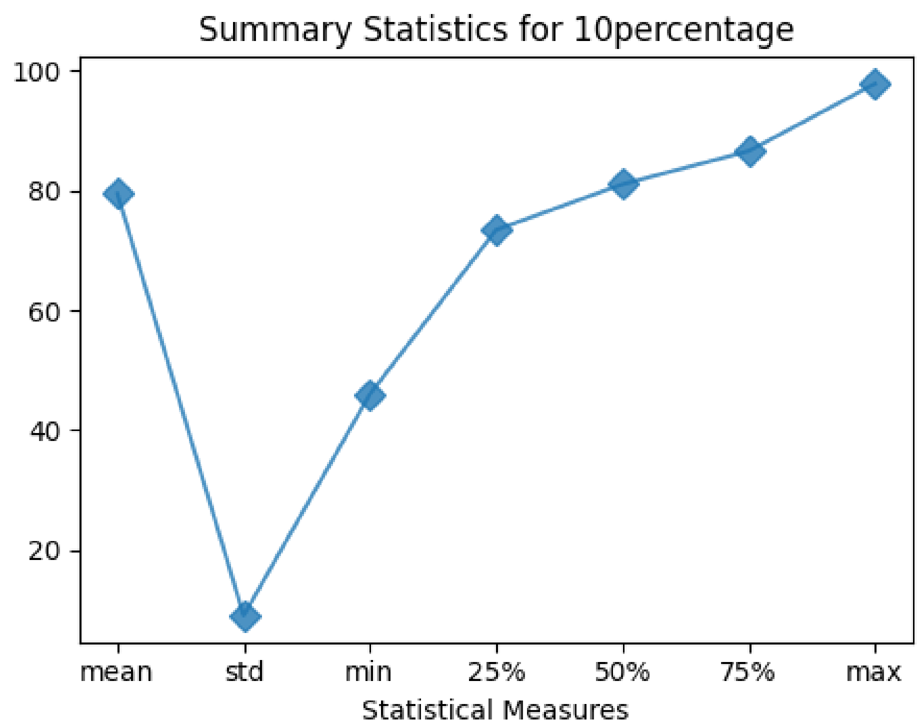

Summary Statistics for Salary
1e6
2.5
2.0
1.5
1.0
0.5
0.0
mean
std
min
25%
50%
75%
max
Statistical Measures

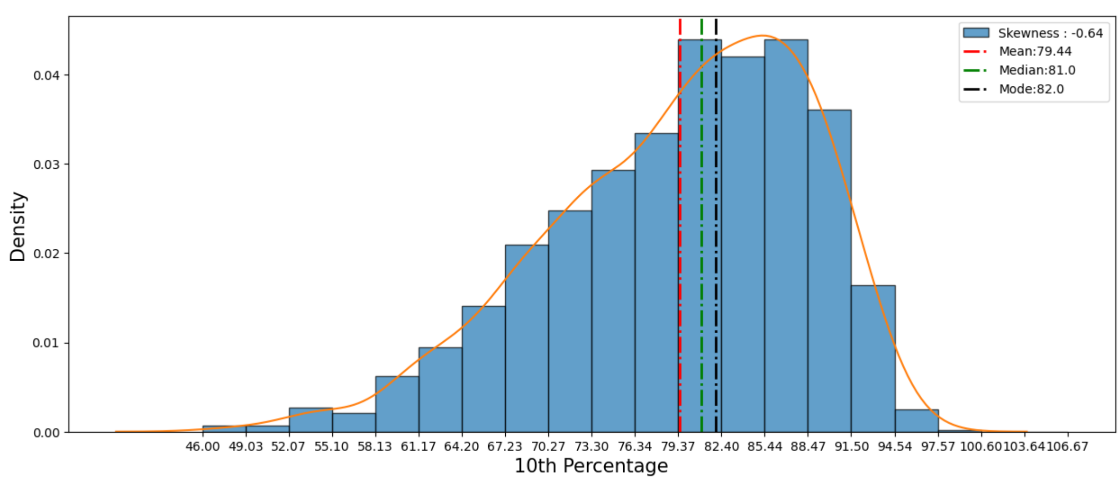

1e−6
Skewness : 3.8
Mean: 345633.91
Median: 325000.0
Mode: 300000.0
Density
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
1e6
Salary

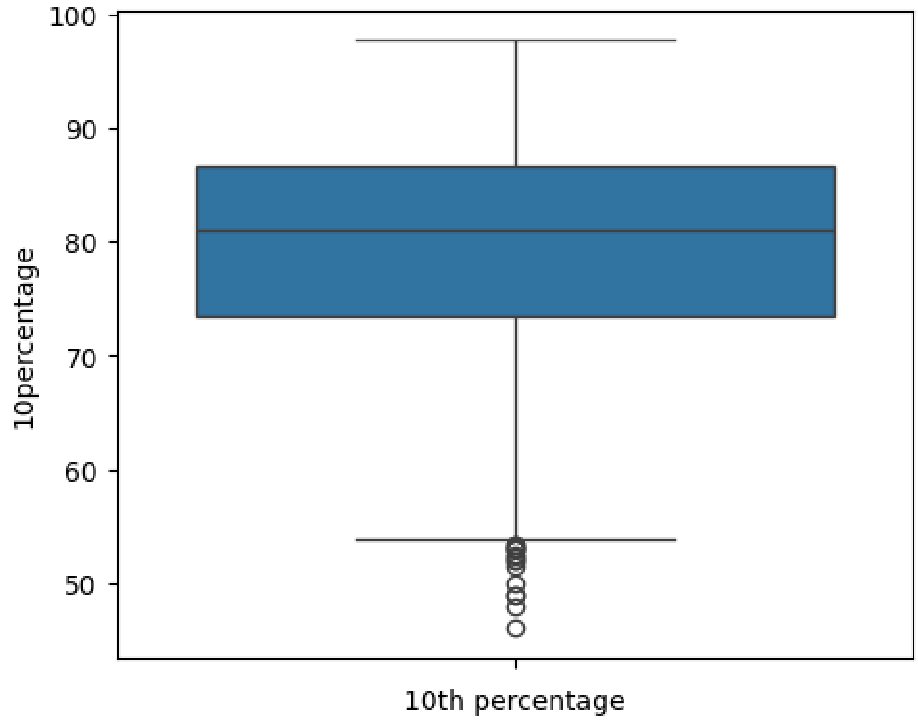


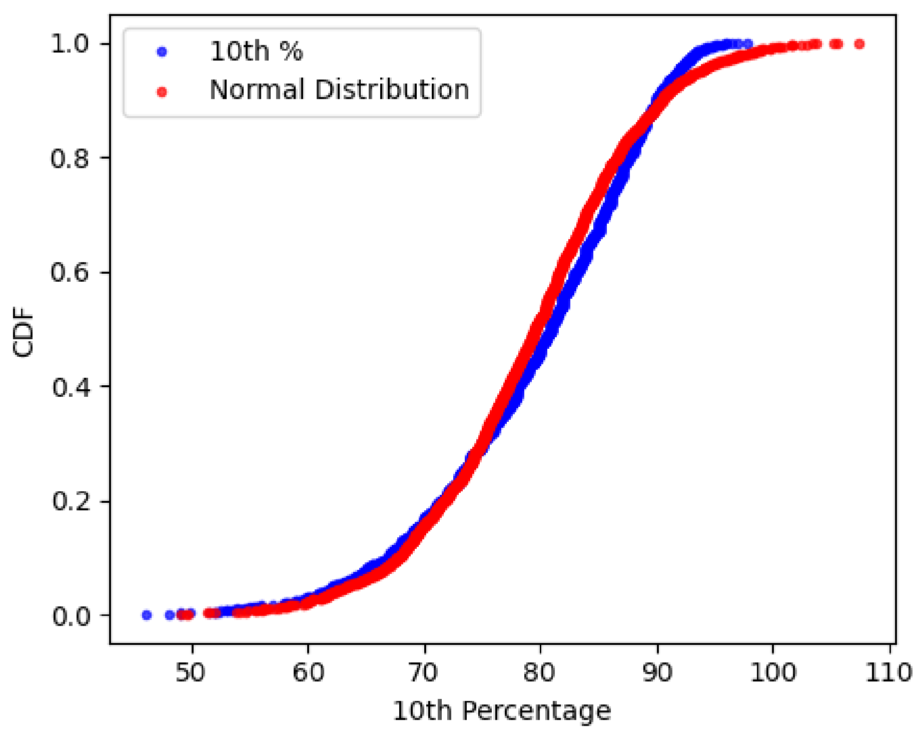
[82]:
```
# Summary Plot
plt.figure(figsize=(5,4))
df['10percentage'].describe()[1:].plot(alpha = 0.8,marker = 'D', markersize = 8)
plt.title('Summary Statistics for 10percentage')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

#Histogram
bins = np.arange(df['10percentage'].min(), df['10percentage'].
 ↪max()+df['10percentage'].std(),df['10percentage'].std()/3)
plt.figure(figsize = (15,6))
plt.hist(df['10percentage'], ec = 'k',bins = bins,label = f"Skewness :␣
 ↪{round(df['10percentage'].skew(),2)}",alpha = 0.7,density = True)
plt.xticks(bins)
plt.xlabel('10th Percentage', size = 15)
plt.ylabel('Density', size = 15)
plt.axvline(df['10percentage'].mean(), label = f"Mean:{round(df['10percentage'].
 ↪mean(),2)}", linestyle = '-.',color = 'red', linewidth = 2)
plt.axvline(df['10percentage'].median(), label = f"Median:
 ↪{round(df['10percentage'].median(),2)}", linestyle = '-.',color = 'green',␣
 ↪linewidth = 2)
```

```
plt.axvline(df['10percentage'].mode()[0], label = f"Mode:
↪{round(df['10percentage'].mode()[0],2)}", linestyle = '-.',color = 'k',␣
↪linewidth = 2)
sns.kdeplot(df['10percentage'])
plt.legend()
plt.show()

#Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['10percentage'])
plt.xlabel('10th percentage')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_10, y_10 = cdf(df['10percentage'])
x_sample_10 , y_sample_10 = \
cdf(np.random.normal(df['10percentage'].mean(), df['10percentage'].std(),size =␣
↪len(df['10percentage'])))
plt.plot(x_10, y_10, linestyle = 'None',marker = '.', color = 'blue',alpha = 0.
↪7, label = '10th %')
plt.plot(x_sample_10, y_sample_10, linestyle = 'None',marker ='.', color =␣
↪'red',alpha = 0.7, label = 'Normal Distribution')
plt.xlabel('10th Percentage')
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```


Summary Statistics for 10percentage
100
80
60
40
20
mean
std
min
25%
50%
75%
max
Statistical Measures

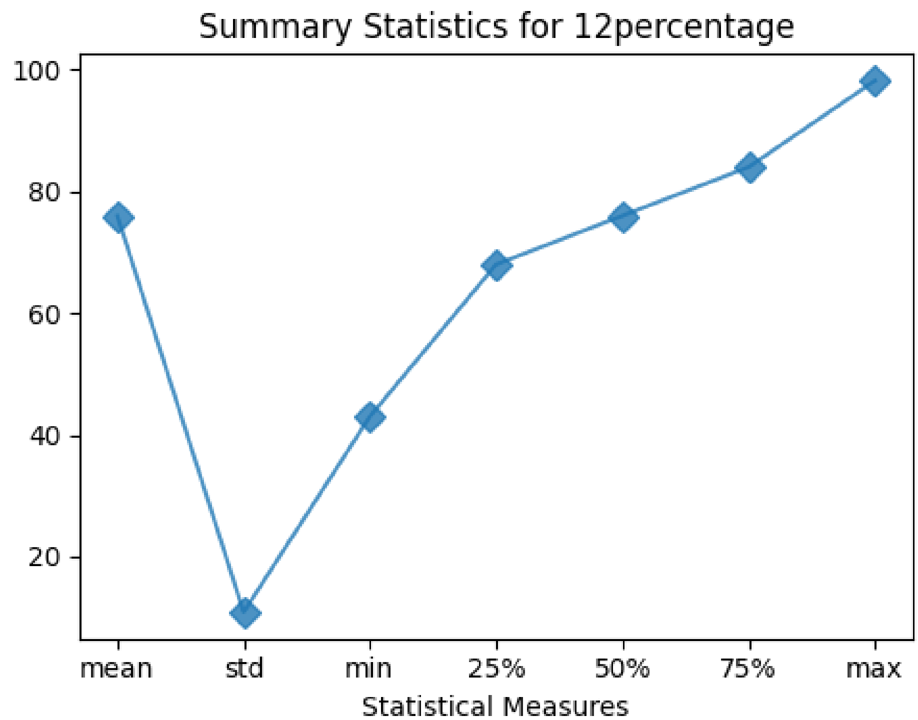

Skewness : -0.64
Mean:79.44
Median:81.0
Mode:82.0
Density
0.04
0.03
0.02
0.01
0.00
46.00 49.03 52.07 55.10 58.13 61.17 64.20 67.23 70.27 73.30 76.34 79.37 82.40 85.44 88.47 91.50 94.54 97.57 100.60 103.64 106.67
10th Percentage

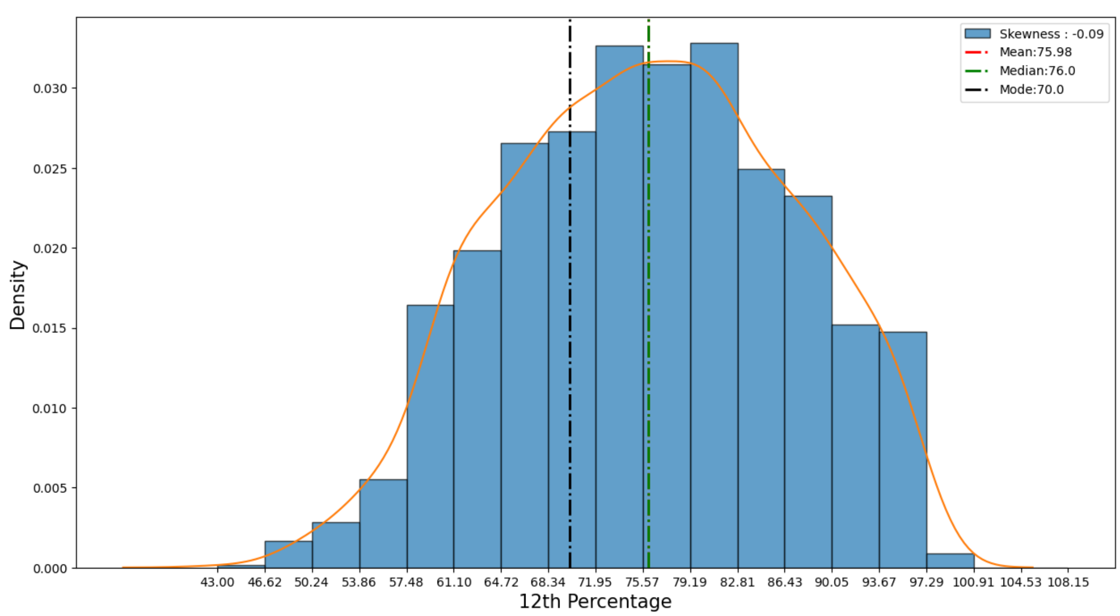

100
90
80
70
60
50
10percentage
10th percentage

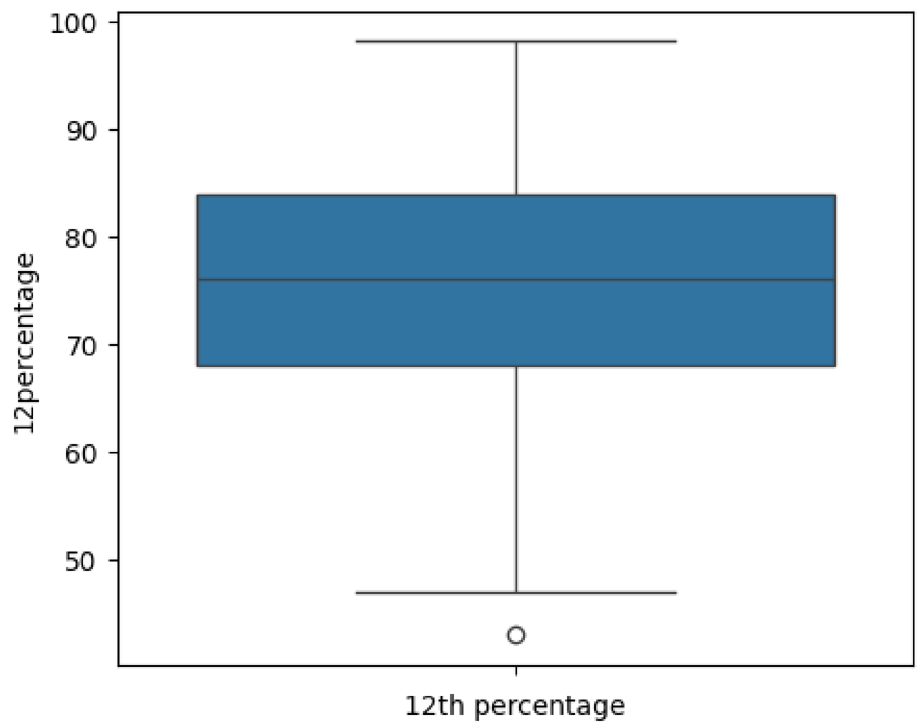

10th %
Normal Distribution
1.0
0.8
0.6
0.4
0.2
0.0
CDF
50
60
70
80
90
100
110
10th Percentage

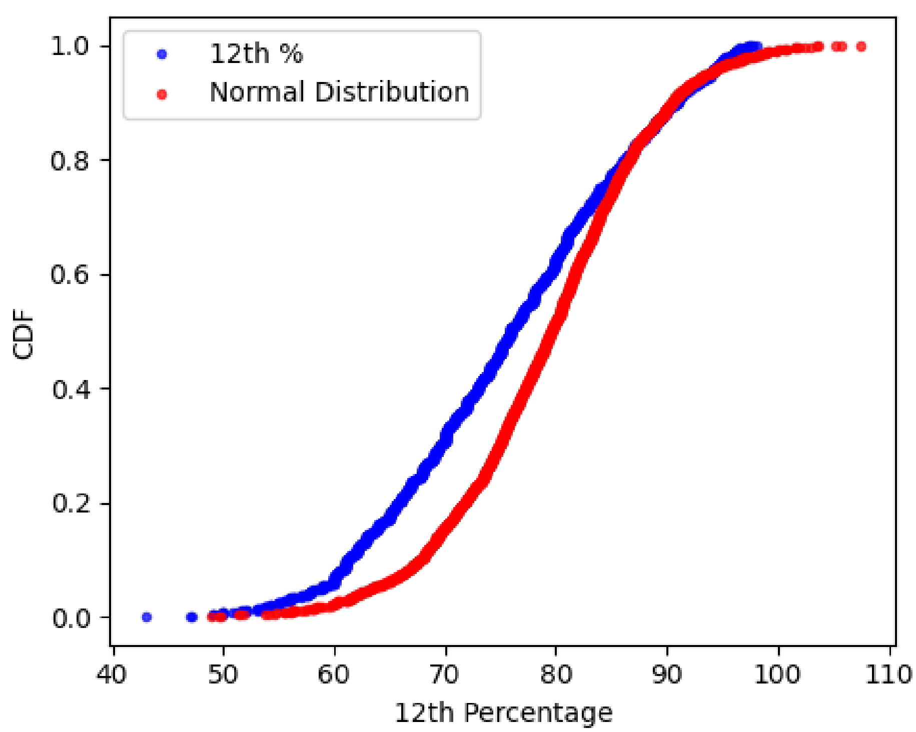
[85]:
```
# Summary Plot
plt.figure(figsize=(5,4))
df['12percentage'].describe()[1:].plot(alpha = 0.8,marker = 'D', markersize = 8)
plt.title('Summary Statistics for 12percentage')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['12percentage'].min(), df['12percentage'].
 ↪max()+df['12percentage'].std(),df['12percentage'].std()/3)
plt.figure(figsize = (15,8))
plt.hist(df['12percentage'], ec = 'k',bins = bins,label = f"Skewness :␣
 ↪{round(df['12percentage'].skew(),2)}",alpha = 0.7,density = True)
plt.xticks(bins)
plt.xlabel('12th Percentage', size = 15)
plt.ylabel('Density', size = 15)
plt.axvline(df['12percentage'].mean(), label = f"Mean:{round(df['12percentage'].
 ↪mean(),2)}", linestyle = '-.',color = 'red', linewidth = 2)
plt.axvline(df['12percentage'].median(), label = f"Median:
 ↪{round(df['12percentage'].median(),2)}", linestyle = '-.',color = 'green',␣
 ↪linewidth = 2)
plt.axvline(df['12percentage'].mode()[0], label = f"Mode:
 ↪{round(df['12percentage'].mode()[0],2)}", linestyle = '-.',color = 'k',␣
 ↪linewidth = 2)
sns.kdeplot(df['12percentage'])
plt.legend()
plt.show()

#Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['12percentage'])
plt.xlabel('12th percentage')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_12, y_12 = cdf(df['12percentage'])
x_sample_12 , y_sample_12 = cdf(np.random.normal(df['12percentage'].mean(),␣
 ↪df['12percentage'].std(),size = len(df['12percentage'])))
plt.plot(x_12, y_12, linestyle = 'None',marker = '.', color = 'blue',alpha = 0.
 ↪7, label = '12th %')
plt.plot(x_sample_10, y_sample_10, linestyle = 'None',marker ='.', color =␣
 ↪'red',alpha = 0.7, label = 'Normal Distribution')
plt.xlabel('12th Percentage')
```

```python
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```





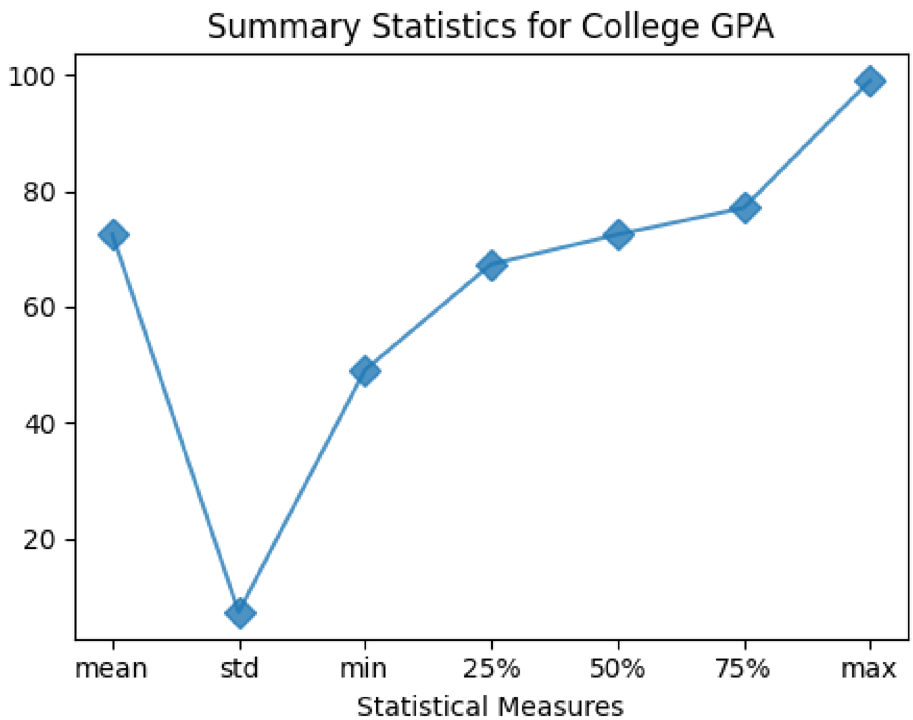

100
90
80
70
60
50
12percentage
12th percentage

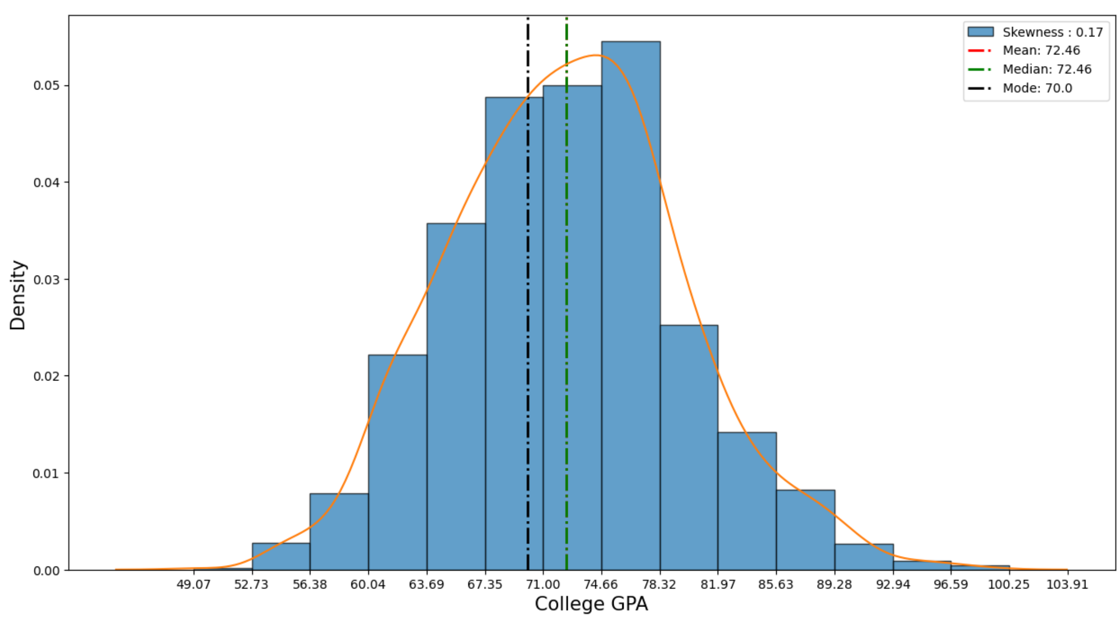

12th %
Normal Distribution
1.0
0.8
0.6
0.4
0.2
0.0
CDF
40
50
60
70
80
90
100
110
12th Percentage

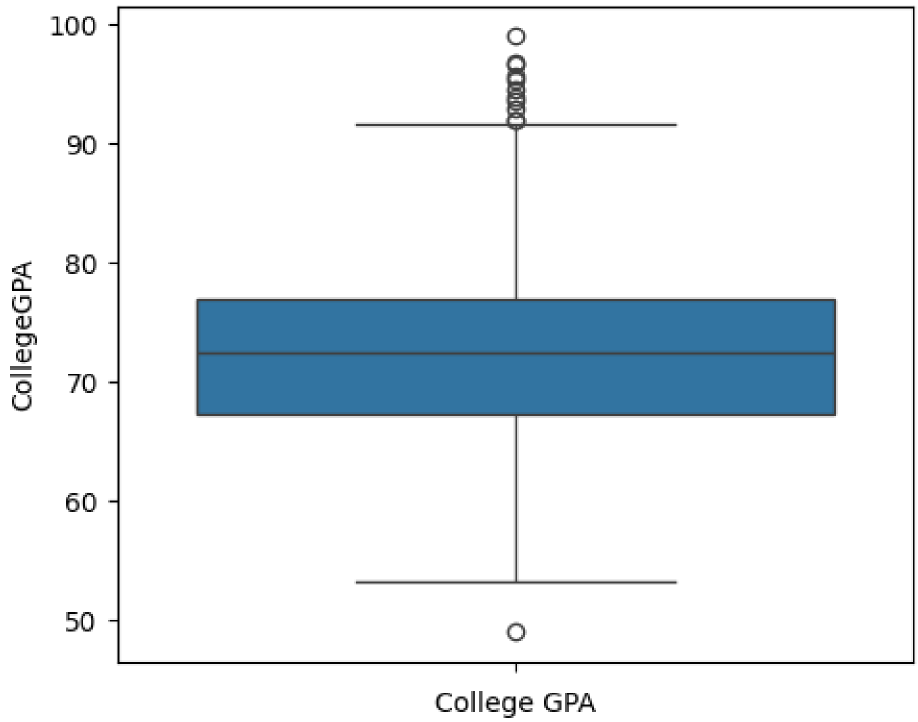
[87]:

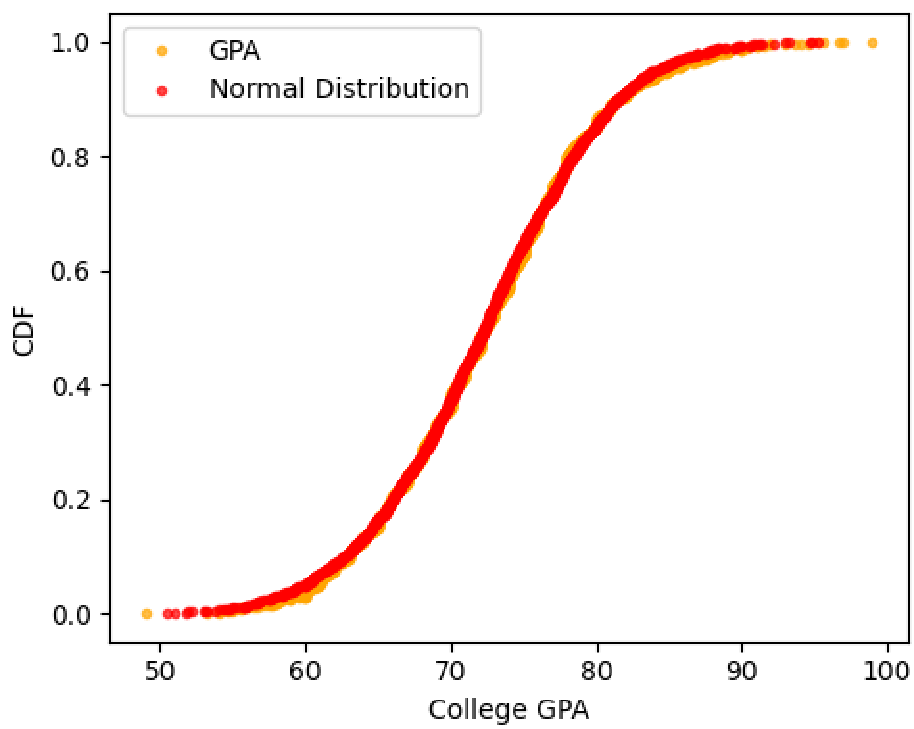
```
# Summary Statistics for College GPA
plt.figure(figsize=(5,4))
df['CollegeGPA'].describe()[1:].plot(alpha=0.8, marker='D', markersize=8)
plt.title('Summary Statistics for College GPA')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['CollegeGPA'].min(), df['CollegeGPA'].max() +␣
↪df['CollegeGPA'].std(), df['CollegeGPA'].std() / 2)
plt.figure(figsize=(15,8))
plt.hist(df['CollegeGPA'], ec='k', bins=bins,
         label=f"Skewness : {round(df['CollegeGPA'].skew(), 2)}",
         alpha=0.7, density=True)
plt.xticks(bins)
plt.xlabel('College GPA', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['CollegeGPA'].mean(), label=f"Mean: {round(df['CollegeGPA'].
↪mean(), 2)}", linestyle='-.', color='red', linewidth=2)
plt.axvline(df['CollegeGPA'].median(), label=f"Median: {round(df['CollegeGPA'].
↪median(), 2)}", linestyle='-.', color='green', linewidth=2)
plt.axvline(df['CollegeGPA'].mode()[0], label=f"Mode: {round(df['CollegeGPA'].
↪mode()[0], 2)}", linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['CollegeGPA'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['CollegeGPA'])
plt.xlabel('College GPA')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_gpa, y_gpa = cdf(df['CollegeGPA'])
x_sample_gpa, y_sample_gpa = cdf(np.random.normal(df['CollegeGPA'].mean(),␣
↪df['CollegeGPA'].std(), size=len(df['CollegeGPA'])))
plt.plot(x_gpa, y_gpa, linestyle='None', marker='.', color='orange', alpha=0.7,␣
↪label='GPA')
plt.plot(x_sample_gpa, y_sample_gpa, linestyle='None', marker='.', color='red',␣
↪alpha=0.7, label='Normal Distribution')
plt.xlabel('College GPA')
```

```
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```





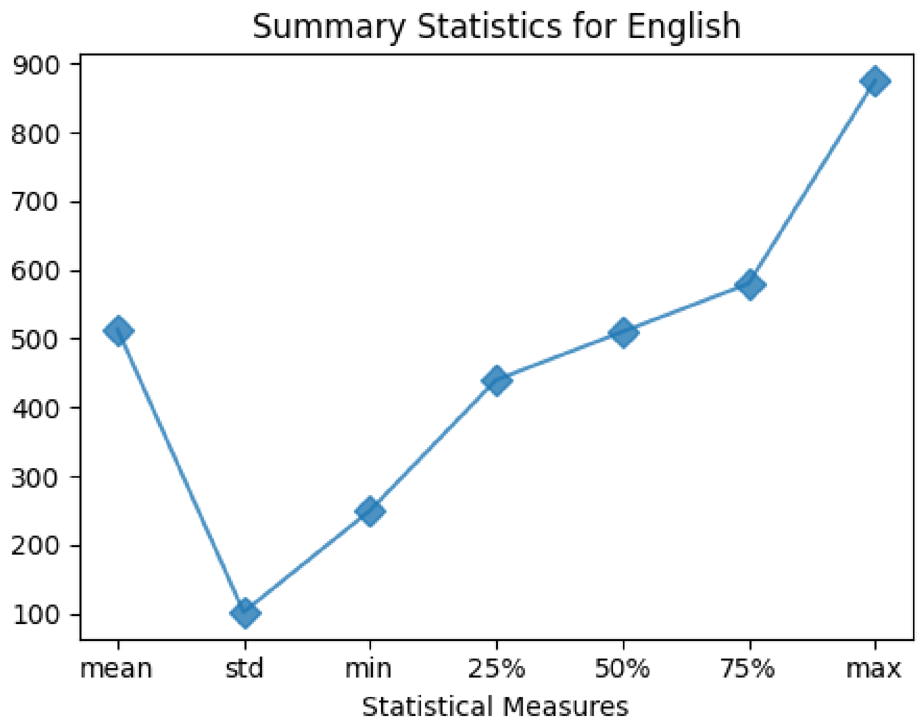

100
90
80
70
60
50
CollegeGPA
College GPA

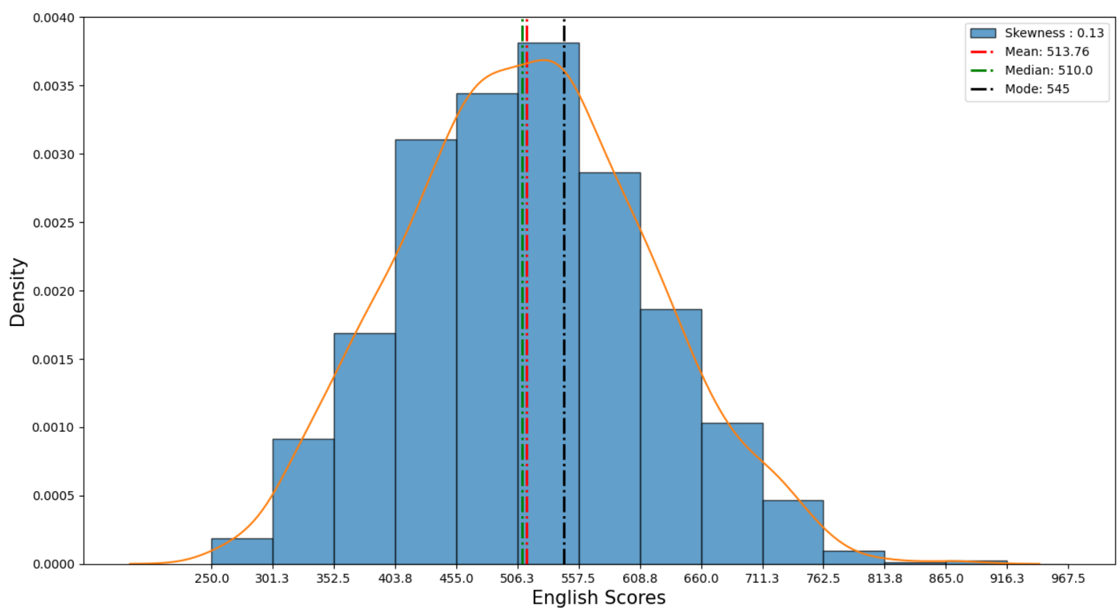

GPA
Normal Distribution
1.0
0.8
0.6
0.4
0.2
0.0
CDF
50
60
70
80
90
100
College GPA

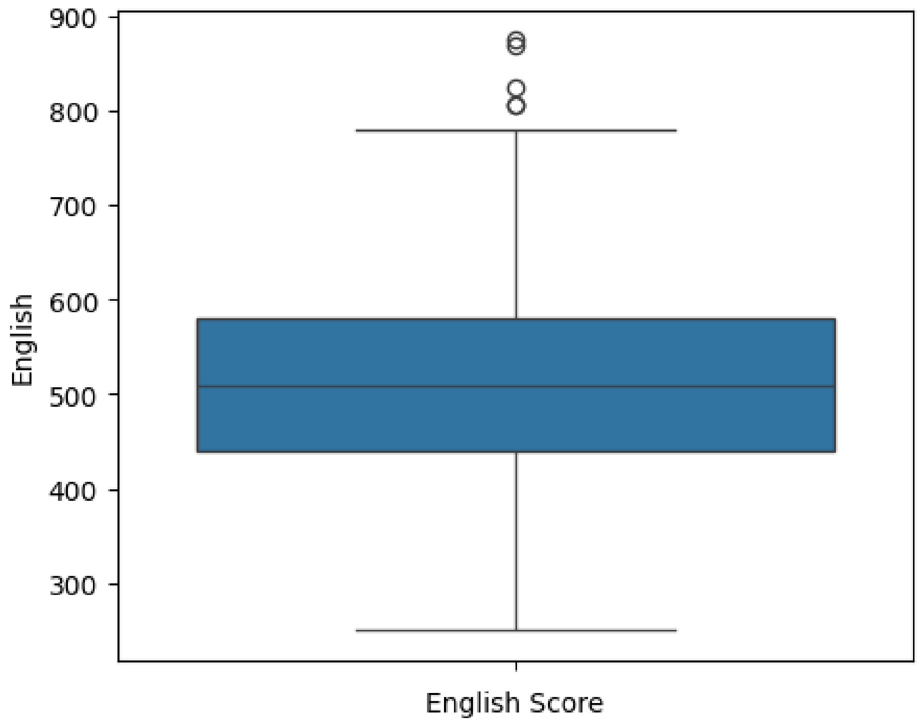
[88]:

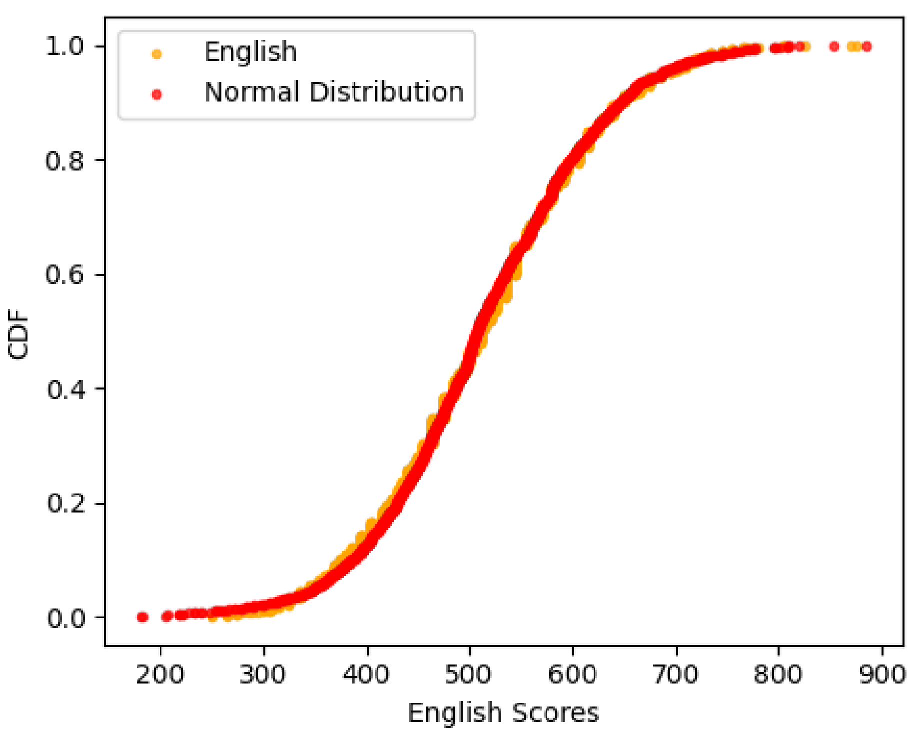
```
# Summary Plot
plt.figure(figsize=(5,4))
df['English'].describe()[1:].plot(alpha=0.8, marker='D', markersize=8)
plt.title('Summary Statistics for English')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['English'].min(), df['English'].max() + df['English'].std(), df['English'].std() / 2)
plt.figure(figsize=(15,8))
plt.hist(df['English'], ec='k', bins=bins,
         label=f"Skewness : {round(df['English'].skew(), 2)}",
         alpha=0.7, density=True)
plt.xticks(bins)
plt.xlabel('English Scores', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['English'].mean(), label=f"Mean: {round(df['English'].mean(), 2)}", linestyle='-.', color='red', linewidth=2)
plt.axvline(df['English'].median(), label=f"Median: {round(df['English'].median(), 2)}", linestyle='-.', color='green', linewidth=2)
plt.axvline(df['English'].mode()[0], label=f"Mode: {round(df['English'].mode()[0], 2)}", linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['English'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['English'])
plt.xlabel('English Score')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_eng, y_eng = cdf(df['English'])
x_sample_eng, y_sample_eng = cdf(np.random.normal(df['English'].mean(), df['English'].std(), size=len(df['English'])))
plt.plot(x_eng, y_eng, linestyle='None', marker='.', color='orange', alpha=0.7, label='English')
plt.plot(x_sample_eng, y_sample_eng, linestyle='None', marker='.', color='red', alpha=0.7, label='Normal Distribution')
plt.xlabel('English Scores')
```

```
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```





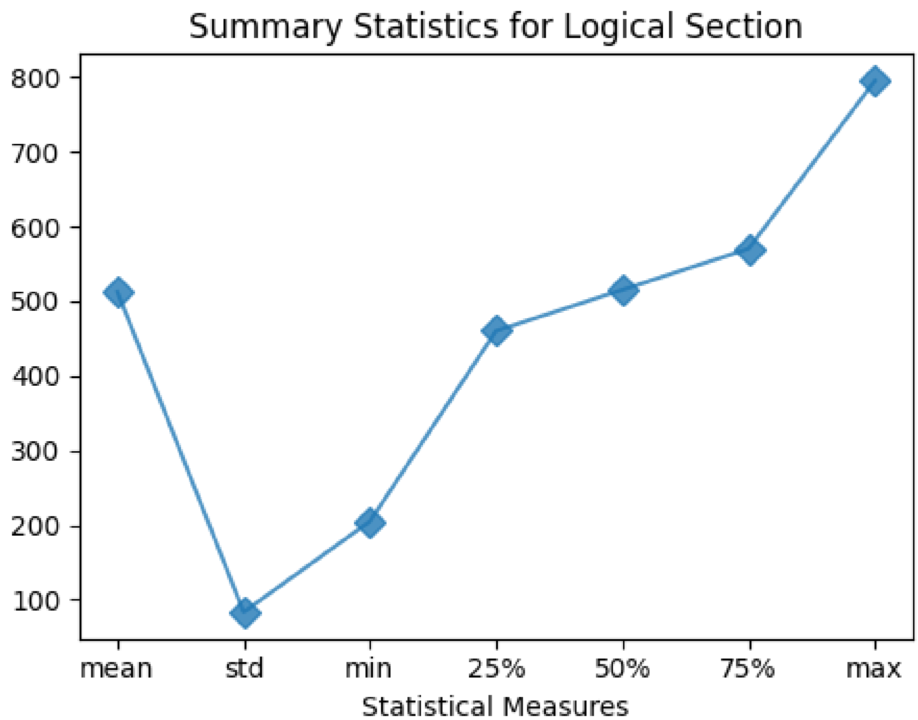

900
800
700
600
500
400
300
English
English Score

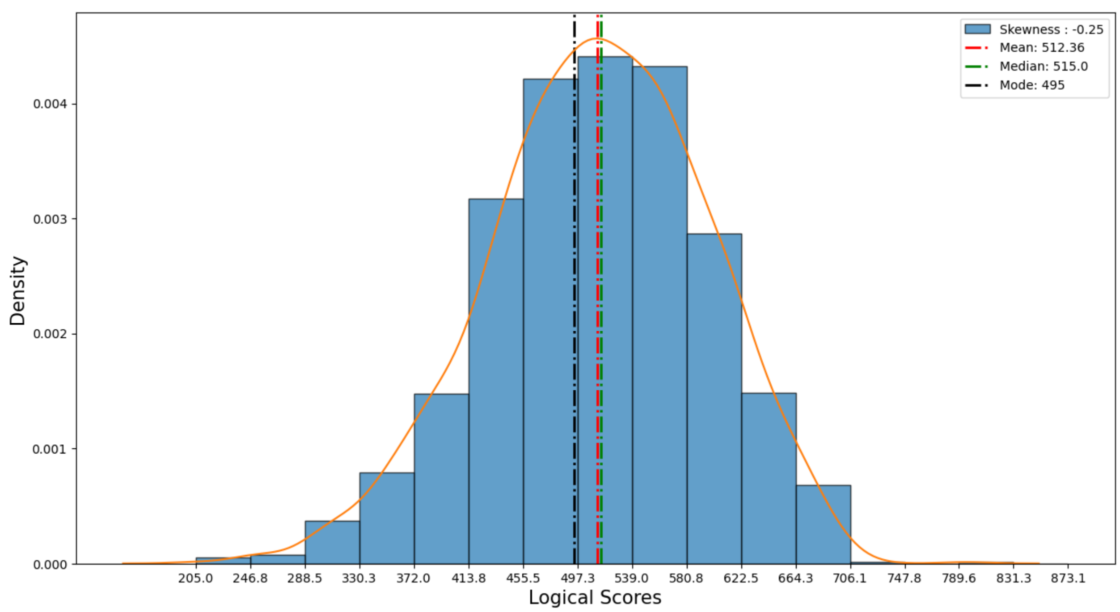

1.0
0.8
0.6
0.4
0.2
0.0
CDF
English
Normal Distribution
200
300
400
500
600
700
800
900
English Scores

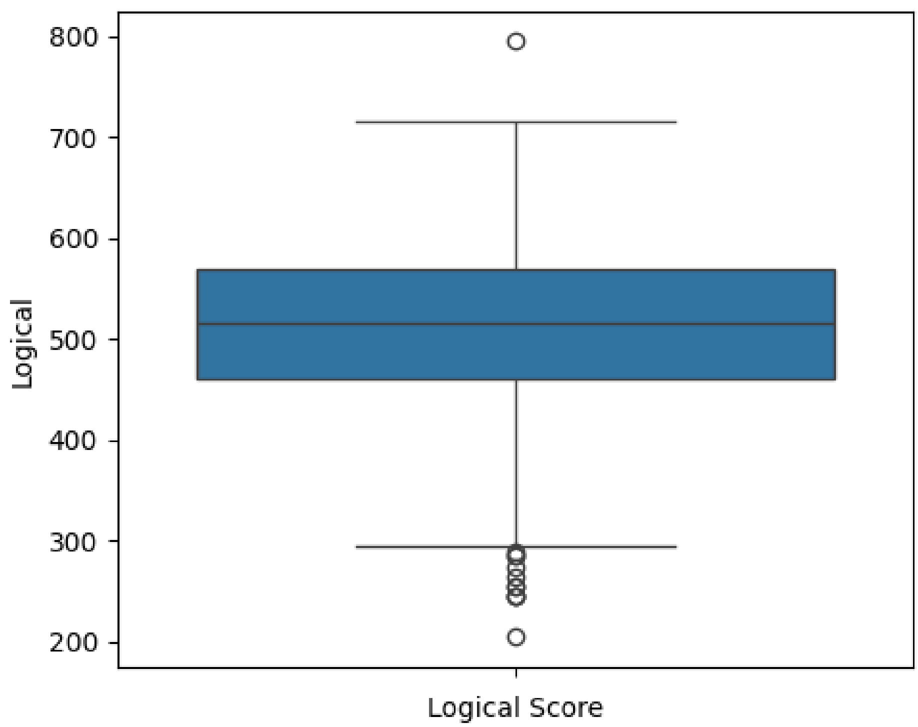
[89]:

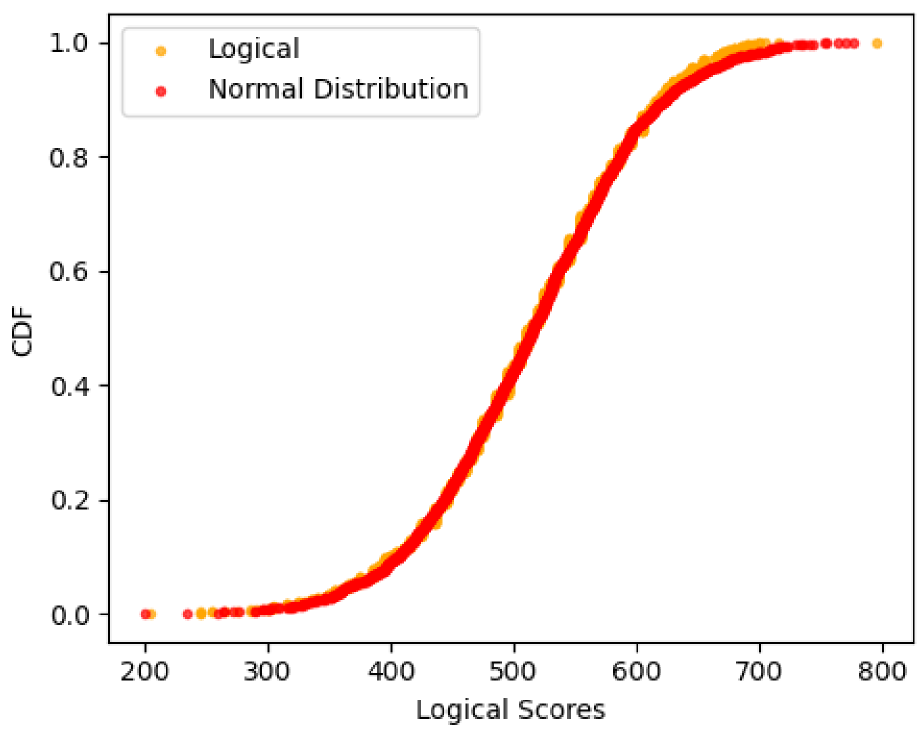
```python
# Summary Plot
plt.figure(figsize=(5,4))
df['Logical'].describe()[1:].plot(alpha=0.8, marker='D', markersize=8)
plt.title('Summary Statistics for Logical Section')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['Logical'].min(), df['Logical'].max() + df['Logical'].
 ↪std(), df['Logical'].std() / 2)
plt.figure(figsize=(15,8))
plt.hist(df['Logical'], ec='k', bins=bins,
         label=f"Skewness : {round(df['Logical'].skew(), 2)}",
         alpha=0.7, density=True)
plt.xticks(bins)
plt.xlabel('Logical Scores', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['Logical'].mean(), label=f"Mean: {round(df['Logical'].mean(),␣
 ↪2)}", linestyle='-.', color='red', linewidth=2)
plt.axvline(df['Logical'].median(), label=f"Median: {round(df['Logical'].
 ↪median(), 2)}", linestyle='-.', color='green', linewidth=2)
plt.axvline(df['Logical'].mode()[0], label=f"Mode: {round(df['Logical'].
 ↪mode()[0], 2)}", linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['Logical'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['Logical'])
plt.xlabel('Logical Score')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_log, y_log = cdf(df['Logical'])
x_sample_log, y_sample_log = cdf(np.random.normal(df['Logical'].mean(),␣
 ↪df['Logical'].std(), size=len(df['Logical'])))
plt.plot(x_log, y_log, linestyle='None', marker='.', color='orange', alpha=0.7,␣
 ↪label='Logical')
plt.plot(x_sample_log, y_sample_log, linestyle='None', marker='.', color='red',␣
 ↪alpha=0.7, label='Normal Distribution')
plt.xlabel('Logical Scores')
```

```
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```





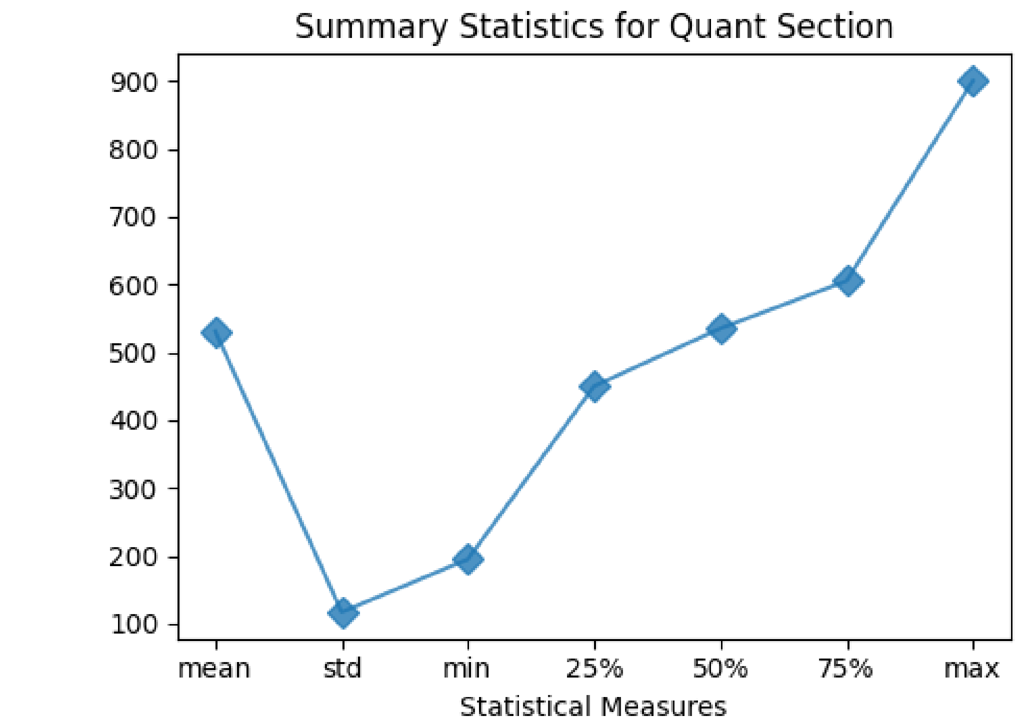

800
700
600
500
400
300
200
Logical
Logical Score

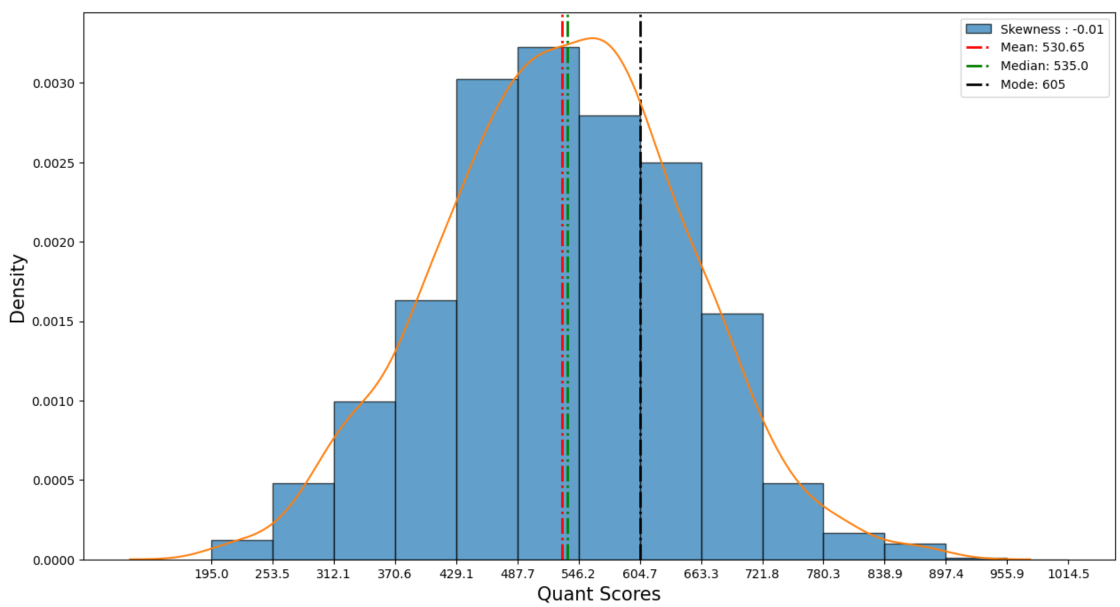

1.0
0.8
0.6
0.4
0.2
0.0
CDF
Logical
Normal Distribution
200
300
400
500
600
700
800
Logical Scores

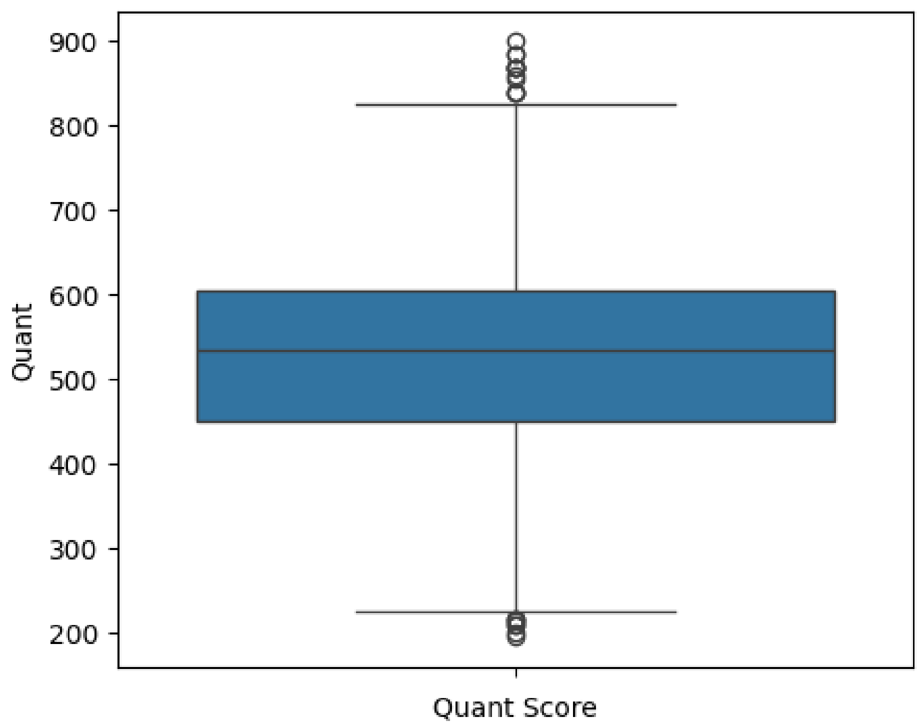
[91]:

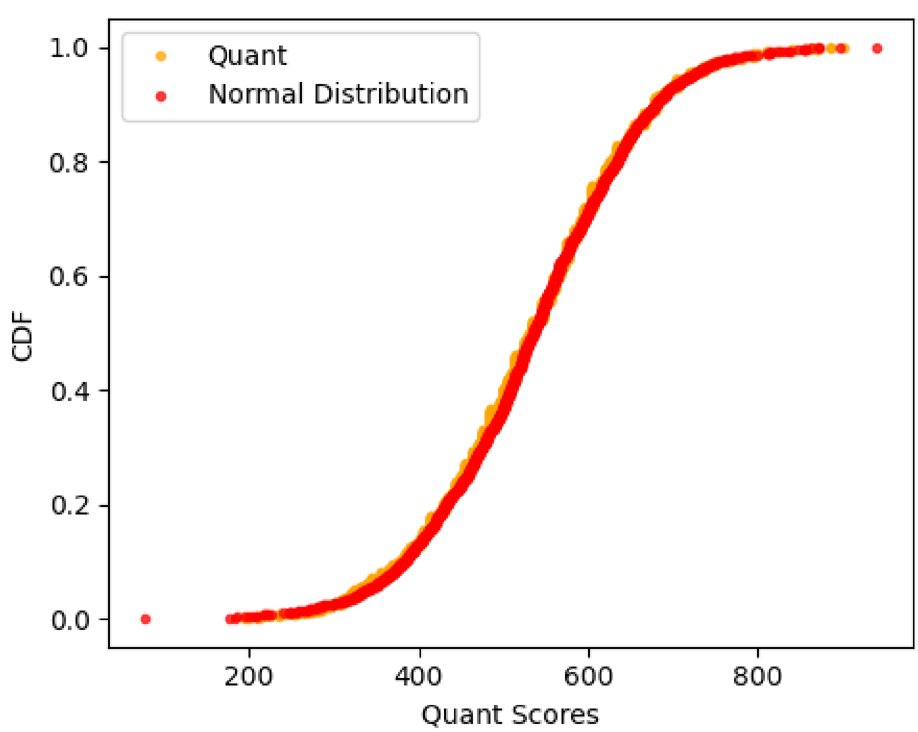
```
# Summary Plot
plt.figure(figsize=(5,4))
df['Quant'].describe()[1:].plot(alpha=0.8, marker='D', markersize=8)
plt.title('Summary Statistics for Quant Section')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['Quant'].min(), df['Quant'].max() + df['Quant'].std(),␣
 ↪df['Quant'].std() / 2)
plt.figure(figsize=(15,8))
plt.hist(df['Quant'], ec='k', bins=bins,
         label=f"Skewness : {round(df['Quant'].skew(), 2)}",
         alpha=0.7, density=True)
plt.xticks(bins)
plt.xlabel('Quant Scores', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['Quant'].mean(), label=f"Mean: {round(df['Quant'].mean(), 2)}",␣
 ↪linestyle='-.', color='red', linewidth=2)
plt.axvline(df['Quant'].median(), label=f"Median: {round(df['Quant'].median(),␣
 ↪2)}", linestyle='-.', color='green', linewidth=2)
plt.axvline(df['Quant'].mode()[0], label=f"Mode: {round(df['Quant'].mode()[0],␣
 ↪2)}", linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['Quant'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['Quant'])
plt.xlabel('Quant Score')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_q, y_q = cdf(df['Quant'])
x_sample_q, y_sample_q = cdf(np.random.normal(df['Quant'].mean(), df['Quant'].
 ↪std(), size=len(df['Quant'])))
plt.plot(x_q, y_q, linestyle='None', marker='.', color='orange', alpha=0.7,␣
 ↪label='Quant')
plt.plot(x_sample_q, y_sample_q, linestyle='None', marker='.', color='red',␣
 ↪alpha=0.7, label='Normal Distribution')
plt.xlabel('Quant Scores')
```

```
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```



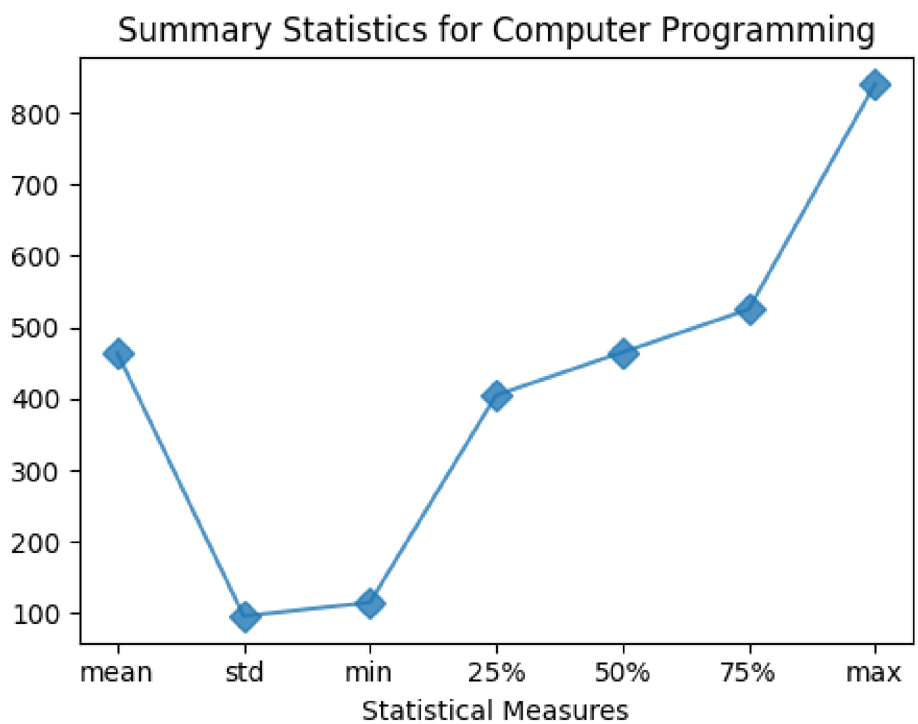


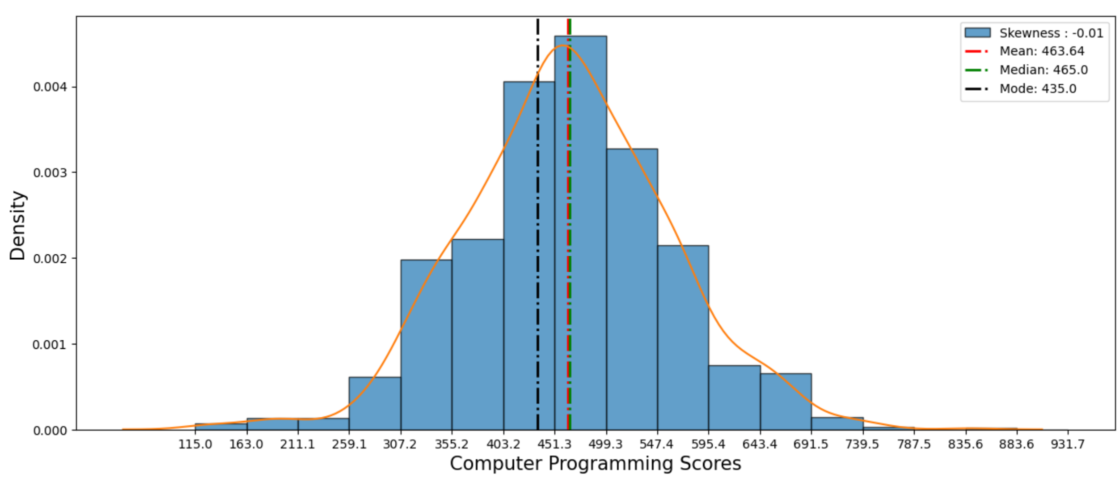

900
800
700
600
500
400
300
200
Quant
Quant Score

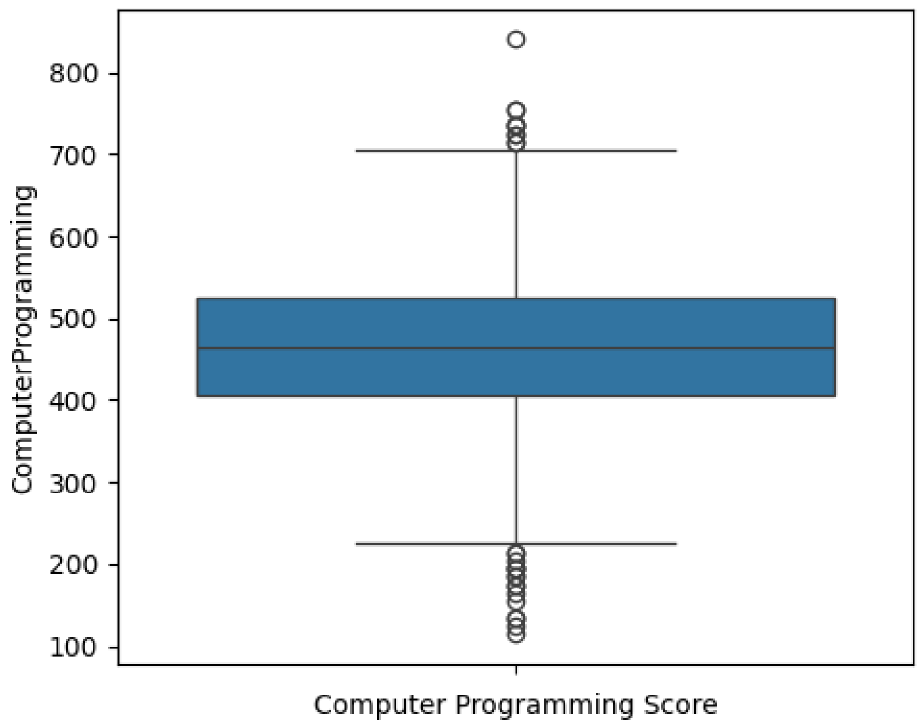

1.0
0.8
0.6
0.4
0.2
0.0
CDF
Quant
Normal Distribution
200
400
600
800
Quant Scores

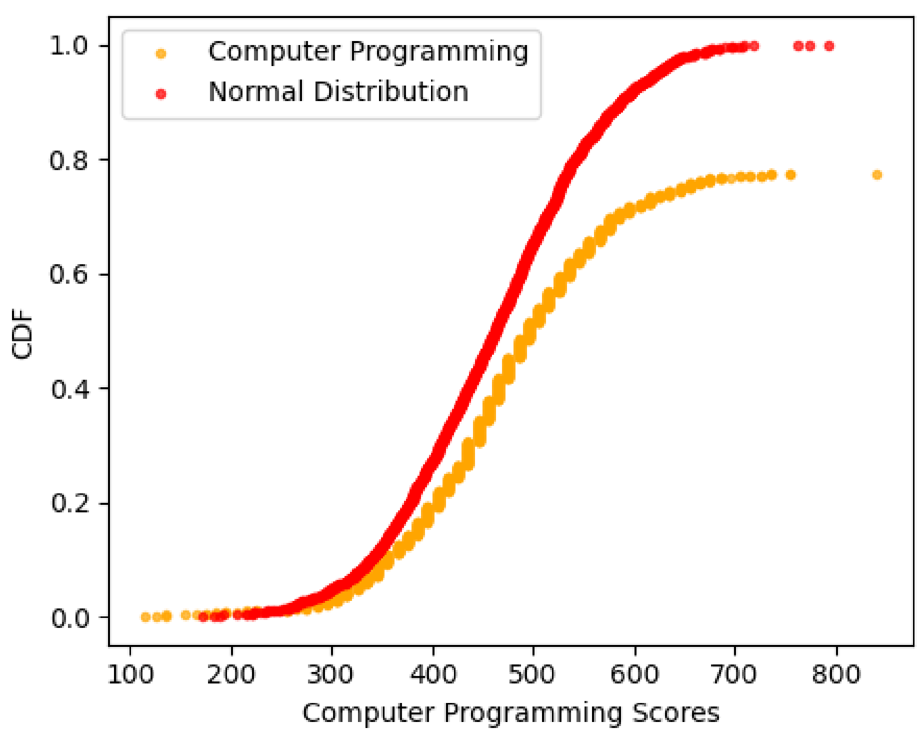
[94]:
```
# Summary Plot
plt.figure(figsize=(5,4))
df['ComputerProgramming'].describe()[1:].plot(alpha=0.8, marker='D',␣
↪markersize=8)
plt.title('Summary Statistics for Computer Programming')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['ComputerProgramming'].min(), df['ComputerProgramming'].
↪max() + df['ComputerProgramming'].std(), df['ComputerProgramming'].std() / 2)
plt.figure(figsize=(15,6))
plt.hist(df['ComputerProgramming'], ec='k', bins=bins,
         label=f"Skewness : {round(df['ComputerProgramming'].skew(), 2)}",
         alpha=0.7, density=True)
plt.xticks(bins)
plt.xlabel('Computer Programming Scores', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['ComputerProgramming'].mean(), label=f"Mean:␣
↪{round(df['ComputerProgramming'].mean(), 2)}", linestyle='-.', color='red',␣
↪linewidth=2)
plt.axvline(df['ComputerProgramming'].median(), label=f"Median:␣
↪{round(df['ComputerProgramming'].median(), 2)}", linestyle='-.',␣
↪color='green', linewidth=2)
plt.axvline(df['ComputerProgramming'].mode()[0], label=f"Mode:␣
↪{round(df['ComputerProgramming'].mode()[0], 2)}", linestyle='-.', color='k',␣
↪linewidth=2)
sns.kdeplot(df['ComputerProgramming'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5,4))
sns.boxplot(df['ComputerProgramming'])
plt.xlabel('Computer Programming Score')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5,4))
x_cp, y_cp = cdf(df['ComputerProgramming'])
x_sample_cp, y_sample_cp = cdf(np.random.normal(df['ComputerProgramming'].
↪mean(), df['ComputerProgramming'].std(),␣
↪size=len(df['ComputerProgramming'])))
```

```python
plt.plot(x_cp, y_cp, linestyle='None', marker='.', color='orange', alpha=0.7,␣
↪label='Computer Programming')
plt.plot(x_sample_cp, y_sample_cp, linestyle='None', marker='.', color='red',␣
↪alpha=0.7, label='Normal Distribution')
plt.xlabel('Computer Programming Scores')
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```



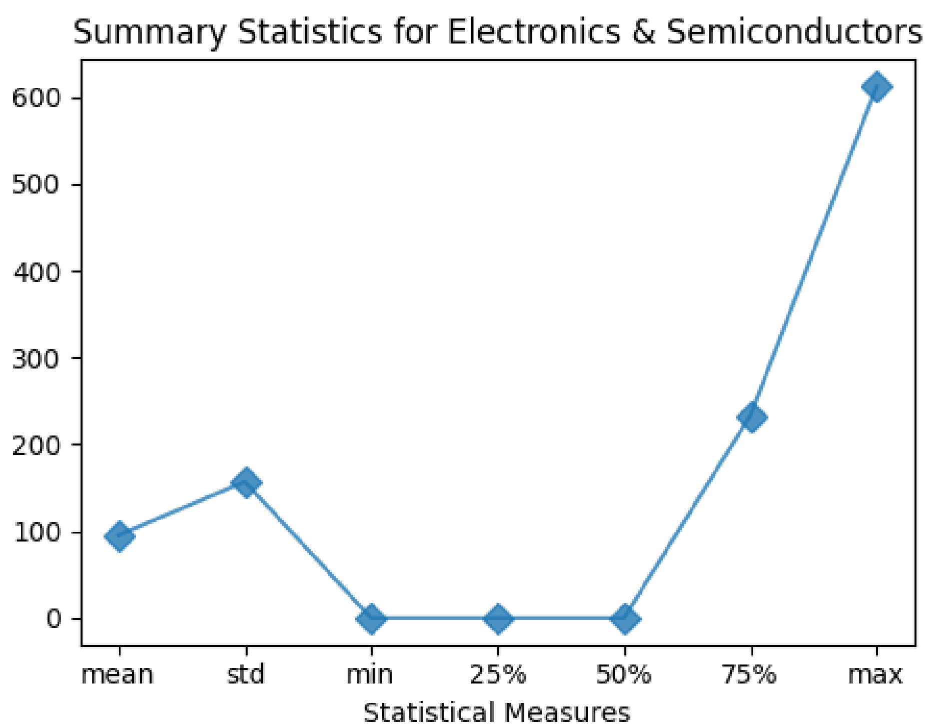


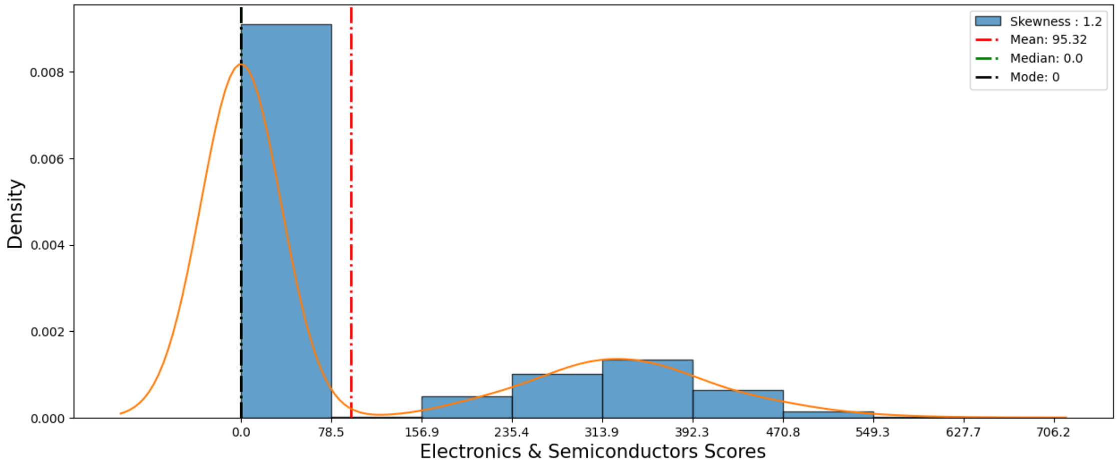

800
700
600
500
400
300
200
100
ComputerProgramming
Computer Programming Score

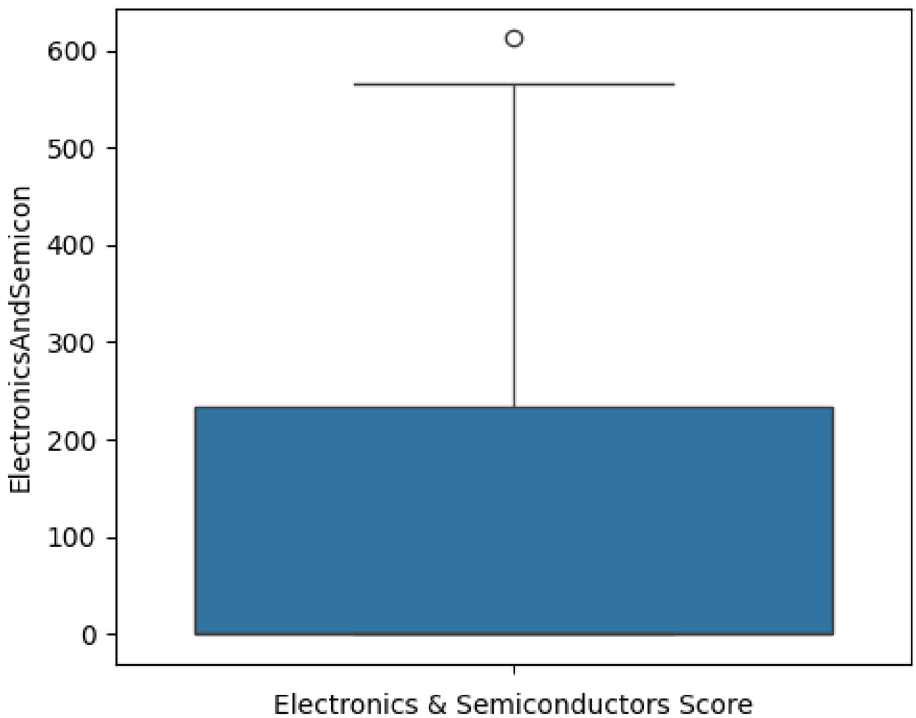

Computer Programming
Normal Distribution
1.0
0.8
0.6
0.4
0.2
0.0
CDF
100
200
300
400
500
600
700
800
Computer Programming Scores

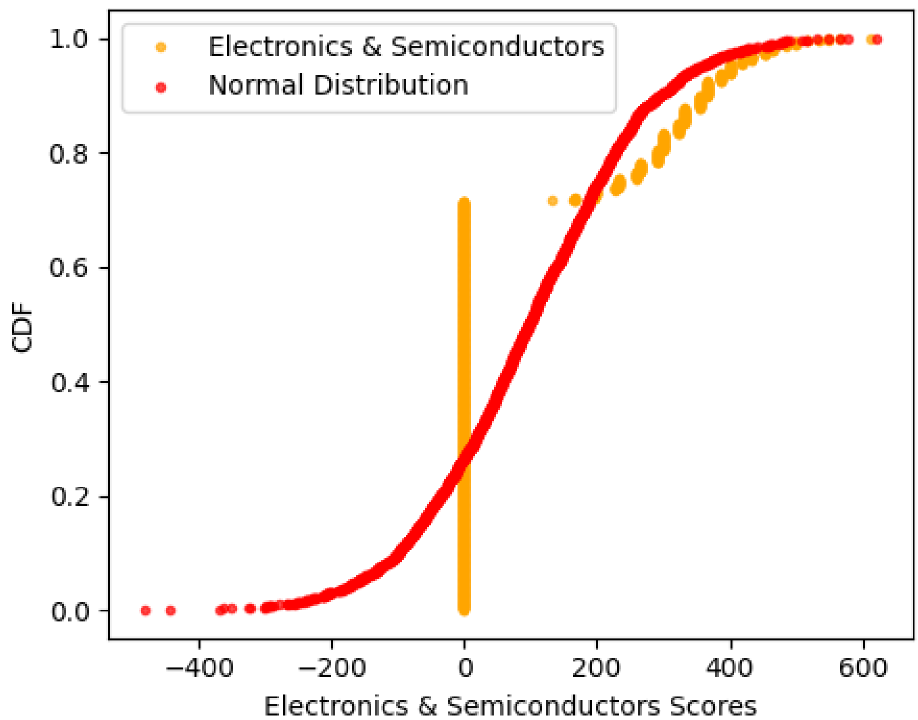
[95]:
```python
# Summary Plot
plt.figure(figsize=(5, 4))
df['ElectronicsAndSemicon'].describe()[1:].plot(alpha=0.8, marker='D',␣
↪markersize=8)
plt.title('Summary Statistics for Electronics & Semiconductors')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['ElectronicsAndSemicon'].min(),
                 df['ElectronicsAndSemicon'].max() +␣
↪df['ElectronicsAndSemicon'].std(),
                 df['ElectronicsAndSemicon'].std() / 2)
plt.figure(figsize=(15, 6))
plt.hist(df['ElectronicsAndSemicon'], ec='k',
         bins=bins,
         label=f"Skewness : {round(df['ElectronicsAndSemicon'].skew(), 2)}",
         alpha=0.7,
         density=True)
plt.xticks(bins)
plt.xlabel('Electronics & Semiconductors Scores', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['ElectronicsAndSemicon'].mean(),
            label=f"Mean: {round(df['ElectronicsAndSemicon'].mean(), 2)}",
            linestyle='-.', color='red', linewidth=2)
plt.axvline(df['ElectronicsAndSemicon'].median(),
            label=f"Median: {round(df['ElectronicsAndSemicon'].median(), 2)}",
            linestyle='-.', color='green', linewidth=2)
plt.axvline(df['ElectronicsAndSemicon'].mode()[0],
            label=f"Mode: {round(df['ElectronicsAndSemicon'].mode()[0], 2)}",
            linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['ElectronicsAndSemicon'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5, 4))
sns.boxplot(df['ElectronicsAndSemicon'])
plt.xlabel('Electronics & Semiconductors Score')
plt.tight_layout()
plt.show()

# CDF
```

```
plt.figure(figsize=(5, 4))
x_cp, y_cp = cdf(df['ElectronicsAndSemicon'])
x_sample_cp, y_sample_cp = cdf(np.random.normal(df['ElectronicsAndSemicon'].mean(),
                                                              df['ElectronicsAndSemicon'].std(),
                                                              size=len(df['ElectronicsAndSemicon'])))
plt.plot(x_cp, y_cp, linestyle='None', marker='.', color='orange', alpha=0.7, label='Electronics & Semiconductors')
plt.plot(x_sample_cp, y_sample_cp, linestyle='None', marker='.', color='red', alpha=0.7, label='Normal Distribution')
plt.xlabel('Electronics & Semiconductors Scores')
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```



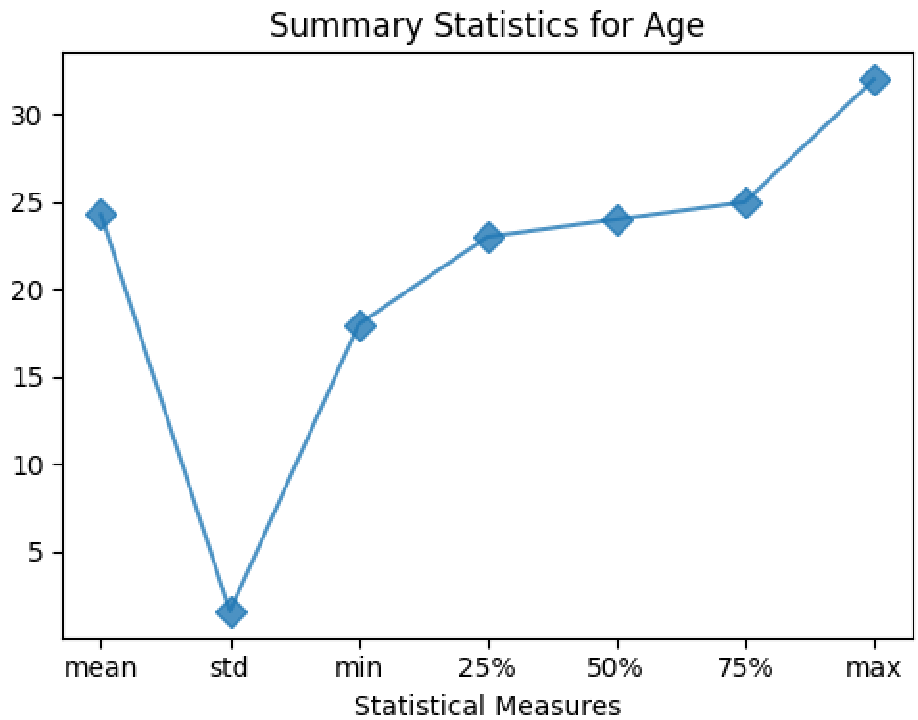

Skewness : 1.2
Mean: 95.32
Median: 0.0
Mode: 0
Density
0.000
0.002
0.004
0.006
0.008
0.0
78.5
156.9
235.4
313.9
392.3
470.8
549.3
627.7
706.2
Electronics & Semiconductors Scores

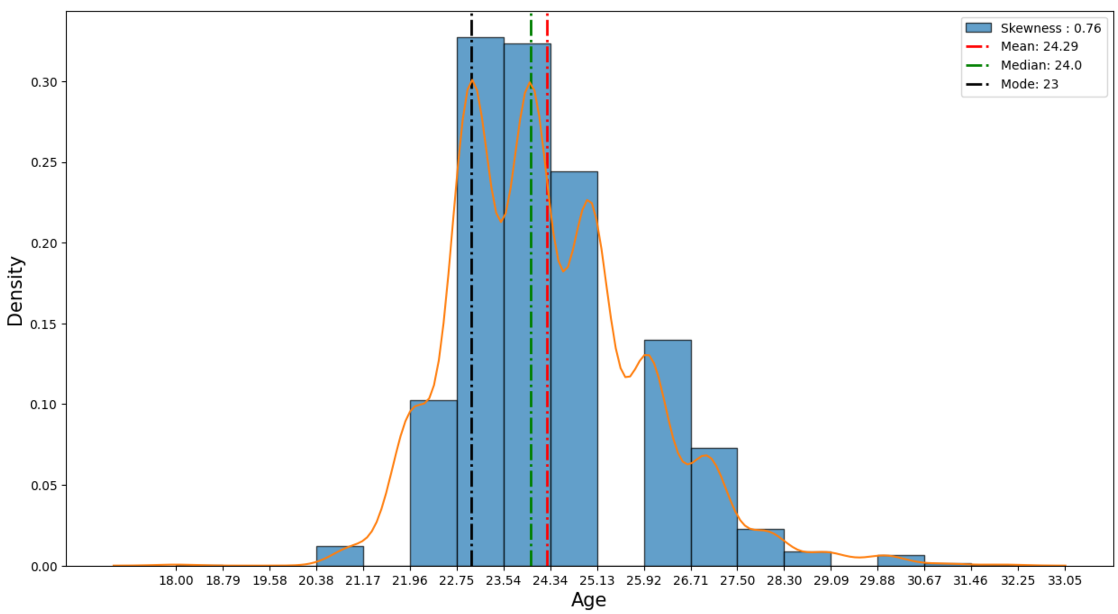

ElectronicsAndSemicon
0
100
200
300
400
500
600
Electronics & Semiconductors Score

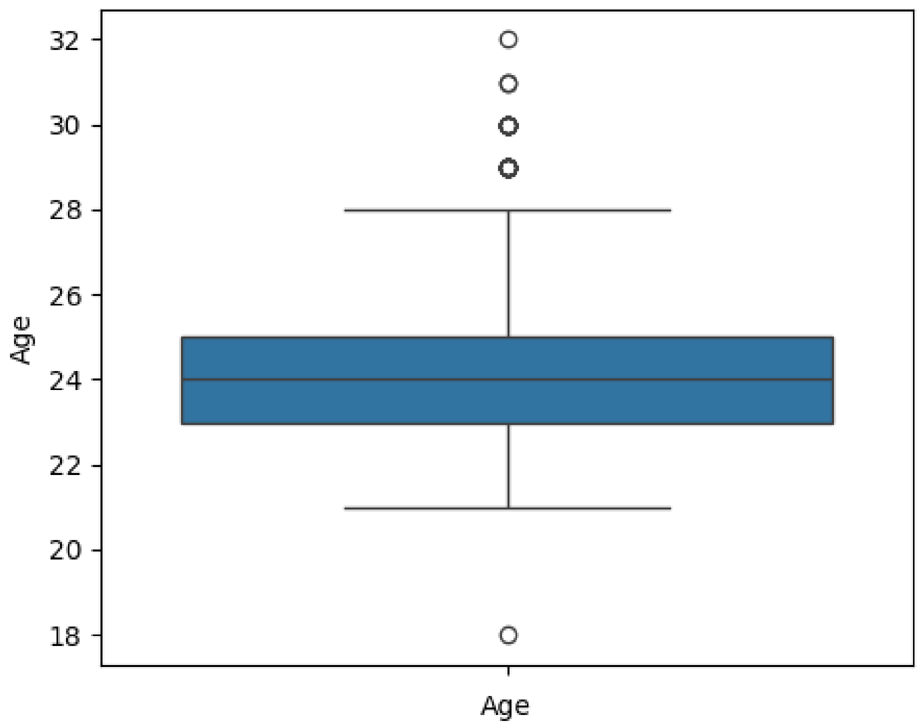


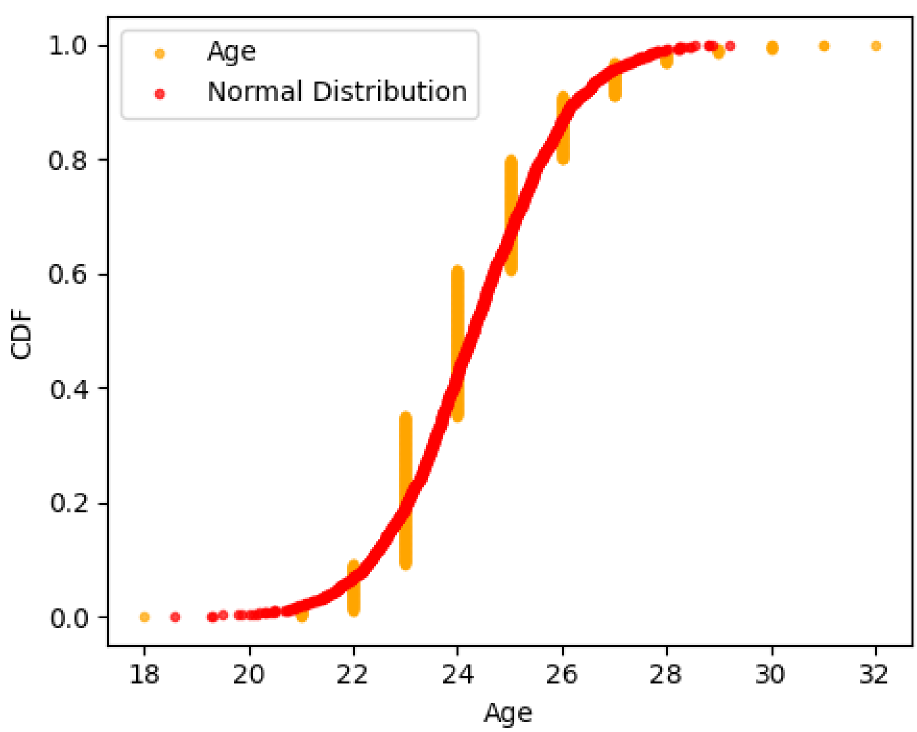
[96]:
```
# Summary Plot
plt.figure(figsize=(5, 4))
df['Age'].describe()[1:].plot(alpha=0.8, marker='D', markersize=8)
plt.title('Summary Statistics for Age')
plt.xlabel('Statistical Measures')
plt.tight_layout()
plt.show()

# Histogram
bins = np.arange(df['Age'].min(),
                 df['Age'].max() + df['Age'].std(),
                 df['Age'].std() / 2)
plt.figure(figsize=(15, 8))
plt.hist(df['Age'], ec='k',
         bins=bins,
         label=f"Skewness : {round(df['Age'].skew(), 2)}",
         alpha=0.7,
         density=True)
plt.xticks(bins)
plt.xlabel('Age', size=15)
plt.ylabel('Density', size=15)
plt.axvline(df['Age'].mean(),
            label=f"Mean: {round(df['Age'].mean(), 2)}",
```

```
            linestyle='-.', color='red', linewidth=2)
plt.axvline(df['Age'].median(),
            label=f"Median: {round(df['Age'].median(), 2)}",
            linestyle='-.', color='green', linewidth=2)
plt.axvline(df['Age'].mode()[0],
            label=f"Mode: {round(df['Age'].mode()[0], 2)}",
            linestyle='-.', color='k', linewidth=2)
sns.kdeplot(df['Age'])
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5, 4))
sns.boxplot(df['Age'])
plt.xlabel('Age')
plt.tight_layout()
plt.show()

# CDF
plt.figure(figsize=(5, 4))
x_cp, y_cp = cdf(df['Age'])
x_sample_cp, y_sample_cp = cdf(np.random.normal(df['Age'].mean(),
                                                df['Age'].std(),
                                                size=len(df['Age'])))
plt.plot(x_cp, y_cp, linestyle='None', marker='.', color='orange', alpha=0.7,␣
 ↪label='Age')
plt.plot(x_sample_cp, y_sample_cp, linestyle='None', marker='.', color='red',␣
 ↪alpha=0.7, label='Normal Distribution')
plt.xlabel('Age')
plt.ylabel('CDF')
plt.legend()
plt.tight_layout()
plt.show()
```

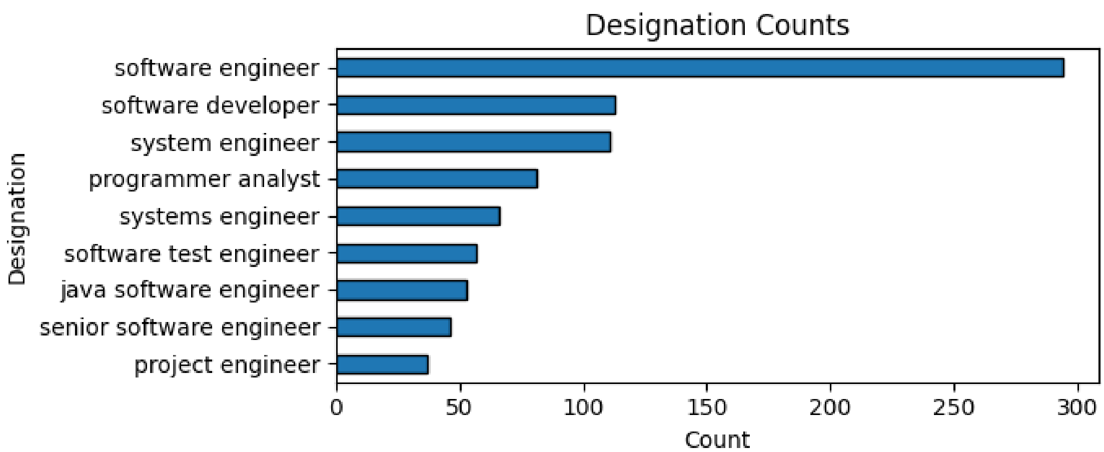

Summary Statistics for Age
30
25
20
15
10
5
mean
std
min
25%
50%
75%
max
Statistical Measures


Skewness : 0.76
Mean: 24.29
Median: 24.0
Mode: 23
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
18.00 18.79 19.58 20.38 21.17 21.96 22.75 23.54 24.34 25.13 25.92 26.71 27.50 28.30 29.09 29.88 30.67 31.46 32.25 33.05
Age

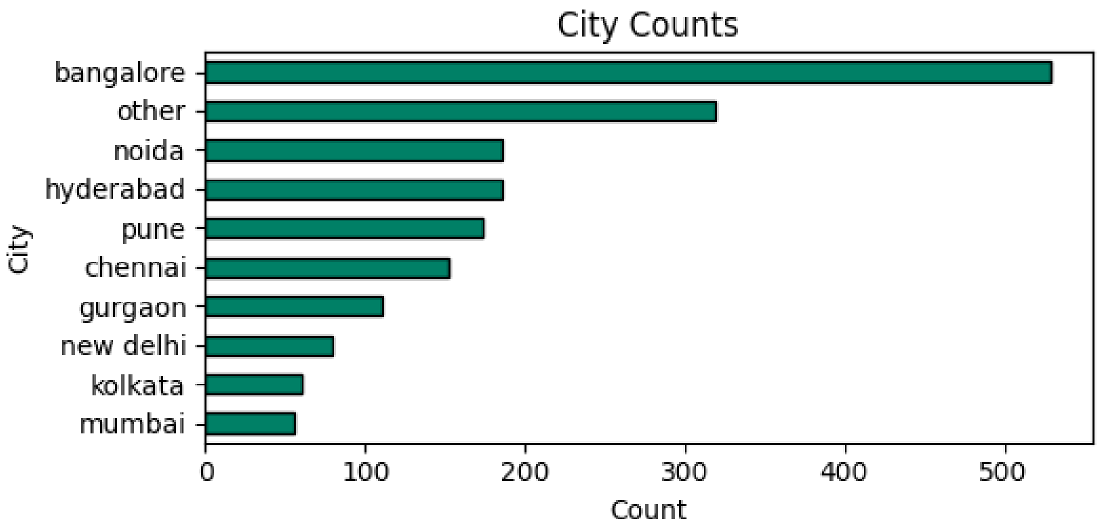

32
30
28
26
24
22
20
18
Age
Age


1.0
0.8
0.6
0.4
0.2
0.0
CDF
Age
Normal Distribution
18
20
22
24
26
28
30
32
Age

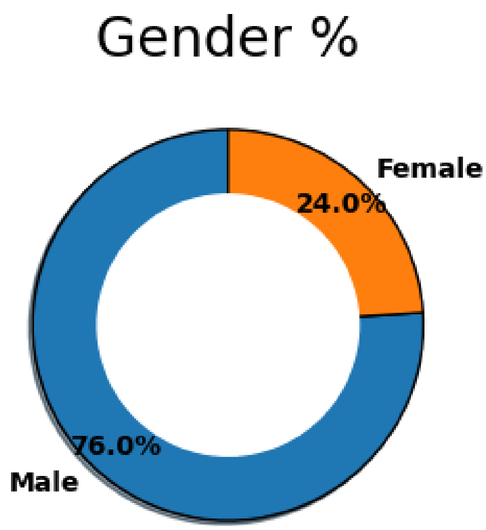
[97]:
```
df['Designation'].value_counts()[1:].sort_values(
    ascending=True
).plot(
    kind='barh',
    title='Designation Counts',
    figsize=(7, 3),
    ec='k'
)
plt.ylabel('Designation')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



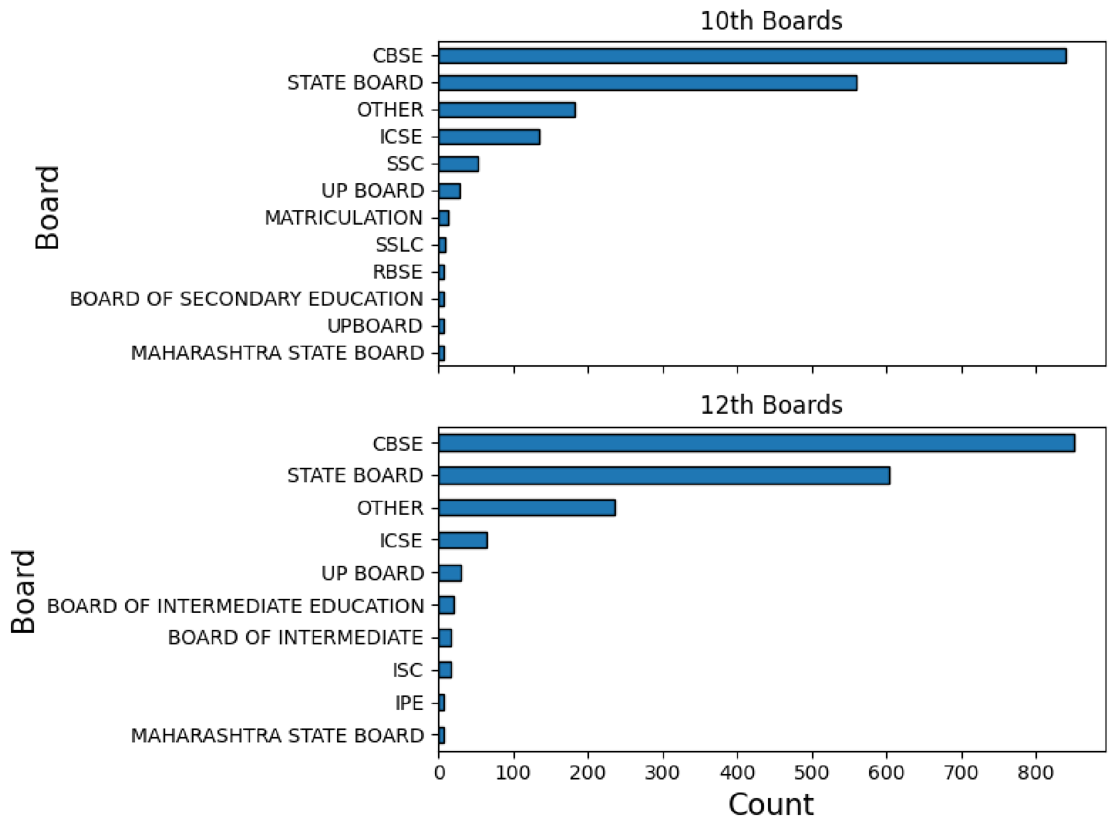
[98]:
```
df['JobCity'].value_counts().sort_values(ascending=True).plot(
    kind='barh',
    cmap='summer',
    title='City Counts',
    figsize=(6, 3),
    ec='k'
)
plt.ylabel('City')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



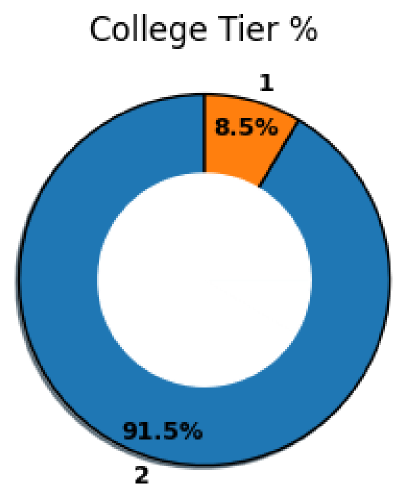
[99]:
```
plt.figure(figsize=(3, 3))
plt.pie(df['Gender'].value_counts().tolist(),
        labels=df['Gender'].value_counts().index,
        autopct='%1.1f%%',
        radius=1.5,
        wedgeprops={'edgecolor': 'k'},
        textprops={'fontsize': 10, 'fontweight': 'bold'},
        shadow=True,
        startangle=90,
        pctdistance=0.85)
plt.pie(df['Gender'].value_counts().tolist(),
        colors=['white'],
        wedgeprops={'edgecolor': 'white'},
        radius=1)
plt.title('Gender %', pad=40, size=20)
plt.tight_layout()
plt.show()
```



[100]:
```
fig, ax = plt.subplots(2, 1, figsize=(8, 6), sharex=True)

# Plot for 10th Boards
df['10board'].str.upper().value_counts().sort_values(ascending=True).plot(
    kind='barh',
    ax=ax[0],
    ec='k',
    title='10th Boards'
)
ax[0].set_ylabel('Board', size=15)

# Plot for 12th Boards
df['12board'].str.upper().value_counts().sort_values(ascending=True).plot(
    kind='barh',
    ax=ax[1],
    ec='k',
    title='12th Boards'
)
ax[1].set_ylabel('Board', size=15)
ax[1].set_xlabel('Count', size=15)

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

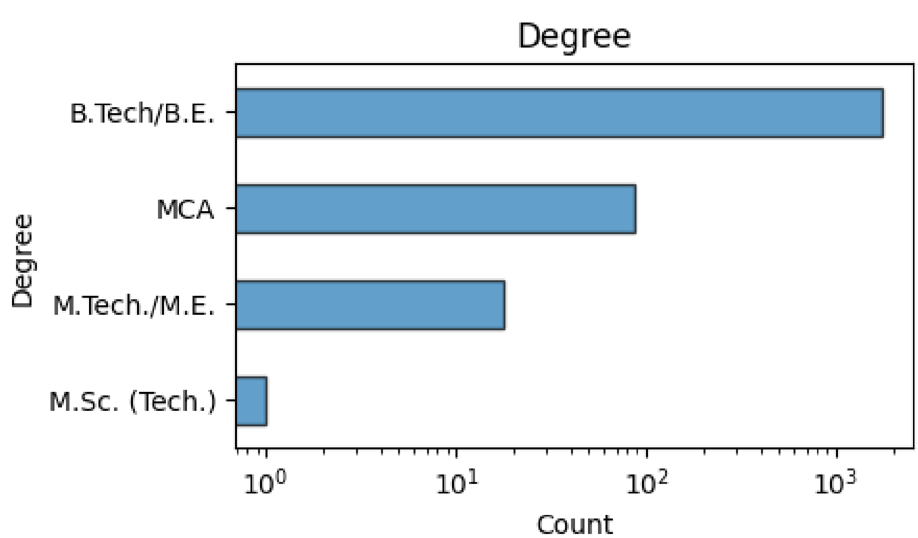


[101]:
```
plt.figure(figsize=(3, 3))
plt.pie(df['CollegeTier'].value_counts().tolist(),
        labels=df['CollegeTier'].value_counts().index,
        autopct='%1.1f%%',
        radius=1.75,
        wedgeprops={'edgecolor': 'k'},
        textprops={'fontsize': 9, 'fontweight': 'bold'},
        shadow=True,
        startangle=90,
        pctdistance=0.85)

plt.pie(df['CollegeTier'].value_counts().tolist(), colors=['white'],
        wedgeprops={'edgecolor': 'white'},
        radius=1)

plt.title('College Tier %', pad=40, size=12)
plt.margins(0.02)
plt.tight_layout()
plt.show()
```



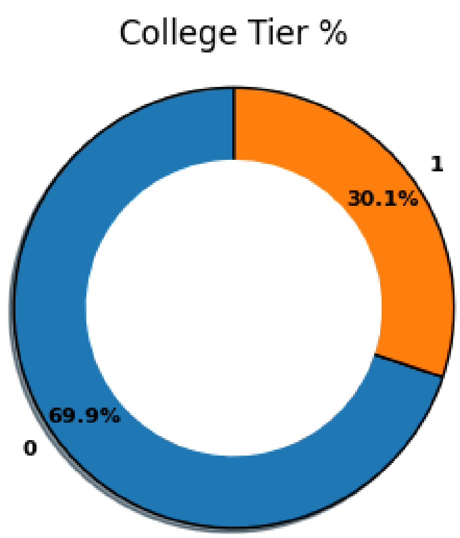
[103]:

```
df['Degree'].value_counts().sort_values(ascending=True).plot(
    kind='barh',
    title='Degree',
    figsize=(5, 3),
    ec='k',
    alpha=0.7
)
plt.ylabel('Degree')
plt.xlabel('Count')
plt.xscale('log')
plt.tight_layout()
plt.show()
```



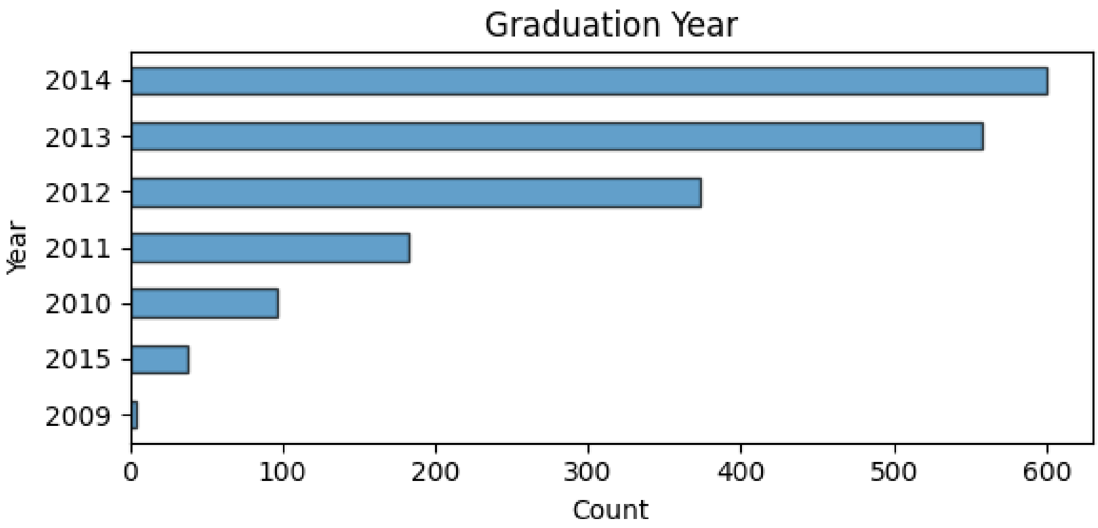
[105]:

```
plt.figure(figsize=(3, 3))
plt.pie(df['CollegeCityTier'].value_counts().tolist(),␣
↪labels=df['CollegeCityTier'].value_counts().index,
        autopct='%1.1f%%', radius=1.5,
        wedgeprops={'edgecolor': 'k'},
        textprops={'fontsize': 8, 'fontweight': 'bold'},
        shadow=True, startangle=90, pctdistance=0.84)
plt.pie(df['CollegeCityTier'].value_counts().tolist(), colors=['white'],
        wedgeprops={'edgecolor': 'white'}, radius=1)
plt.title('College Tier %', pad=30, size=12)
plt.margins(0.02)
plt.tight_layout()
plt.show()
```



[106]:

```
df['GraduationYear'].value_counts().sort_values(ascending=True).plot(
    kind='barh',
    title='Graduation Year',
    figsize=(6, 3),
    ec='k',
    alpha=0.7
)
plt.ylabel('Year')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



[107]:
```
def outlier_treatment(datacolumn):
    sorted_data = np.sort(datacolumn)  # Sort the data
    Q1, Q3 = np.percentile(sorted_data, [25, 75])  # Calculate the 1st and 3rd␣
↪quartiles
    IQR = Q3 - Q1  # Calculate the Interquartile Range
    lower_range = Q1 - (1.5 * IQR)  # Calculate the lower range for outliers
    upper_range = Q3 + (1.5 * IQR)  # Calculate the upper range for outliers
    return lower_range, upper_range
```

[108]:
```
df.columns
```

[108]:
```
Index(['ID', 'Salary', 'DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB',
       '10percentage', '10board', '12graduation', '12percentage', '12board',
       'CollegeID', 'CollegeTier', 'Degree', 'Specialization', 'CollegeGPA',
       'CollegeCityID', 'CollegeCityTier', 'CollegeState', 'GraduationYear',
       'English', 'Logical', 'Quant', 'Domain', 'ComputerProgramming',
       'ElectronicsAndSemicon', 'ComputerScience', 'conscientiousness',
       'agreeableness', 'extraversion', 'neuroticism', 'openess_to_experience',
       'Age', 'Tenure'],
      dtype='object')
```

[113]:
```
columns =␣
↪['Salary','10percentage','12percentage','English','Logical','Quant','Domain',␣
↪'ComputerProgramming','ElectronicsAndSemicon', 'ComputerScience',␣
↪'conscientiousness','agreeableness', 'extraversion', 'neuroticism',␣
↪'openess_to_experience','Age', 'Tenure']
df2 = df.copy()
```

[115]:
```
for cols in columns:
    lowerbound, upperbound = outlier_treatment(df2[cols])
    df2 = df2.drop(df2[(df2[cols] < lowerbound) | (df2[cols] > upperbound)].
 ↪index)
```

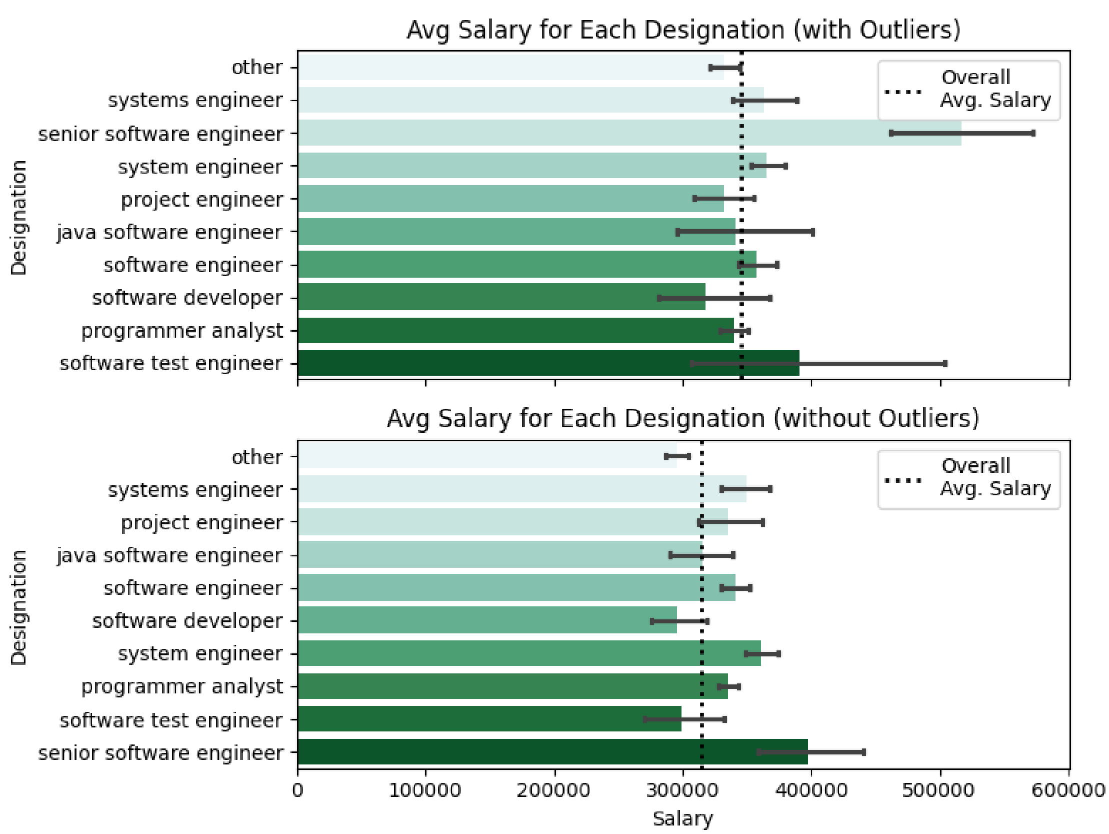
[116]:
```
print(f'Number of observation with outliers: {df1.shape[0]}')
print(f'Number of observations without outliers: {df2.shape[0]}')
```

```
Number of observation with outliers: 1852
Number of observations without outliers: 1551
```

2 Bivariate Analysis

[117]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Check for null values
if df1['Salary'].isnull().any() or df1['Designation'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Salary'].isnull().any() or df2['Designation'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for visualizing average salary by designation
fig, ax = plt.subplots(2, 1, figsize=(8, 6), sharex=True)

# Bar plot for df1 with outliers
sns.barplot(x='Salary', y='Designation', data=df1, palette='BuGn', capsize=0.1,␣
 ↪ax=ax[0])
ax[0].axvline(df1['Salary'].mean(), color='k', linestyle=':', linewidth=2,␣
 ↪label='Overall\nAvg. Salary')
ax[0].set_title('Avg Salary for Each Designation (with Outliers)')
ax[0].legend()
ax[0].set_xlabel('')

# Bar plot for df2 without outliers
sns.barplot(x='Salary', y='Designation', data=df2, palette='BuGn', capsize=0.1,␣
 ↪ax=ax[1])
ax[1].axvline(df2['Salary'].mean(), color='k', linestyle=':', linewidth=2,␣
 ↪label='Overall\nAvg. Salary')
ax[1].set_title('Avg Salary for Each Designation (without Outliers)')
ax[1].legend()
ax[1].set_xlabel('Salary')

# Adjust layout and show plot
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2793694086.py:14:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Salary', y='Designation', data=df1, palette='BuGn',
capsize=0.1, ax=ax[0])
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2793694086.py:21:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Salary', y='Designation', data=df2, palette='BuGn',
capsize=0.1, ax=ax[1])
```

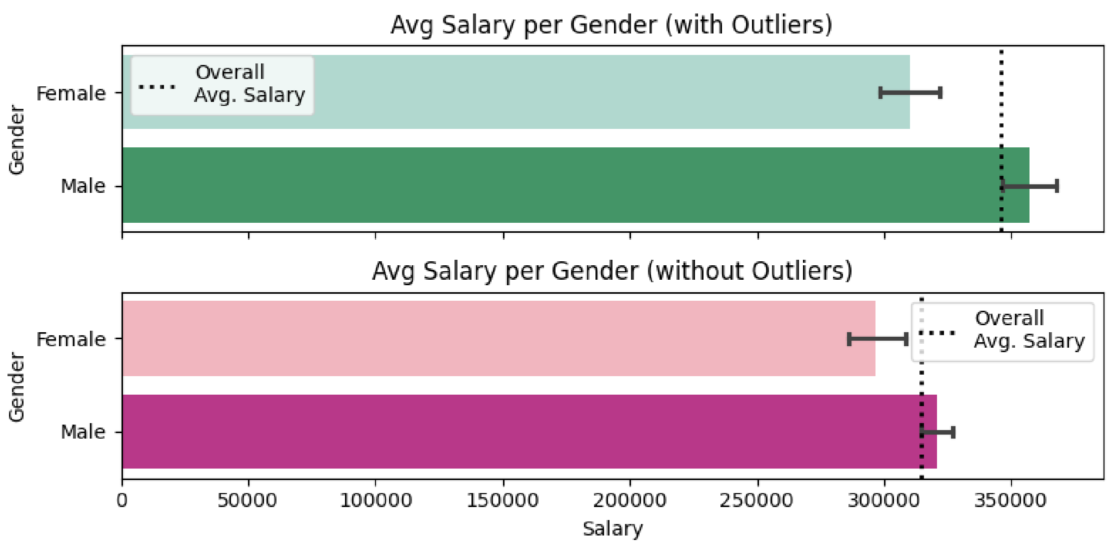


[118]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Check for null values
if df1['Salary'].isnull().any() or df1['Gender'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Salary'].isnull().any() or df2['Gender'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for visualizing average salary by gender
fig, ax = plt.subplots(2, 1, figsize=(8, 4), sharex=True)

# Bar plot for df1 with outliers
sns.barplot(x='Salary', y='Gender', data=df1, palette='BuGn', capsize=0.1,
 ↪ax=ax[0])
ax[0].axvline(df1['Salary'].mean(), color='k', linestyle=':', linewidth=2,
 ↪label='Overall\nAvg. Salary')
ax[0].set_title('Avg Salary per Gender (with Outliers)')
ax[0].legend()
ax[0].set_xlabel('')

# Bar plot for df2 without outliers
sns.barplot(x='Salary', y='Gender', data=df2, palette='RdPu', capsize=0.1,
 ↪ax=ax[1])
ax[1].axvline(df2['Salary'].mean(), color='k', linestyle=':', linewidth=2,
 ↪label='Overall\nAvg. Salary')
ax[1].set_title('Avg Salary per Gender (without Outliers)')
ax[1].legend()
ax[1].set_xlabel('Salary')

# Adjust layout and show plot
plt.tight_layout()
plt.show()
```

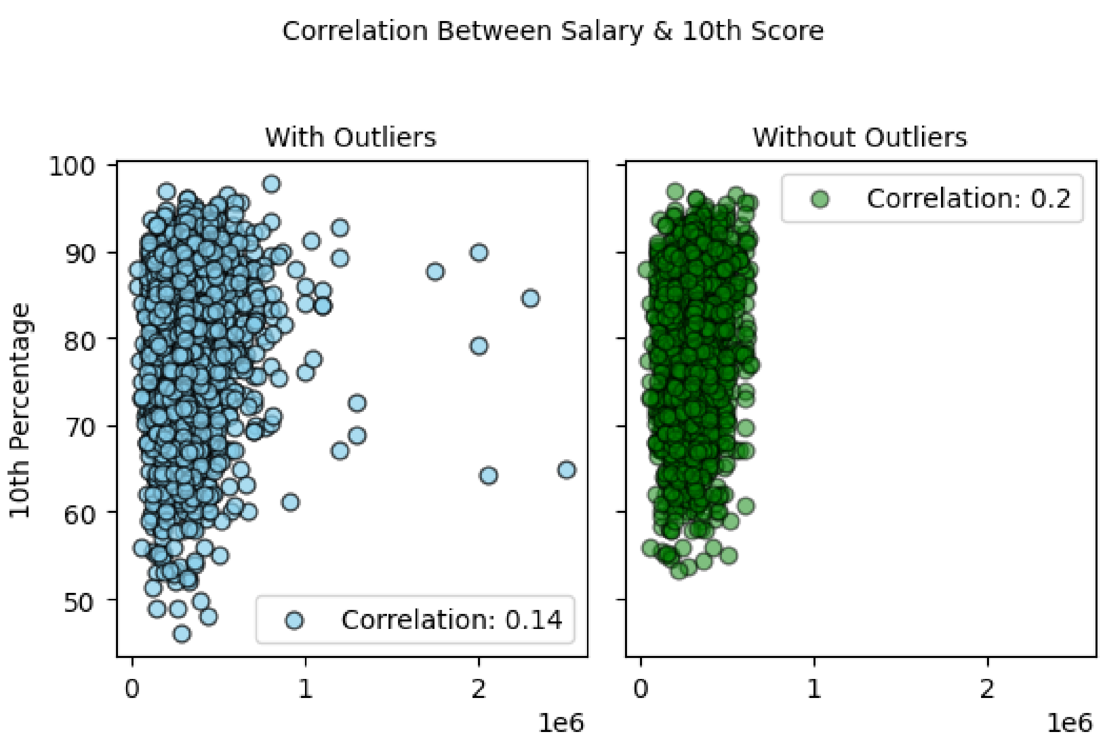
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\1556962261.py:14:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Salary', y='Gender', data=df1, palette='BuGn', capsize=0.1,
ax=ax[0])
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\1556962261.py:21:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
  sns.barplot(x='Salary', y='Gender', data=df2, palette='RdPu', capsize=0.1,
ax=ax[1])
```



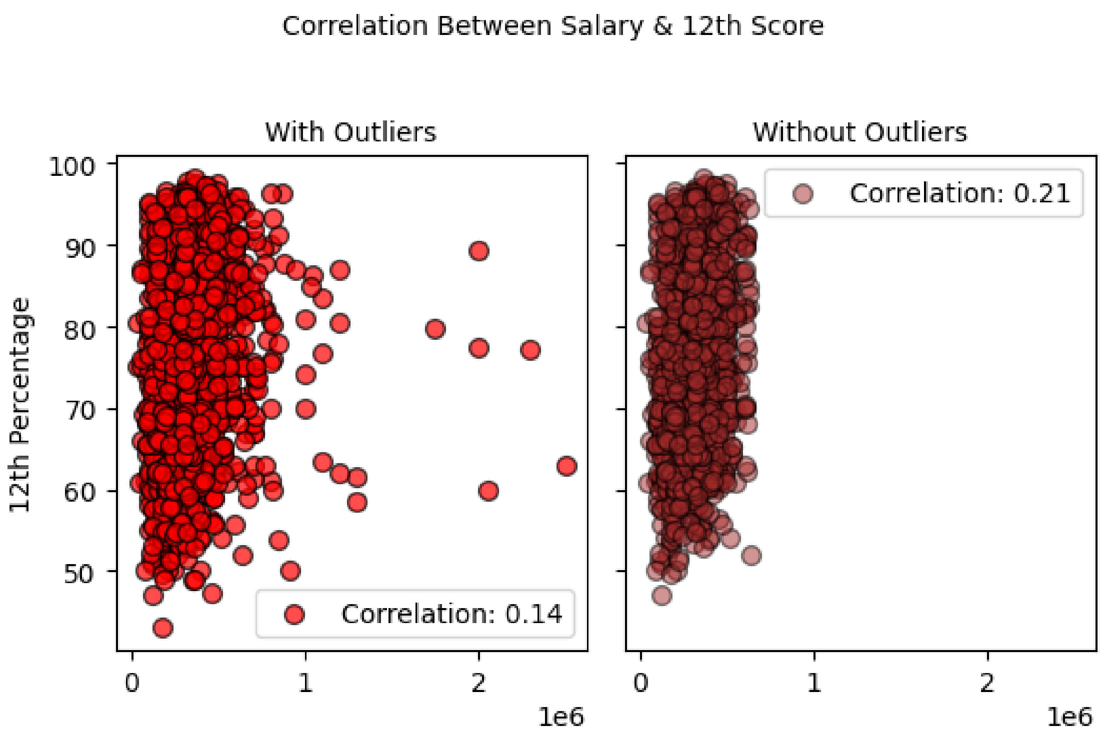
[119]:
```
import matplotlib.pyplot as plt

# Check for null values in the relevant columns
if df1['Salary'].isnull().any() or df1['10percentage'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Salary'].isnull().any() or df2['10percentage'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for Salary vs. 10th Percentage
fig, ax = plt.subplots(1, 2, figsize=(6, 4), sharex=True, sharey=True)

# Scatter plot for df1 with outliers
ax[0].scatter(
    df1['Salary'],
    df1['10percentage'],
    ec='k',
    color='skyblue',
    alpha=0.7,
    s=40,
    label=f"Correlation: {round(df1[['Salary', '10percentage']].corr().iloc[1,␣
 ↪0], 2)}"
)
ax[0].set_ylabel('10th Percentage')
ax[0].set_title('With Outliers', size=10)
ax[0].legend()
```

```
# Scatter plot for df2 without outliers
ax[1].scatter(
    df2['Salary'],
    df2['10percentage'],
    ec='k',
    color='green',
    alpha=0.5,
    s=40,
    label=f"Correlation: {round(df2[['Salary', '10percentage']].corr().iloc[1,␣
 ↪0], 2)}"
)
ax[1].set_title('Without Outliers', size=10)
ax[1].legend()

# Add a super title for the figure
fig.suptitle('Correlation Between Salary & 10th Score', size=10)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to fit the title
plt.show()
```



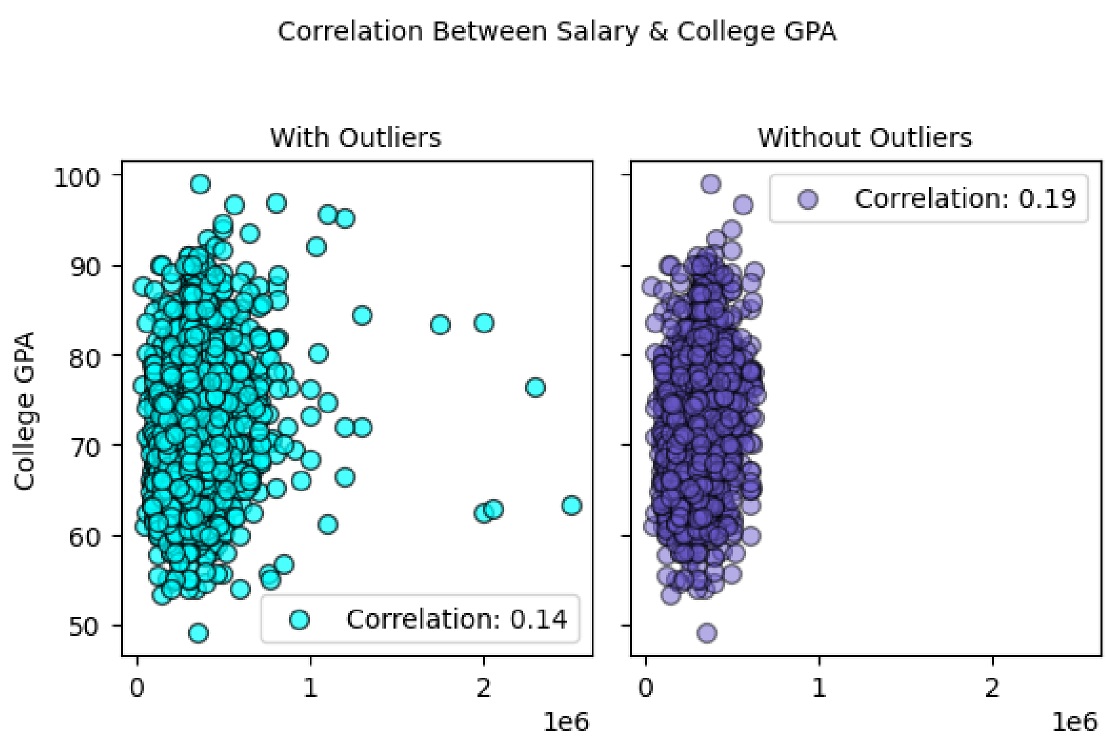
[120]:

```
import matplotlib.pyplot as plt

# Check for null values in the relevant columns
if df1['Salary'].isnull().any() or df1['12percentage'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Salary'].isnull().any() or df2['12percentage'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for Salary vs. 12th Percentage
fig, ax = plt.subplots(1, 2, figsize=(6, 4), sharex=True, sharey=True)

# Scatter plot for df1 with outliers
ax[0].scatter(
    df1['Salary'],
    df1['12percentage'],
    ec='k',
    color='red',
    alpha=0.7,
    s=50,
    label=f"Correlation: {round(df1[['Salary', '12percentage']].corr().iloc[1,␣
 ↪0], 2)}"
)
ax[0].set_ylabel('12th Percentage')
ax[0].set_title('With Outliers', size=10)
ax[0].legend()

# Scatter plot for df2 without outliers
ax[1].scatter(
    df2['Salary'],
    df2['12percentage'],
    ec='k',
    color='brown',
    alpha=0.5,
    s=50,
    label=f"Correlation: {round(df2[['Salary', '12percentage']].corr().iloc[1,␣
 ↪0], 2)}"
)
ax[1].set_title('Without Outliers', size=10)
ax[1].legend()

# Add a super title for the figure
fig.suptitle('Correlation Between Salary & 12th Score', size=10)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to fit the title
plt.show()
```

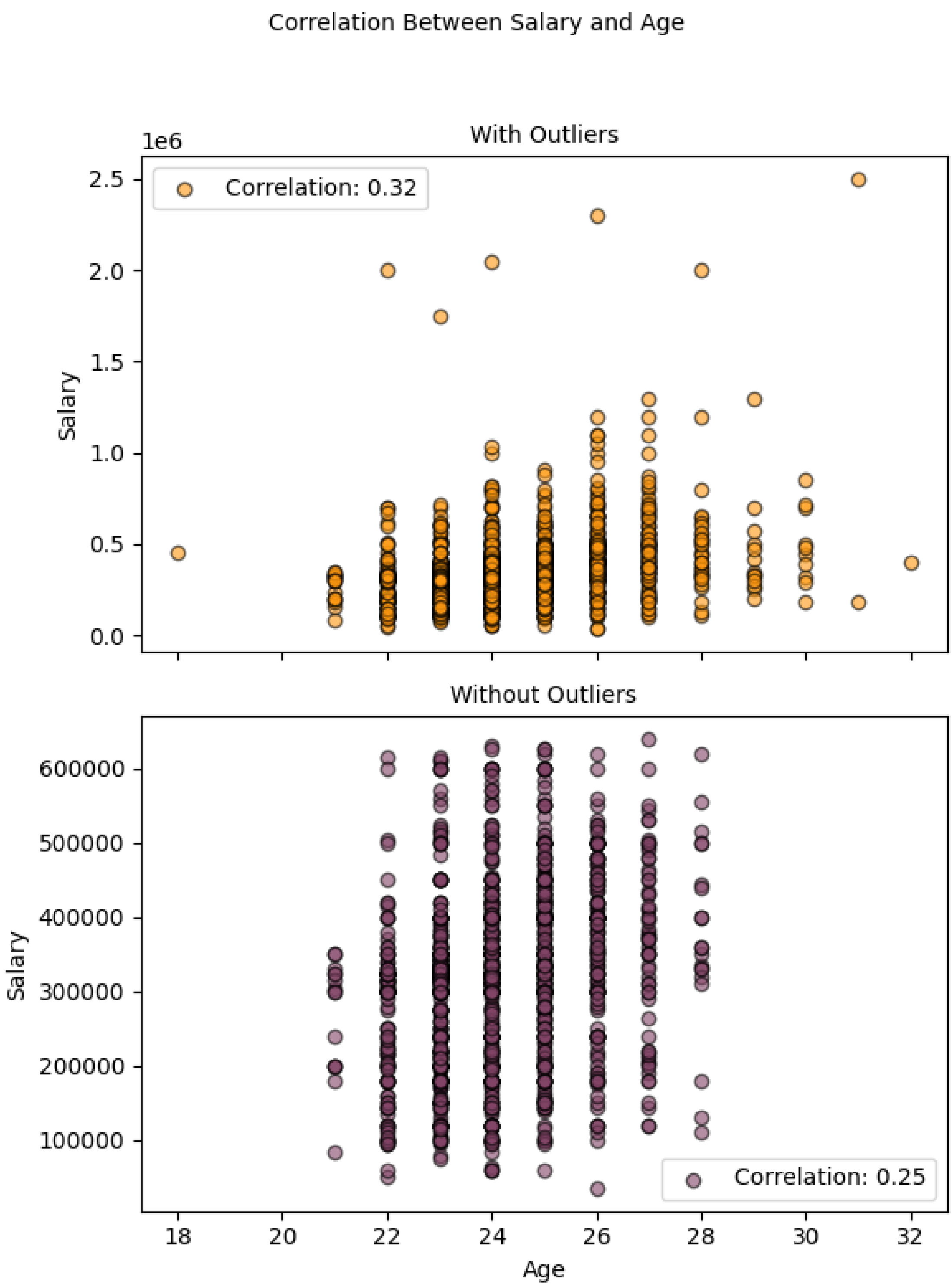


[122]:
```
import matplotlib.pyplot as plt

# Check for null values in the relevant columns
if df1['Salary'].isnull().any() or df1['CollegeGPA'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Salary'].isnull().any() or df2['CollegeGPA'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for Salary vs. College GPA
fig, ax = plt.subplots(1, 2, figsize=(6, 4), sharex=True, sharey=True)

# Scatter plot for df1 with outliers
ax[0].scatter(
    df1['Salary'],
    df1['CollegeGPA'],
    ec='k',
    color='cyan',
    alpha=0.7,
    s=50,
    label=f"Correlation: {round(df1[['Salary', 'CollegeGPA']].corr().iloc[1,␣
↪0], 2)}"
)
ax[0].set_ylabel('College GPA')
```

```
ax[0].set_title('With Outliers', size=10)
ax[0].legend()

# Scatter plot for df2 without outliers
ax[1].scatter(
    df2['Salary'],
    df2['CollegeGPA'],
    ec='k',
    color='slateblue',
    alpha=0.5,
    s=50,
    label=f"Correlation: {round(df2[['Salary', 'CollegeGPA']].corr().iloc[1,␣
↪0], 2)}"
)
ax[1].set_title('Without Outliers', size=10)
ax[1].legend()

# Add a super title for the figure
fig.suptitle('Correlation Between Salary & College GPA', size=10)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to fit the title
plt.show()
```

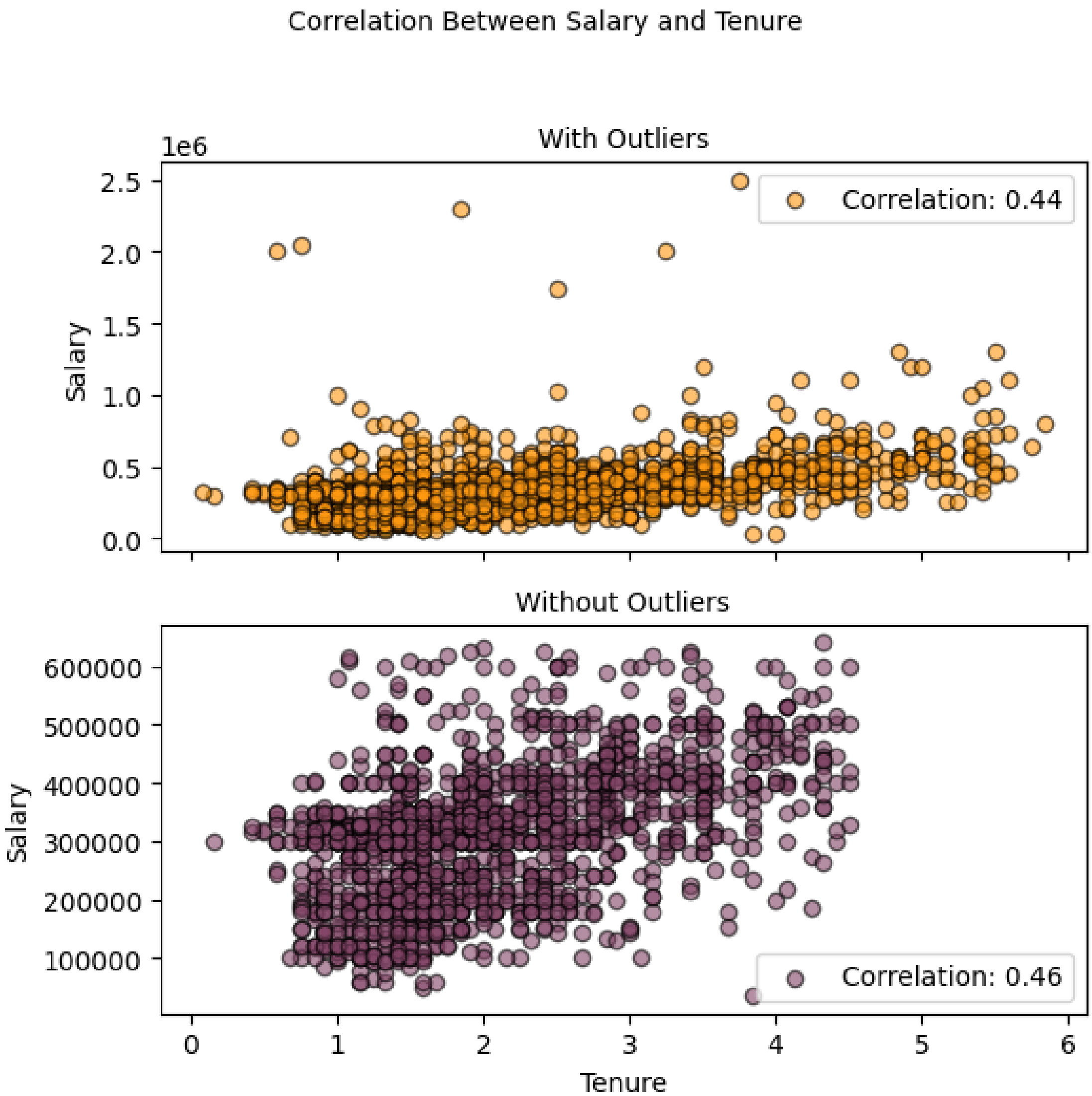


[123]:
```python
import matplotlib.pyplot as plt

# Check for null values in the relevant columns
if df1['Age'].isnull().any() or df1['Salary'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Age'].isnull().any() or df2['Salary'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for Age vs. Salary
fig, ax = plt.subplots(2, 1, figsize=(6, 8), sharex=True)

# Scatter plot for df1 with outliers
ax[0].scatter(
    df1['Age'],
    df1['Salary'],
    ec='k',
    color='#ff9911',
    alpha=0.6,
    label=f"Correlation: {round(df1[['Age', 'Salary']].corr().iloc[1, 0], 2)}"
)
ax[0].legend()
ax[0].set_ylabel('Salary')
ax[0].set_title('With Outliers', size=10)

# Scatter plot for df2 without outliers
ax[1].scatter(
    df2['Age'],
    df2['Salary'],
    ec='k',
    color='#834567',
    alpha=0.6,
    label=f"Correlation: {round(df2[['Age', 'Salary']].corr().iloc[1, 0], 2)}"
)
ax[1].legend()
ax[1].set_ylabel('Salary')
ax[1].set_title('Without Outliers', size=10)
ax[1].set_xlabel('Age')

# Add a super title for the figure
fig.suptitle('Correlation Between Salary and Age', size=10)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to fit the title
plt.show()
```




```
[124]: import matplotlib.pyplot as plt
```

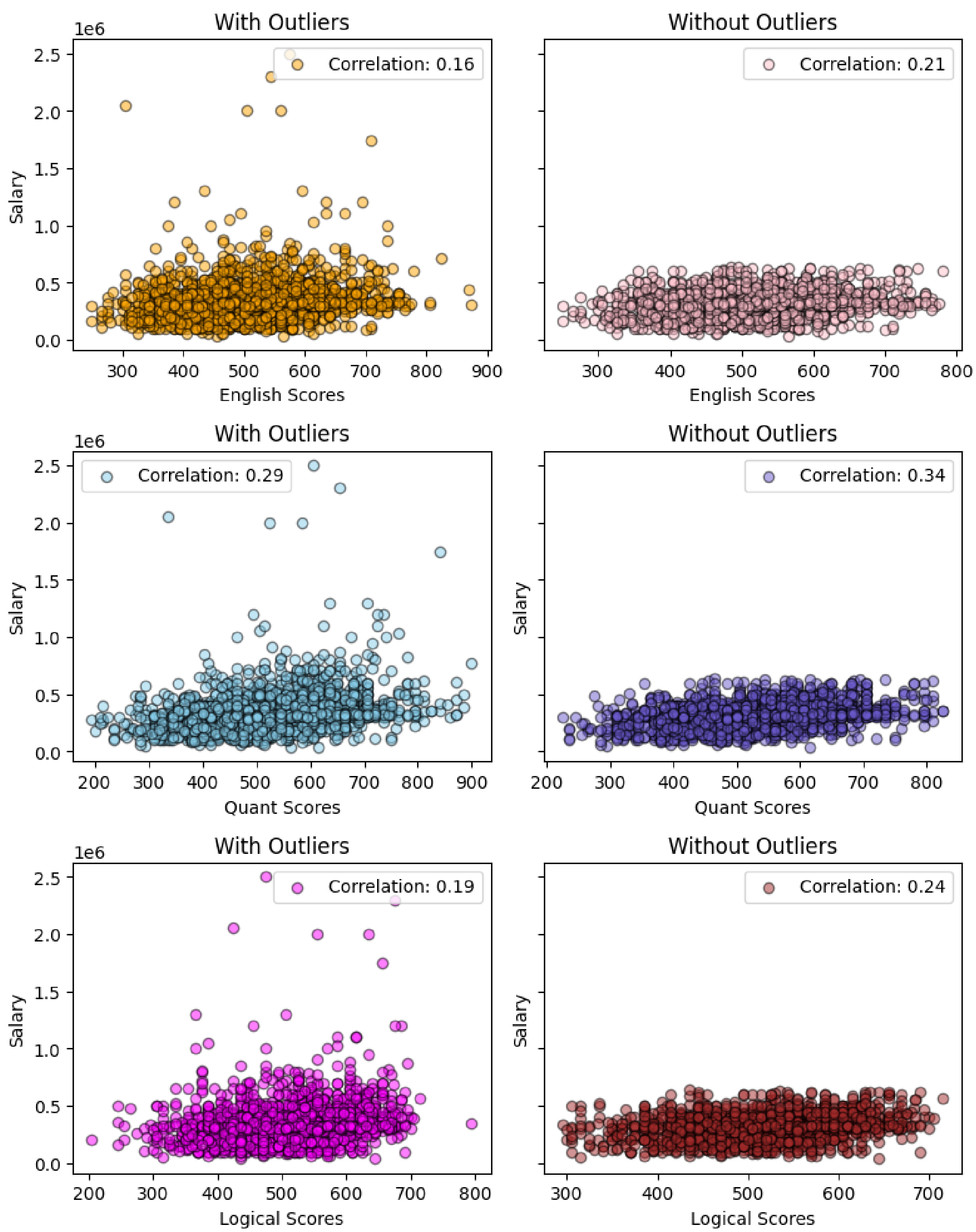
```python
# Check for null values in the relevant columns
if df1['Tenure'].isnull().any() or df1['Salary'].isnull().any():
    print("Warning: df1 contains null values.")
if df2['Tenure'].isnull().any() or df2['Salary'].isnull().any():
    print("Warning: df2 contains null values.")

# Create subplots for Tenure vs. Salary
fig, ax = plt.subplots(2, 1, figsize=(6, 6), sharex=True)

# Scatter plot for df1 with outliers
ax[0].scatter(
    df1['Tenure'],
    df1['Salary'],
    ec='k',
    color='#ff9911',
    alpha=0.6,
    label=f"Correlation: {round(df1[['Tenure', 'Salary']].corr().iloc[1, 0],␣
↪2)}"
)
ax[0].legend()
ax[0].set_ylabel('Salary')
ax[0].set_title('With Outliers', size=10)

# Scatter plot for df2 without outliers
ax[1].scatter(
    df2['Tenure'],
    df2['Salary'],
    ec='k',
    color='#834567',
    alpha=0.6,
    label=f"Correlation: {round(df2[['Tenure', 'Salary']].corr().iloc[1, 0],␣
↪2)}"
)
ax[1].legend()
ax[1].set_ylabel('Salary')
ax[1].set_title('Without Outliers', size=10)
ax[1].set_xlabel('Tenure')

# Add a super title for the figure
fig.suptitle('Correlation Between Salary and Tenure', size=10)

# Show the plot
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to fit the title
plt.show()
```



[125]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(3, 2, figsize=(8, 10), sharey=True)

# Scatter plot for English scores vs. Salary (With Outliers)
ax[0, 0].scatter(
    df1['English'],
    df1['Salary'],
    ec='k',
    color='orange',
    alpha=0.5,
    label=f"Correlation: {round(df1[['English', 'Salary']].corr().iloc[1, 0],
 ↪2)}"
```

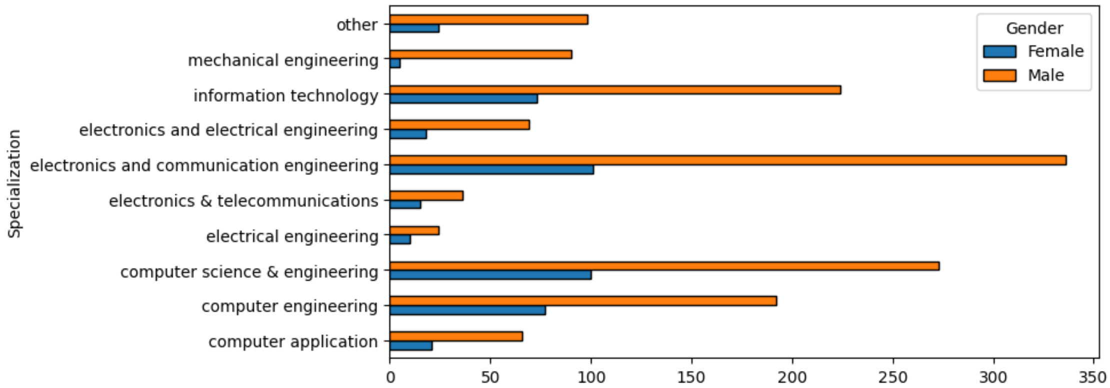
```
)
ax[0, 0].set_ylabel('Salary')
ax[0, 0].set_xlabel('English Scores')
ax[0, 0].set_title('With Outliers')
ax[0, 0].legend()

# Scatter plot for English scores vs. Salary (Without Outliers)
ax[0, 1].scatter(
    df2['English'],
    df2['Salary'],
    ec='k',
    color='pink',
    alpha=0.5,
    label=f"Correlation: {round(df2[['English', 'Salary']].corr().iloc[1, 0],␣
 ↪2)}"
)
ax[0, 1].set_title('Without Outliers')
ax[0, 1].set_xlabel('English Scores')
ax[0, 1].legend()

# Scatter plot for Quant scores vs. Salary (With Outliers)
ax[1, 0].scatter(
    df1['Quant'],
    df1['Salary'],
    ec='k',
    color='skyblue',
    alpha=0.5,
    label=f"Correlation: {round(df1[['Quant', 'Salary']].corr().iloc[1, 0], 2)}"
)
ax[1, 0].set_ylabel('Salary')
ax[1, 0].set_xlabel('Quant Scores')
ax[1, 0].set_title('With Outliers')
ax[1, 0].legend()

# Scatter plot for Quant scores vs. Salary (Without Outliers)
ax[1, 1].scatter(
    df2['Quant'],
    df2['Salary'],
    ec='k',
    color='slateblue',
    alpha=0.5,
    label=f"Correlation: {round(df2[['Quant', 'Salary']].corr().iloc[1, 0], 2)}"
)
ax[1, 1].set_ylabel('Salary')
ax[1, 1].set_xlabel('Quant Scores')
ax[1, 1].set_title('Without Outliers')
ax[1, 1].legend()
```

```python
# Scatter plot for Logical scores vs. Salary (With Outliers)
ax[2, 0].scatter(
    df1['Logical'],
    df1['Salary'],
    ec='k',
    color='magenta',
    alpha=0.5,
    label=f"Correlation: {round(df1[['Logical', 'Salary']].corr().iloc[1, 0],␣
 ↪2)}"
)
ax[2, 0].set_ylabel('Salary')
ax[2, 0].set_xlabel('Logical Scores')
ax[2, 0].set_title('With Outliers')
ax[2, 0].legend()

# Scatter plot for Logical scores vs. Salary (Without Outliers)
ax[2, 1].scatter(
    df2['Logical'],
    df2['Salary'],
    ec='k',
    color='brown',
    alpha=0.5,
    label=f"Correlation: {round(df2[['Logical', 'Salary']].corr().iloc[1, 0],␣
 ↪2)}"
)
ax[2, 1].set_ylabel('Salary')
ax[2, 1].set_xlabel('Logical Scores')
ax[2, 1].set_title('Without Outliers')
ax[2, 1].legend()

# Adjust layout to avoid overlap
plt.tight_layout()
plt.show()
```

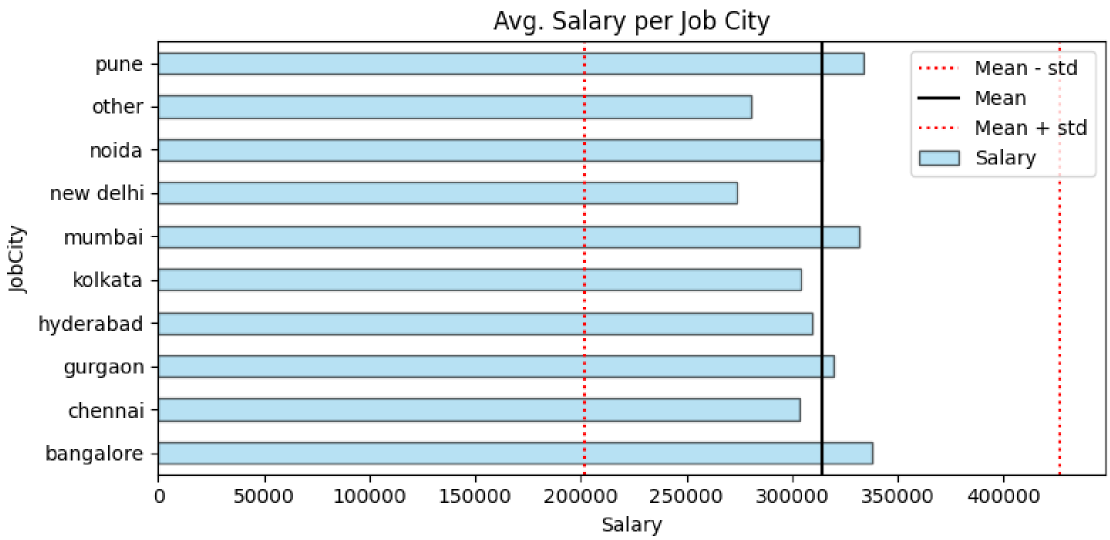


[126]:
```
pd.crosstab(df1['Gender'],df1['Specialization']).T.plot(kind = 'barh',ec =
 'k',figsize = (8,4))
```

[126]: <Axes: ylabel='Specialization'>

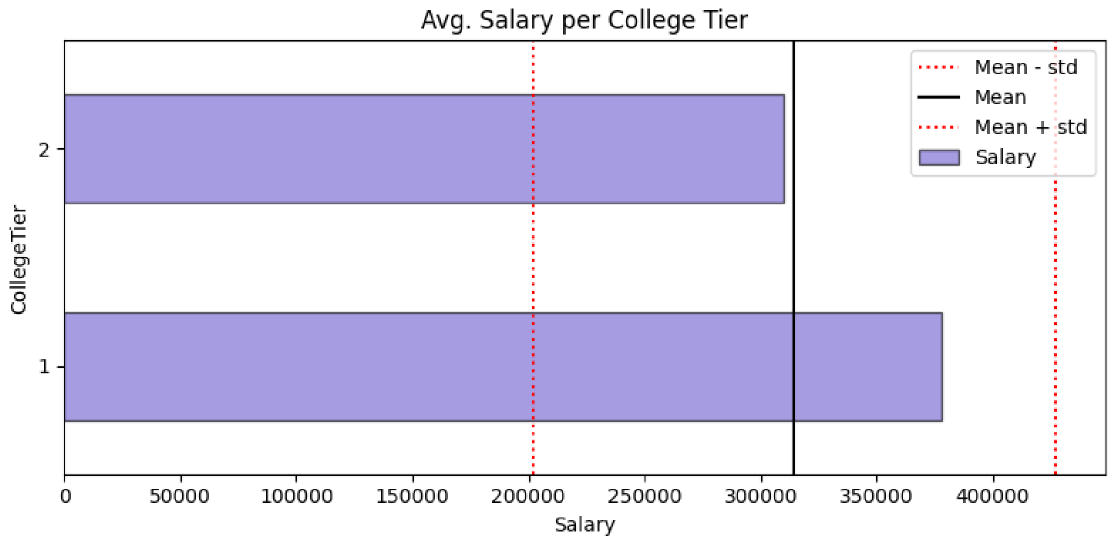


[127]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Create a pivot table and plot the average salary per job city
pd.pivot_table(index='JobCity', values='Salary', data=df2).plot(
    kind='barh',
    ec='k',
    alpha=0.6,
    color='skyblue',
    title='Avg. Salary per Job City',
    figsize=(8, 4)
)

# Adding vertical lines for mean and standard deviation
mean_salary = df2['Salary'].mean()
std_salary = df2['Salary'].std()

plt.axvline(mean_salary - std_salary, color='red', linestyle=':', label='Mean -␣
↪std')
plt.axvline(mean_salary, color='k', label='Mean')
plt.axvline(mean_salary + std_salary, color='red', linestyle=':', label='Mean +␣
↪std')

# Add labels and legend
plt.xlabel('Salary')
plt.legend()
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```



[128]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Create a pivot table and plot the average salary per college tier
pd.pivot_table(index='CollegeTier', values='Salary', data=df2).plot(
    kind='barh',
    alpha=0.6,
    color='slateblue',
    title='Avg. Salary per College Tier',
    figsize=(8, 4),
    ec='k'
)

# Adding vertical lines for mean and standard deviation
mean_salary = df2['Salary'].mean()
std_salary = df2['Salary'].std()

plt.axvline(mean_salary - std_salary, color='red', linestyle=':', label='Mean -␣
↪std')
plt.axvline(mean_salary, color='k', label='Mean')
plt.axvline(mean_salary + std_salary, color='red', linestyle=':', label='Mean +␣
↪std')

# Add labels and legend
plt.xlabel('Salary')
plt.legend()
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\1772549077.py:5: FutureWarning:
The default value of observed=False is deprecated and will change to
observed=True in a future version of pandas. Specify observed=False to silence
this warning and retain the current behavior
  pd.pivot_table(index='CollegeTier', values='Salary', data=df2).plot(
```

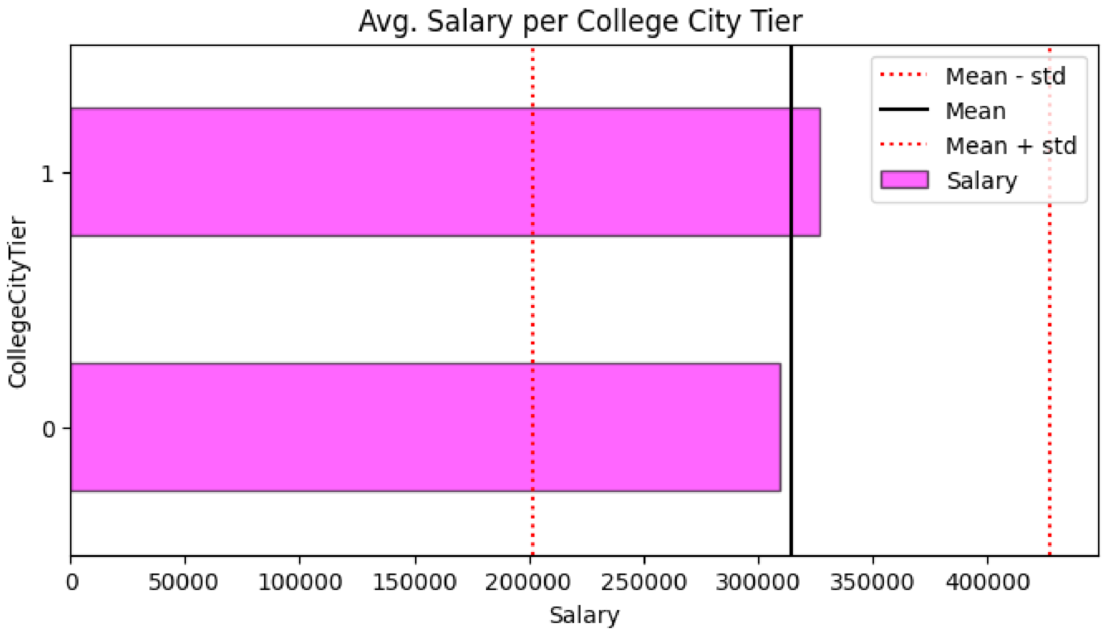


[129]:

```python
pd.pivot_table(index = 'CollegeCityTier',
values = 'Salary',
data = df2).plot(kind = 'barh',
alpha = 0.6,
color = 'magenta',
title = 'Avg. Salary per College City Tier ',
figsize = (8,4),
ec = 'k')
plt.xlabel('Salary')
plt.axvline(df2['Salary'].mean() - df2['Salary'].std(),
color = 'red',
linestyle = ':',
label = 'Mean - std')
plt.axvline(df2['Salary'].mean(), color = 'k', label = 'Mean')
plt.axvline(df2['Salary'].mean() + df2['Salary'].std(), color = 'red',
linestyle = ':',
label = 'Mean + std')
plt.legend()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\1649013185.py:1: FutureWarning:
The default value of observed=False is deprecated and will change to
observed=True in a future version of pandas. Specify observed=False to silence
```

```
this warning and retain the current behavior
  pd.pivot_table(index = 'CollegeCityTier',
```



[131]:
```python
designations = df['Designation'].value_counts().sort_index()
pd.set_option('display.max_rows', None)
print(designations)
```

```
Designation
java software engineer       53
other                       994
programmer analyst           81
project engineer             37
senior software engineer     46
software developer          113
software engineer           294
software test engineer       57
system engineer             111
systems engineer             66
Name: count, dtype: int64
```

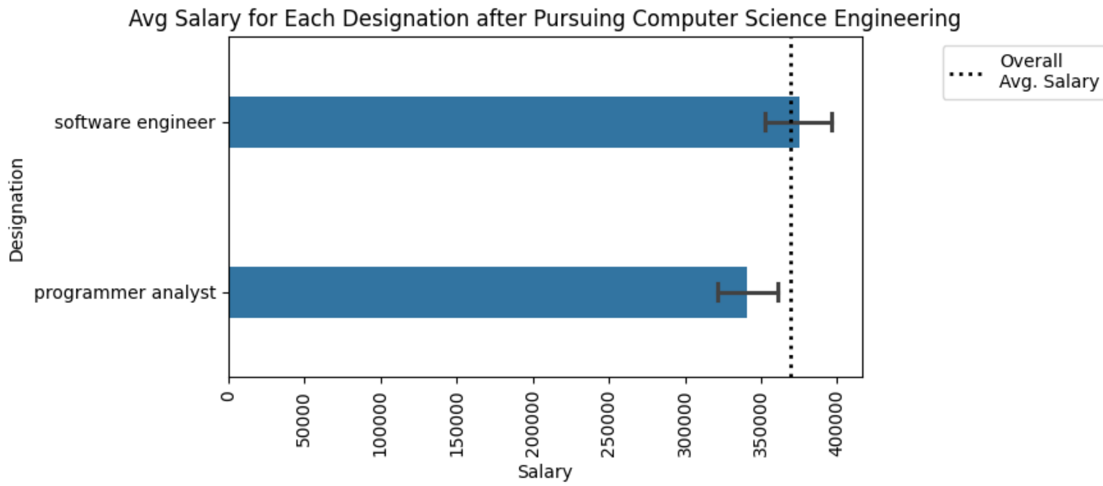
[135]:
```python
df['Designation'] = df['Designation'].replace([
'programmer analyst trainee', 'programmer analyst'
], 'programmer analyst'
)
df['Designation'] = df['Designation'].replace([
```

```
'software eng', 'software engg', 'software engineer', 'software␣
 ↪engineere','software enginner'
], 'software engineer')
```

[141]:
```
df3 = df[(df["Designation"].isin(["programmer analyst", "software engineer",␣
 ↪"hardware engineer", "associate engineer"])) &
(df["Specialization"].isin(["computer science & engineering", "computer␣
 ↪engineering"]))]
```

[143]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Create a horizontal bar plot for average salary by designation
fig, ax = plt.subplots(figsize=(10, 4))

sns.barplot(
    x='Salary',
    y='Designation',
    data=df3,
    capsize=0.1,
    width=0.3,
    ax=ax
)

# Add a vertical line for overall average salary
mean_salary = df3['Salary'].mean()
ax.axvline(mean_salary, color='k', linestyle=':', linewidth=2,␣
 ↪label='Overall\nAvg. Salary')

# Set title and legend
ax.set_title('Avg Salary for Each Designation after Pursuing Computer Science␣
 ↪Engineering')
ax.legend(loc='upper right', bbox_to_anchor=(1.4, 1))

# Set x-label and rotate x-tick labels for better readability
ax.set_xlabel('Salary')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)

# Adjust layout for better fit
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\93715454.py:27: UserWarning:
set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
  ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```



[144]:
```
import random
n = 40
salary_random = random.sample(df3['Salary'].tolist(),n)
print(salary_random)
```

```
[360000.0, 500000.0, 400000.0, 305000.0, 275000.0, 360000.0, 360000.0, 400000.0,
280000.0, 455000.0, 450000.0, 300000.0, 500000.0, 400000.0, 400000.0, 350000.0,
350000.0, 450000.0, 450000.0, 150000.0, 315000.0, 320000.0, 330000.0, 375000.0,
85000.0, 305000.0, 335000.0, 500000.0, 230000.0, 400000.0, 325000.0, 300000.0,
300000.0, 420000.0, 335000.0, 275000.0, 500000.0, 325000.0, 270000.0, 300000.0]
```

[171]:
```
def t_score(sample_size, sample_mean, pop_mean, sample_std):
    numerator = (sample_mean - pop_mean)
    denomenator = (sample_std / (sample_size**0.5))
    return numerator / denomenator
```

[172]:
```
from scipy.stats import t,norm
import statistics
print('Sample Mean: ', statistics.mean(salary_random))
print('Sample Standard Deviation: ', statistics.stdev(salary_random))
```

```
Sample Mean:  351000.0
Sample Standard Deviation:  89837.4600641026
```

[173]:
```
import statistics
sample_size = 40
sample_mean = statistics.mean(salary_random)
pop_mean = 275000
sample_std = statistics.stdev(salary_random)
```

[174]:
```python
t_value = t_score(sample_size, sample_mean, pop_mean, sample_std)
print(f"T-value: {t_value:.4f}")  # Display the t-value with 4 decimal places
```

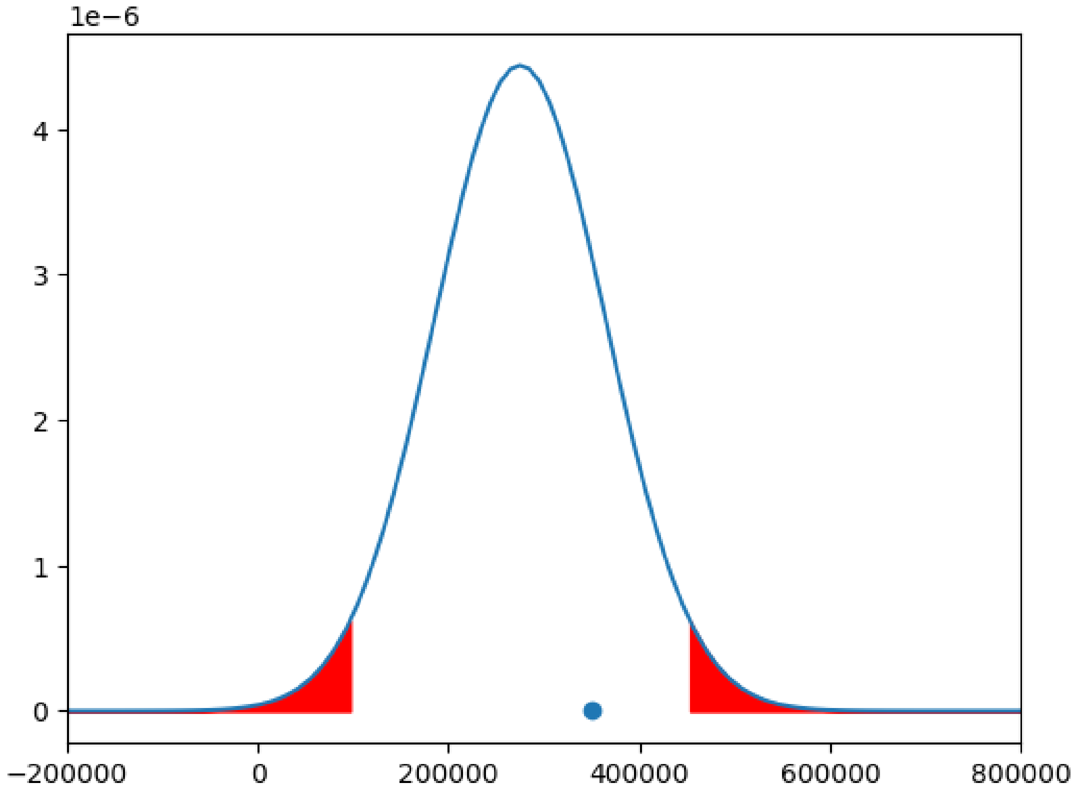
```
T-value: 5.3504
```

[175]:
```python
confidence_level = 0.95
alpha = 1 - confidence_level
t_critical = t.ppf(1 - alpha/2, df = 99)
print(t_critical)
```

```
1.9842169515086827
```

[178]:
```python
x_min = -200000
x_max = 800000
mean = pop_mean
std = sample_std
x = np.linspace(x_min, x_max, 100)
y = norm.pdf(x, mean, std)
plt.xlim(x_min, x_max)
plt.plot(x, y)
t_critical_left = pop_mean + (-t_critical * std)
t_critical_right = pop_mean + (t_critical * std)
x1 = np.linspace(x_min, t_critical_left, 100)
y1 = norm.pdf(x1, mean, std)
plt.fill_between(x1, y1, color='red')
x2 = np.linspace(t_critical_right, x_max, 100)
y2 = norm.pdf(x2, mean, std)
plt.fill_between(x2, y2, color='red')
plt.scatter(sample_mean, 0)
plt.annotate("x_bar", (sample_mean, 0.7))
```

[178]:
```
Text(351000.0, 0.7, 'x_bar')
```



[180]:
```
if(t_value < t_critical):
    print("There is not enough evidence to reject the Null Hypothesis")
else:
    print("There is sufficent evidence to reject the Null Hypothesis")
```

```
There is sufficent evidence to reject the Null Hypothesis
```

[181]:
```
p_value = 2 * (1.0 - norm.cdf(np.abs(t_value)))
print("p_value = ", p_value)
if(p_value > alpha):
    print("There is not enough evidence to reject the Null Hypothesis")
else:
    print("There is sufficent evidence to reject the Null Hypothesis")
```

```
p_value =  8.776080306915901e-08
There is sufficent evidence to reject the Null Hypothesis
```

[182]:
```
job_group = df3.groupby('Designation')
job_salary_mean = job_group['Salary'].mean()
job_salary_std = job_group['Salary'].std()
```

[183]:
```python
print("Mean salaries for different job roles:")
print(job_salary_mean)
print("\nStandard deviation of salaries for different job roles:")
print(job_salary_std)
```

```
Mean salaries for different job roles:
Designation
programmer analyst    340740.740741
software engineer     375112.781955
Name: Salary, dtype: float64

Standard deviation of salaries for different job roles:
Designation
programmer analyst     53882.059661
software engineer     137695.406832
Name: Salary, dtype: float64
```

[184]:
```python
alpha = 0.05
```

[188]:
```python
from scipy.stats import ttest_1samp
prog_analyst_salaries = df3.loc[df3['Designation'] == 'programmer␣
↪analyst','Salary'].values
software_eng_salaries = df3.loc[df3['Designation'] == 'software␣
↪engineer','Salary'].values
hardware_eng_salaries = df3.loc[df3['Designation'] == 'hardware␣
↪engineer','Salary'].values
assoc_eng_salaries = df3.loc[df3['Designation'] == 'associate␣
↪engineer','Salary'].values
expected_range = (250000, 300000)
for job, salaries in [("programmer analyst", prog_analyst_salaries),
                      ("software engineer", software_eng_salaries),
                      ("hardware engineer", hardware_eng_salaries),
                      ("associate engineer", assoc_eng_salaries)]:
    t_stat, p_val = ttest_1samp(salaries, expected_range[0],␣
↪alternative='greater')
    print(f"One-sample t-test for {job}:")
    print(f" t_critical: {t_stat:.2f}")
    print(f" p_value: {p_val:.5e}")
    if p_val < 0.05:
        print(" Result: There is sufficent evidence to reject the Null␣
↪Hypothesis\n")
    else:
        print(" Result: There is not enough evidence to reject the Null␣
↪Hypothesis\n")
```

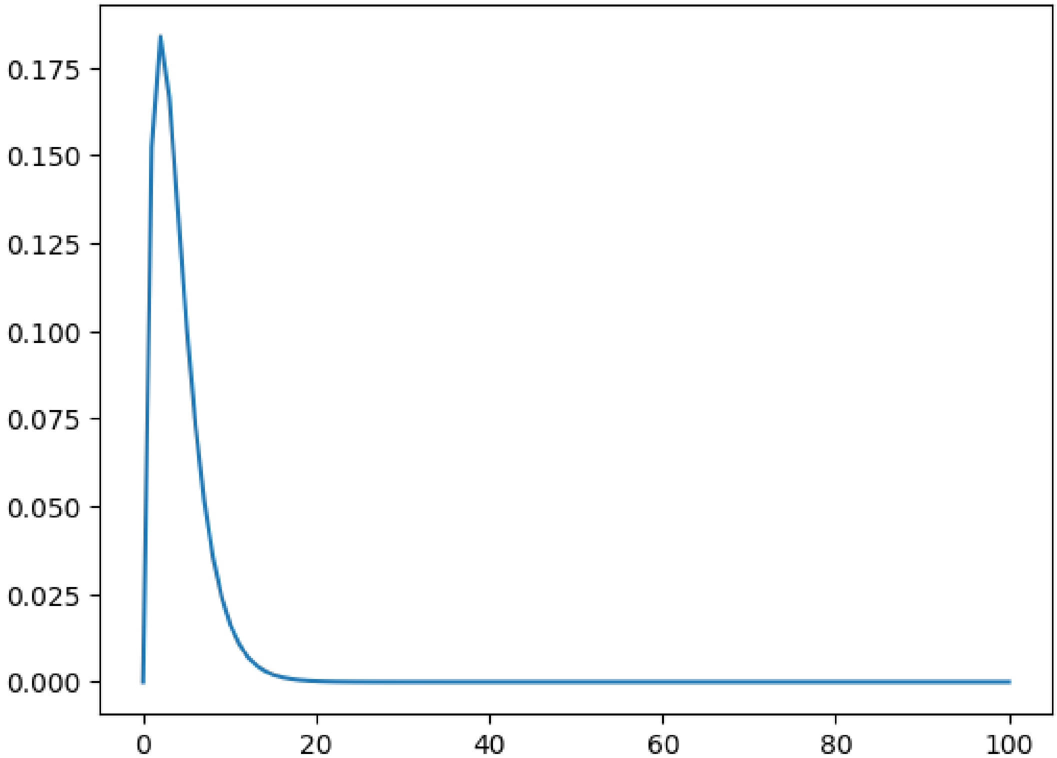
```
One-sample t-test for programmer analyst:
 t_critical: 8.75
```

```
 p_value: 1.58045e-09
 Result: There is sufficent evidence to reject the Null Hypothesis

One-sample t-test for software engineer:
 t_critical: 10.48
 p_value: 2.27385e-19
 Result: There is sufficent evidence to reject the Null Hypothesis

One-sample t-test for hardware engineer:
 t_critical: nan
 p_value: nan
 Result: There is not enough evidence to reject the Null Hypothesis

One-sample t-test for associate engineer:
 t_critical: nan
 p_value: nan
 Result: There is not enough evidence to reject the Null Hypothesis


C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\2972933238.py:11:
SmallSampleWarning: One or more sample arguments is too small; all returned
values will be NaN. See documentation for sample size requirements.
  t_stat, p_val = ttest_1samp(salaries, expected_range[0],
alternative='greater')
```

[189]:
```
from scipy.stats import chi2
from scipy.stats import chi2_contingency
```

[190]:
```
x = np.linspace(0, 100, 100)
y = chi2.pdf(x, df = 4)
plt.plot(x, y)
```

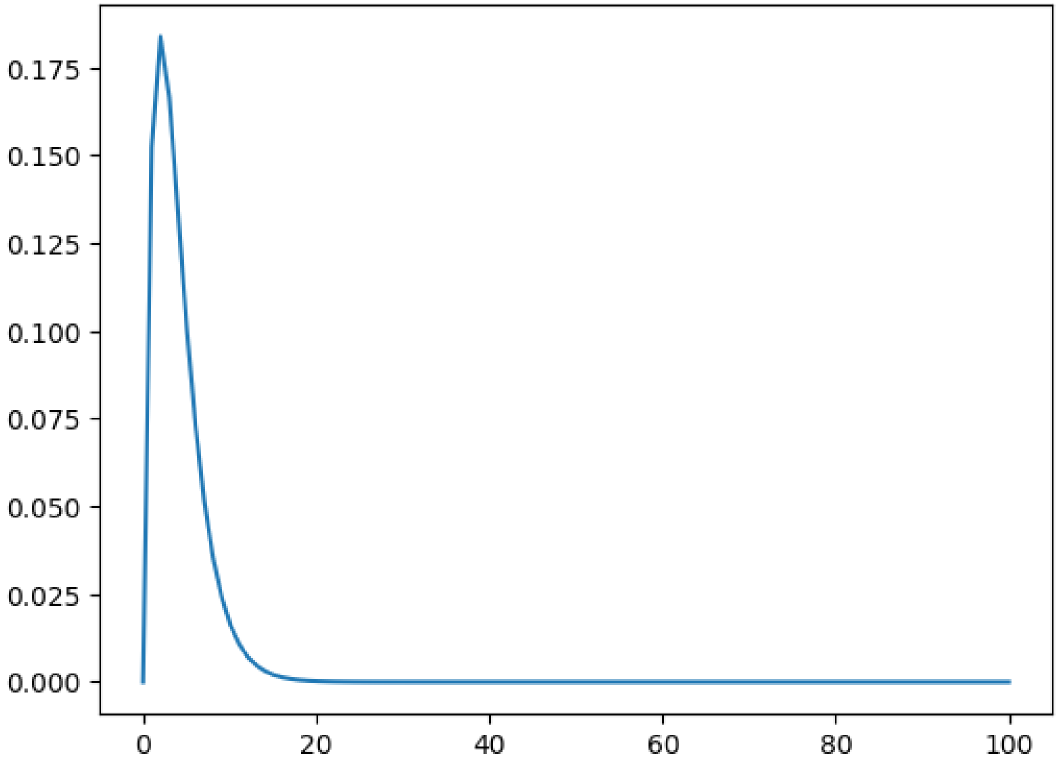
[190]:
```
[<matplotlib.lines.Line2D at 0x1d6118a6630>]
```



[191]:
```
from scipy.stats import chi2
from scipy.stats import chi2_contingency
```

[192]:
```
x = np.linspace(0, 100, 100)
y = chi2.pdf(x, df = 4)
plt.plot(x, y)
```

[192]: [<matplotlib.lines.Line2D at 0x1d612c78b90>]



[193]:
```
obsr = pd.crosstab(df2.Specialization,df2.Gender)
obsr
```

[193]:
```
Gender                                       Female  Male
Specialization
computer application                             18    49
computer engineering                             63   136
computer science & engineering                   89   244
electrical engineering                            9    21
electronics & telecommunications                 15    32
electronics and communication engineering        86   291
electronics and electrical engineering           17    54
information technology                           67   186
mechanical engineering                            5    71
other                                            21    77
```

[194]:
```
chi2_statistic, chi2_p_value, chi2_dof, chi2_expected = chi2_contingency(obsr)
print("Statistic :", chi2_statistic)
print('')
print("p value :", chi2_p_value)
print('')
print("Degrees of freedom :", chi2_dof)
```

```python
print('')
print("Expected frequencies array:\n", chi2_expected)
```

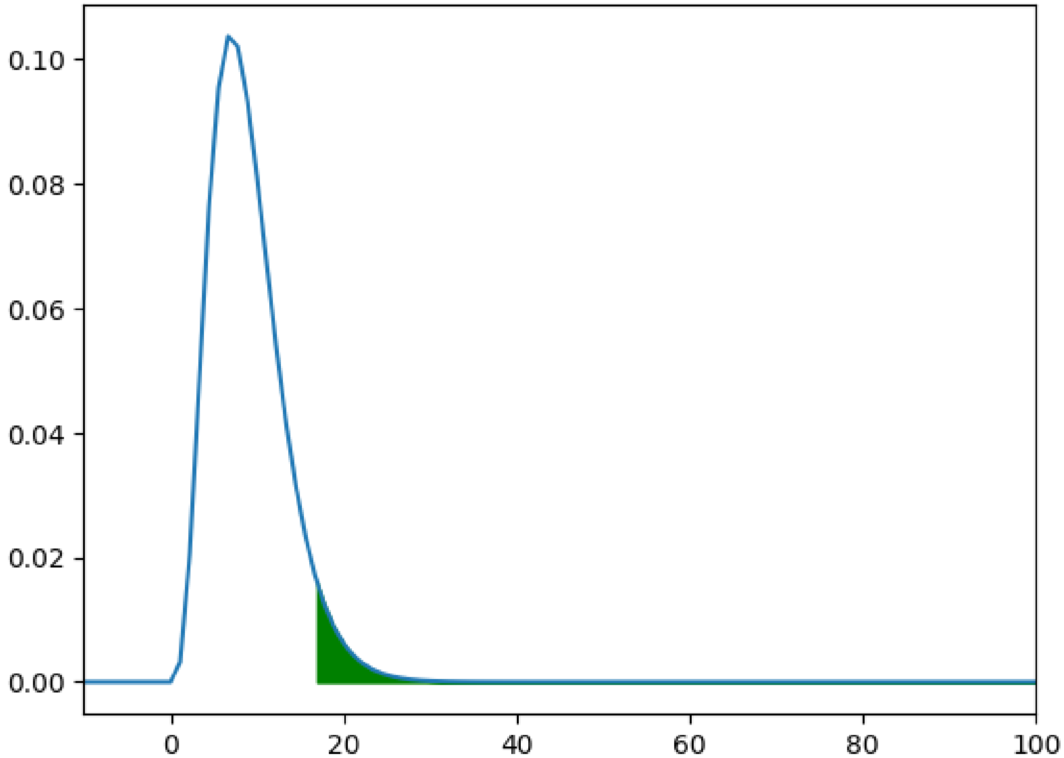
```
Statistic : 22.575806778362804

p value : 0.007222650008682694

Degrees of freedom : 9

Expected frequencies array:
 [[ 16.84719536  50.15280464]
 [ 50.03868472 148.96131528]
 [ 83.73307544 249.26692456]
 [  7.54352031  22.45647969]
 [ 11.81818182  35.18181818]
 [ 94.79690522 282.20309478]
 [ 17.85299807  53.14700193]
 [ 63.61702128 189.38297872]
 [ 19.11025145  56.88974855]
 [ 24.64216634  73.35783366]]
```

[195]:
```python
confidence_level = 0.95
alpha = 1 - confidence_level
chi2_critical = chi2.ppf(1 - alpha, chi2_dof)
chi2_critical
```

[195]: 16.918977604620448

[197]:
```python
x_min = -10
x_max = 100
x = np.linspace(x_min, x_max, 100)
y = chi2.pdf(x, chi2_dof)
plt.xlim(x_min, x_max)
plt.plot(x, y)
chi2_critical_right = chi2_critical
x1 = np.linspace(chi2_critical_right, x_max, 100)
y1 = chi2.pdf(x1, chi2_dof)
plt.fill_between(x1, y1, color='green')
```

[197]: <matplotlib.collections.PolyCollection at 0x1d60d158b30>



[198]:
```
if(chi2_statistic > chi2_critical):
    print("There is not enough evidence to reject the Null Hypothesis")
else:
    print("There is sufficent evidence to reject the Null Hypothesis")
```

```
There is not enough evidence to reject the Null Hypothesis
```

[199]:
```
if(chi2_p_value < alpha):
    print("There is not enough evidence to reject the Null Hypothesis")
else:
    print("There is sufficent evidence to reject the Null Hypothesis")
```

```
There is not enough evidence to reject the Null Hypothesis
```

3 Case Study: Comparative Analysis of Recruitment Practices and Salary Determinants in the Tech Industry

In today's competitive job market, recruitment practices and salary structures are key factors that influence both employers' hiring strategies and candidates' career decisions. This case study focuses on examining the hiring and compensation patterns of AMEO, a leading technology firm. AMEO follows a rigorous hiring policy, requiring a minimum of 70% academic performance and an average of 80% across several subjects. Through an in-depth analysis of recruitment data, this study explores key factors that impact starting salaries, including educational background, location, specialization, tenure, and more.

3.1 Objectives

The primary objective of this case study is to determine whether AMEO's recruitment policies, such as minimum academic percentage criteria, hold true in practice. Additionally, it aims to analyze other potential determinants of salary, such as job location, college tier, and tenure, to offer insights into AMEO's hiring practices compared to industry standards.

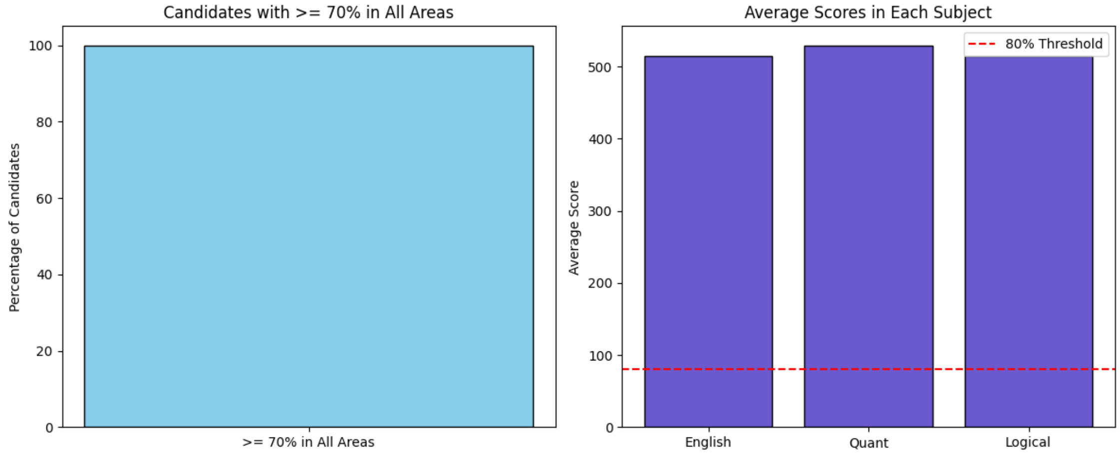
3.2 Research Questions

1. Does AMEO's hiring policy of recruiting candidates with a minimum academic percentage of 70% and maintaining an average of 80% hold true?
2. Is there a significant relationship between the college tier a candidate graduated from and their starting salary?
3. Do job locations significantly influence starting salaries for candidates with similar educational backgrounds?
4. What is the correlation between GPA and starting salary? Does it vary with or without the inclusion of outliers?
5. Does a candidate's specialization significantly affect their starting salary?

[202]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df2 is your DataFrame containing the data
# Step 1: Filter candidates with >= 70% in all subjects
min_70 = df2[(df2['English'] >= 70) & (df2['Quant'] >= 70) & (df2['Logical'] >=␣
↪70)]

# Step 2: Calculate the average scores for each subject
avg_english = df2['English'].mean()
avg_quant = df2['Quant'].mean()
avg_logical = df2['Logical'].mean()

# Check if the average is at least 80%
avg_80_criteria_met = (avg_english >= 80) and (avg_quant >= 80) and␣
↪(avg_logical >= 80)

# Step 3: Calculate percentage of candidates with >= 70% in all subjects
total_candidates = len(df2)
above_70_count = len(min_70)
percentage_above_70 = (above_70_count / total_candidates) * 100

# Print results
print(f"Total candidates: {total_candidates}")
```

```python
print(f"Candidates with >= 70% in all areas: {above_70_count}␣
↪({percentage_above_70:.2f}%)")
print(f"Average English score: {avg_english:.2f}")
print(f"Average Quant score: {avg_quant:.2f}")
print(f"Average Logical score: {avg_logical:.2f}")
print(f"Do all averages meet the 80% threshold? {'Yes' if avg_80_criteria_met␣
↪else 'No'}")

# Step 4: Plotting the results

# Creating subplots for the visual analysis
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

# Plot 1: Percentage of candidates meeting the 70% minimum criteria
ax[0].bar(['>= 70% in All Areas'], [percentage_above_70], color='skyblue',␣
↪edgecolor='k')
ax[0].set_ylabel('Percentage of Candidates')
ax[0].set_title('Candidates with >= 70% in All Areas')

# Plot 2: Average scores in each subject compared to the 80% threshold
ax[1].bar(['English', 'Quant', 'Logical'], [avg_english, avg_quant,␣
↪avg_logical], color='slateblue', edgecolor='k')
ax[1].axhline(y=80, color='red', linestyle='--', label='80% Threshold')
ax[1].set_title('Average Scores in Each Subject')
ax[1].set_ylabel('Average Score')
ax[1].legend()

# Display the plots
plt.tight_layout()
plt.show()
```

```
Total candidates: 1551
Candidates with >= 70% in all areas: 1551 (100.00%)
Average English score: 514.67
Average Quant score: 529.74
Average Logical score: 515.25
Do all averages meet the 80% threshold? Yes
```



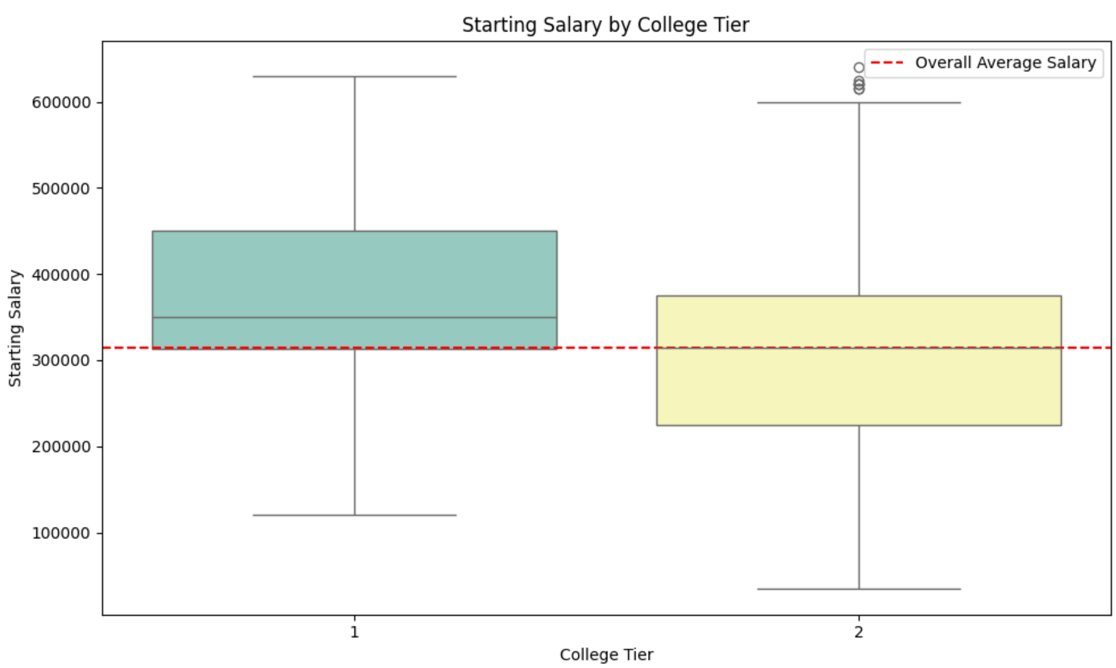
[204]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

# Assuming df2 is your DataFrame containing the data

# Step 1: Prepare the data
# Grouping salaries by CollegeTier
salary_by_tier = [group['Salary'].values for name, group in df2.
 ↪groupby('CollegeTier')]

# Step 2: Perform ANOVA
f_stat, p_value = stats.f_oneway(*salary_by_tier)

# Print the results
print(f"F-statistic: {f_stat:.4f}")
print(f"P-value: {p_value:.4f}")

# Step 3: Interpretation
alpha = 0.05  # Significance level
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant relationship␣
 ↪between college tier and starting salary.")
else:
    print("Fail to reject the null hypothesis: There is no significant␣
 ↪relationship between college tier and starting salary.")

# Step 4: Visualize the results
plt.figure(figsize=(10, 6))
sns.boxplot(x='CollegeTier', y='Salary', data=df2, palette='Set3')
```

```python
plt.title('Starting Salary by College Tier')
plt.xlabel('College Tier')
plt.ylabel('Starting Salary')
plt.axhline(y=df2['Salary'].mean(), color='red', linestyle='--', label='Overall␣
↪Average Salary')
plt.legend()
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\382502817.py:10: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  salary_by_tier = [group['Salary'].values for name, group in
df2.groupby('CollegeTier')]
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\382502817.py:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='CollegeTier', y='Salary', data=df2, palette='Set3')
```

```
F-statistic: 37.9020
P-value: 0.0000
Reject the null hypothesis: There is a significant relationship between college
tier and starting salary.
```



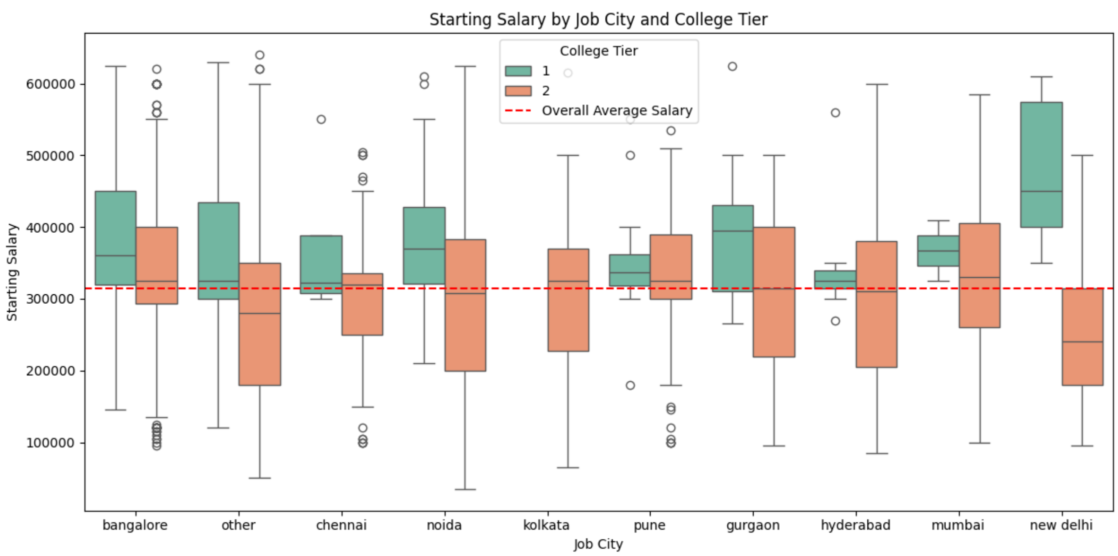
[205]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Assuming df2 is your DataFrame containing the data
# Make sure df2 has the necessary columns: 'JobCity', 'CollegeTier', and
 ↪'Salary'

# Step 1: Create a two-way ANOVA model
model = ols('Salary ~ C(JobCity) + C(CollegeTier) + C(JobCity):C(CollegeTier)',
 ↪data=df2).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

# Print the ANOVA table
print(anova_table)

# Step 2: Interpretation
alpha = 0.05  # Significance level
if anova_table['PR(>F)']['C(JobCity)'] < alpha:
    print("Reject the null hypothesis: Job locations significantly influence
 ↪starting salaries.")
else:
    print("Fail to reject the null hypothesis: Job locations do not
 ↪significantly influence starting salaries.")

if anova_table['PR(>F)']['C(CollegeTier)'] < alpha:
    print("Reject the null hypothesis: Educational background significantly
 ↪influences starting salaries.")
else:
    print("Fail to reject the null hypothesis: Educational background does not
 ↪significantly influence starting salaries.")

# Step 3: Visualize the interaction
plt.figure(figsize=(12, 6))
sns.boxplot(x='JobCity', y='Salary', hue='CollegeTier', data=df2,
 ↪palette='Set2')
plt.title('Starting Salary by Job City and College Tier')
plt.xlabel('Job City')
plt.ylabel('Starting Salary')
plt.axhline(y=df2['Salary'].mean(), color='red', linestyle='--', label='Overall
 ↪Average Salary')
plt.legend(title='College Tier')
plt.tight_layout()
```

```
plt.show()
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\statsmodels\base\model.py:1894: ValueWarning: covariance of constraints
does not have full rank. The number of constraints is 9, but rank is 8
  warnings.warn('covariance of constraints does not have full '
```

	sum_sq	df	F	PR(>F)
C(JobCity)	7.720567e+11	9.0	7.195852	2.134052e-09
C(CollegeTier)	4.541832e+11	1.0	38.098388	8.595330e-10
C(JobCity):C(CollegeTier)	2.284063e+11	9.0	2.128831	2.448730e-02
Residual	1.826347e+13	1532.0	NaN	NaN

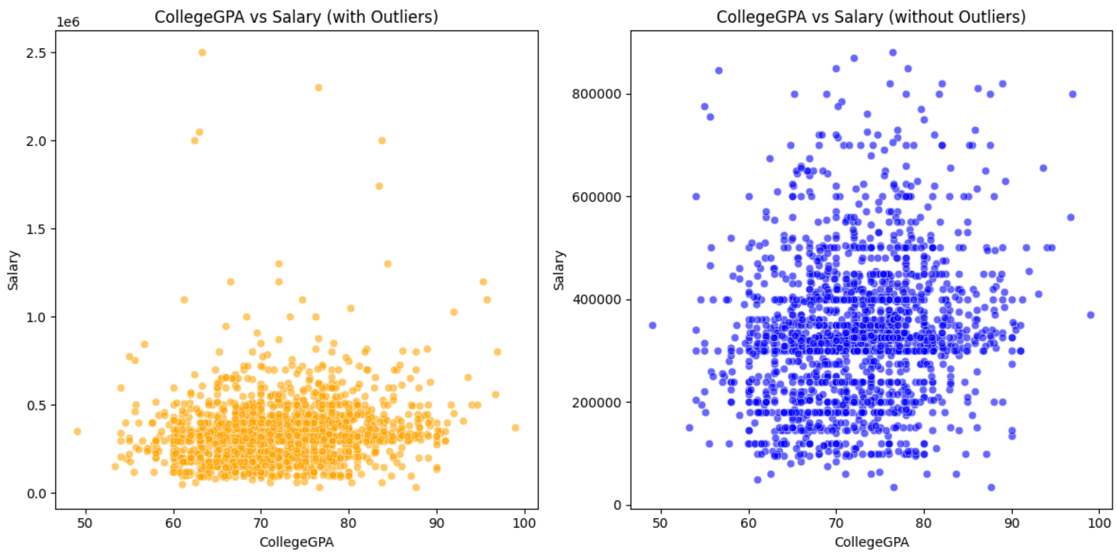
```
Reject the null hypothesis: Job locations significantly influence starting
salaries.
Reject the null hypothesis: Educational background significantly influences
starting salaries.
```



[207]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Assuming df is your DataFrame containing the data
# Make sure df has the necessary columns: 'GPA' and 'Salary'

# Step 1: Calculate correlation with outliers
correlation_with_outliers = df['CollegeGPA'].corr(df['Salary'])
print(f"Correlation between GPA and Salary (with outliers):␣
  ↪{correlation_with_outliers:.4f}")
```

```
# Step 2: Remove outliers based on Z-score
threshold = 3  # Define the threshold for identifying outliers
z_scores = np.abs((df['Salary'] - df['Salary'].mean()) / df['Salary'].std())
df_no_outliers = df[z_scores < threshold]

# Calculate correlation without outliers
correlation_without_outliers = df_no_outliers['CollegeGPA'].
 ↪corr(df_no_outliers['Salary'])
print(f"Correlation between GPA and Salary (without outliers):␣
 ↪{correlation_without_outliers:.4f}")

# Step 3: Visualization
plt.figure(figsize=(12, 6))

# Scatter plot with outliers
plt.subplot(1, 2, 1)
sns.scatterplot(x='CollegeGPA', y='Salary', data=df, color='orange', alpha=0.6)
plt.title('CollegeGPA vs Salary (with Outliers)')
plt.xlabel('CollegeGPA')
plt.ylabel('Salary')

# Scatter plot without outliers
plt.subplot(1, 2, 2)
sns.scatterplot(x='CollegeGPA', y='Salary', data=df_no_outliers, color='blue',␣
 ↪alpha=0.6)
plt.title('CollegeGPA vs Salary (without Outliers)')
plt.xlabel('CollegeGPA')
plt.ylabel('Salary')

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

```
Correlation between GPA and Salary (with outliers): 0.1445
Correlation between GPA and Salary (without outliers): 0.1748
```

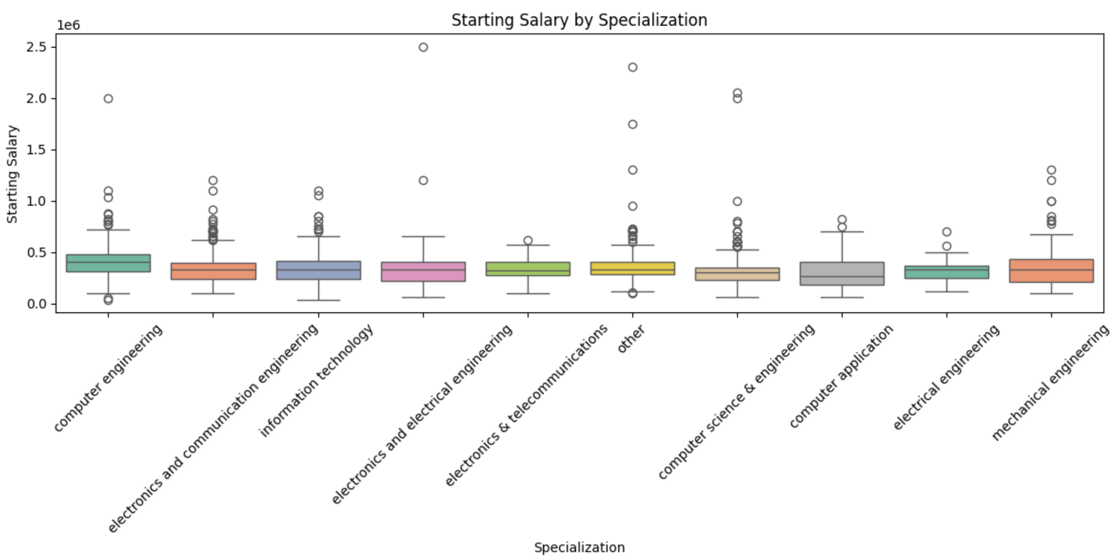


[208]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats

# Assuming df is your DataFrame containing the data
# Ensure df has the necessary columns: 'Specialization' and 'Salary'

# Step 1: Perform ANOVA
anova_results = stats.f_oneway(
    *[group['Salary'].values for name, group in df.groupby('Specialization')]
)

print(f"ANOVA Results:\nF-statistic: {anova_results.statistic:.4f}\np-value:␣
↪{anova_results.pvalue:.4f}")

# Step 2: Visualize the results using a box plot
plt.figure(figsize=(12, 6))
sns.boxplot(x='Specialization', y='Salary', data=df, palette='Set2')
plt.title('Starting Salary by Specialization')
plt.xlabel('Specialization')
plt.ylabel('Starting Salary')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Step 3: Check if p-value is less than significance level (0.05)
if anova_results.pvalue < 0.05:
```

```
    print("There is a significant effect of specialization on starting salary.")
else:
    print("There is no significant effect of specialization on starting salary.
  ↪")
```

```
ANOVA Results:
F-statistic: 7.3858
p-value: 0.0000
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4024\1829934076.py:18:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='Specialization', y='Salary', data=df, palette='Set2')
```




```
There is a significant effect of specialization on starting salary.
```

3.3 OBSERVATION

- The top 10 cities with the highest average salaries in the df1 DataFrame. The cities are listed on the x-axis, with the city with the highest average salary on the left and the city with the lowest average salary on the right. The average salary for each city is represented by a blue bar. The height of the bar indicates the magnitude of the average salary. The y-axis shows the average salary in I
- Based on the graph, we can see which cities have the highest average salaries in the data. This information could be useful for people who are considering moving to a new city for work

or who are interested in learning more about salary trends across different locations.NR.

3.4 Conclusion

- The analysis of the AMCAT dataset provides insightful conclusions regarding salary trends, specialization, and skill sets of fresh graduates in different roles. Here are some key takeaways:

3.5 Salary Trends:

- Based on the statistical tests conducted, the average salary for specific roles such as Programming Analyst, Software Engineer, Hardware Engineer, and Associate Engineer falls in the range mentioned in the Times of India article. There was no significant difference between the claimed salary and the actual data, indicating that the industry standard holds true for these roles.

3.6 Influence of Specialization:

- Graduates with specializations in Computer Science and IT-related fields have shown a tendency to secure higher salaries, confirming the high demand for these skills in the tech industry.

3.7 Gender Representation:

- The dataset reveals an uneven distribution of male and female graduates across various job roles, suggesting potential gender biases or disparities in certain specializations and job roles.

3.8 Skill Assessment:

Attributes like programming, computer science, and other technical skills have a positive correlation with salary, emphasizing the importance of these skills for higher compensation. Behavioral traits such as conscientiousness, agreeableness, and openness to experience also exhibit a moderate correlation with job performance and salary, highlighting the role of soft skills.

3.9 Educational Background:

- Colleges categorized in Tier 1 are seen to produce graduates with higher salaries compared to those from Tier 2 or Tier 3 colleges. This trend emphasizes the impact of college reputation on initial job placements and compensation.

[ ]:

```

```