INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Exploratory Data Analysis on AMCAT's 2015 Employment Outcome Data.

About me

Name : Manoj Kumar

With a B.E. in Computer Science and a keen interest in Data Science, Machine Learning, Python Development, and Web Development & Designing, I have a strong foundation in both theory and practice. I also lead the Google Developer Student Club at Info Institute of Engineering, where I mentor and guide fellow tech enthusiasts towards mastering cutting-edge technologies and industry-relevant skills. I have worked on various data science and web development projects, such as laptop price prediction, nearest pub prediction, URL shortener, Covid-19 dashboard, and more. I enjoy learning new things, solving problems, and sharing my knowledge with others. Let's connect and embark on a shared mission to shape a strong tech future, one coder at a time!

- **GitHub : github.com/ManojKumar2920**
- **LinkedIn : linkedin.com/in/manojkumar20**

INNOMATICS RESEARCH LABS

Objective

The aim of this project is to conduct Exploratory Data Analysis (EDA) on the provided dataset, focusing on understanding the data's characteristics, identifying patterns and relationships within the variables, exploring distributions, detecting outliers, and addressing specific research questions related to earnings based on specialization and the relationship between gender and specialization. The insights gained from this analysis will contribute to a deeper understanding of the dataset and inform decision-making processes or further analysis.

Summary of Data

The dataset was released by Aspiring Minds from the Aspiring Mind Employment Outcome 2015 (AMEO). The study is primarily limited only to students with engineering disciplines. The dataset contains the employment outcomes of engineering graduates as dependent variables (Salary, Job Titles, and Job Locations) along with the standardized scores from three different areas – cognitive skills, technical skills and personality skills. The dataset also contains demographic features. The dataset contains around 40 independent variables and 4000 data points. The independent variables are both continuous and categorical in nature. The dataset contains a unique identifier for each candidate. Below mentioned table contains the details for the original dataset.

EDA

Handling Missing Values:

- Identify missing values in the dataset.
- Decide on appropriate strategies for handling missing values (e.g., imputation, deletion).

Data Type Conversion:

- Check the data types of each column.
- Convert data types to their appropriate formats (e.g., converting string/object types to numerical or categorical types).

Removing Duplicates:

- Identify and remove any duplicate rows from the dataset.

Addressing Outliers:

- Identify outliers in numerical variables using statistical methods (e.g., IQR, z-score).
- Decide on appropriate strategies for handling outliers (e.g., capping, transformation, removal).

Data Aggregation:

Aggregate data if necessary (e.g., group by categorical variables and calculate summary statistics).

Encoding Categorical Variables:

Encode categorical variables using techniques such as one-hot encoding or label encoding.

Feature Scaling:

Scale numerical features if required for certain machine learning algorithms (e.g., normalization).

Probability Density Function (PDF):
Plot PDFs for numerical variables to visualize their distributions.

Histograms:
Create histograms to visualize the frequency distribution of numerical variables.

Boxplots:
Generate boxplots to visualize the spread of numerical variables and identify outliers.

Countplots:
Use countplots to understand the frequency distribution of categorical variables.

Stacked Bar Plots:
Analyze relationships between categorical variables using stacked bar plots.

Correlation Analysis:
Calculate and visualize correlations between numerical variables using correlation matrices or heatmaps.

Times of India Claim Evaluation:
Tested the claim regarding earnings based on specialization by analyzing salary data within the dataset.
Conducted statistical tests or visual comparisons to evaluate the validity of the claim.

Gender-Specialization Relationship:
Investigated the relationship between gender and specialization preferences.
Analyzed specialization choices among different genders to assess any gender-based preferences or disparities.

Conclusion

Summarized key findings and insights obtained from the EDA process.
Highlighted significant patterns, relationships, and outliers identified within the dataset.

THANK YOU

INNOMATICS RESEARCH LABS

eda-amcat

February 23, 2024

[57]:
```
PATH = "D:\\Projects\\DS\\EDA AMCAT\\Dataset\\ameo_data.csv"

import pandas as pd

df = pd.read_csv(PATH)

df.head()
```

[57]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	…	ComputerScience	MechanicalEngg	ElectricalEngg	TelecomEngg	CivilEngg	conscientiousness	agreeableness	extraversion	nueroticism	openess_to_experience
0	train	203097	420000.0	6/1/12 0:00	present	senior quality engineer	Bangalore	f	2/19/90 0:00	84.3	…	-1	-1	-1	-1	-1	0.9737	0.8128	0.5269	1.35490	
1	train	579905	500000.0	9/1/13 0:00	present	assistant manager	Indore	m	10/4/89 0:00	85.4	…	-1	-1	-1	-1	-1	-0.7335	0.3789	1.2396	-0.10760	
2	train	810601	325000.0	6/1/14 0:00	present	systems engineer	Chennai	f	8/3/92 0:00	85.0	…	-1	-1	-1	-1	-1	0.2718	1.7109	0.1637	-0.86820	
3	train	267447	1100000.0	7/1/11 0:00	present	senior software engineer	Gurgaon	m	12/5/89 0:00	85.6	…	-1	-1	-1	-1	-1	0.0464	0.3448	-0.3440	-0.40780	
4	train	343523	200000.0	3/1/14 0:00	3/1/15 0:00	get	Manesar	m	2/27/91 0:00	78.0	…	-1	-1	-1	-1	-1	-0.8810	-0.2793	-1.0697	0.09163	

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

[5 rows x 39 columns]
```

0.0.1 Summary Table for Dataset

Variables	Type	Description
ID	UID	A unique ID to identify a candidate
Salary	Continuous	Annual CTC offered to the candidate (in INR)
DOJ	Date	Date of joining the company
DOL	Date	Date of leaving the company
Designation	Categorical	Designation offered in the job
JobCity	Categorical	Location of the job (city)
Gender	Categorical	Candidate's gender
DOB	Date	Date of birth of candidate
10percentage	Continuous	Overall marks obtained in grade 10 examinations
10board	Continuous	The school board whose curriculum the candidate followed in grade 10
12graduation	Date	Year of graduation - senior year high school
12percentage	Continuous	Overall marks obtained in grade 12 examinations
12board	Continuous	The school board whose curriculum the candidate followed in grade 12
CollegeID	NA/ID	Unique ID identifying the college which the candidate attended
CollegeTier	Categorical	Tier of college
Degree	Categorical	Degree obtained/pursued by the candidate
Specialization	Categorical	Specialization pursued by the candidate
CollegeGPA	Continuous	Aggregate GPA at graduation
CollegeCityID	NA/ID	A unique ID to identify the city in which the college is located in
CollegeCityTier	Categorical	The tier of the city in which the college is located
CollegeState	Categorical	Name of States
GraduationYear	Date	Year of graduation (Bachelor's degree)
English	Continuous	Scores in AMCAT English section
Logical	Continuous	Scores in AMCAT Logical section
Quant	Continuous	Scores in AMCAT Quantitative section
Domain	Continuous/Standardized	Scores in AMCAT's domain module
ComputerProgramming	Continuous	Score in AMCAT's Computer programming section
ElectronicsAndSemicon	Continuous	Score in AMCAT's Electronics & Semiconductor Engineering section
ComputerScience	Continuous	Score in AMCAT's Computer Science section

Variables	Type	Description
MechanicalEngg	Continuous	Score in AMCAT's Mechanical Engineering section
ElectricalEngg	Continuous	Score in AMCAT's Electrical Engineering section
TelecomEngg	Continuous	Score in AMCAT's Telecommunication Engineering section
CivilEngg	Continuous	Score in AMCAT's Civil Engineering section
Conscientiousness	Continuous/Standardized	Scores in one of the sections of AMCAT's personality test
Agreeableness	Continuous/Standardized	Scores in one of the sections of AMCAT's personality test
Extraversion	Continuous/Standardized	Scores in one of the sections of AMCAT's personality test
Neuroticism	Continuous/Standardized	Scores in one of the sections of AMCAT's personality test
Openess_to_experience	Continuous/Standardized	Scores in one of the sections of AMCAT's personality test

[58]:
```
df.shape
```

[58]:
```
(3998, 39)
```

[59]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3998 entries, 0 to 3997
Data columns (total 39 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             3998 non-null   object
 1   ID                     3998 non-null   int64
 2   Salary                 3998 non-null   float64
 3   DOJ                    3998 non-null   object
 4   DOL                    3998 non-null   object
 5   Designation            3998 non-null   object
 6   JobCity                3998 non-null   object
 7   Gender                 3998 non-null   object
 8   DOB                    3998 non-null   object
 9   10percentage           3998 non-null   float64
 10  10board                3998 non-null   object
 11  12graduation           3998 non-null   int64
 12  12percentage           3998 non-null   float64
 13  12board                3998 non-null   object
 14  CollegeID              3998 non-null   int64
 15  CollegeTier            3998 non-null   int64
 16  Degree                 3998 non-null   object
```

```
 17  Specialization          3998 non-null   object
 18  collegeGPA              3998 non-null   float64
 19  CollegeCityID           3998 non-null   int64
 20  CollegeCityTier         3998 non-null   int64
 21  CollegeState            3998 non-null   object
 22  GraduationYear          3998 non-null   int64
 23  English                 3998 non-null   int64
 24  Logical                 3998 non-null   int64
 25  Quant                   3998 non-null   int64
 26  Domain                  3998 non-null   float64
 27  ComputerProgramming     3998 non-null   int64
 28  ElectronicsAndSemicon   3998 non-null   int64
 29  ComputerScience         3998 non-null   int64
 30  MechanicalEngg          3998 non-null   int64
 31  ElectricalEngg          3998 non-null   int64
 32  TelecomEngg             3998 non-null   int64
 33  CivilEngg               3998 non-null   int64
 34  conscientiousness       3998 non-null   float64
 35  agreeableness           3998 non-null   float64
 36  extraversion            3998 non-null   float64
 37  nueroticism             3998 non-null   float64
 38  openess_to_experience   3998 non-null   float64
dtypes: float64(10), int64(17), object(12)
memory usage: 1.2+ MB
```

[60]:
```
df.columns
```

[60]:
```
Index(['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'],
      dtype='object')
```

[61]:
```
df.isnull().sum()
```

[61]:
```
Unnamed: 0              0
ID                      0
Salary                  0
DOJ                     0
DOL                     0
Designation             0
JobCity                 0
```

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

[62]:
```
df.duplicated().sum()
```

[62]:
```
0
```

[63]:
```
df = df.drop(columns= ['Unnamed: 0', 'ID','CollegeID','CollegeCityID'])
```

[64]:
```
df['DOL'].replace('present','2015-12-31', inplace = True)
```

[65]:
```
df['DOL'] = pd.to_datetime(df['DOL'], format = 'mixed')
```

[66]:
```
df['DOJ'] = pd.to_datetime(df['DOJ'])
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\1267054188.py:1: UserWarning:
```

```
Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df['DOJ'] = pd.to_datetime(df['DOJ'])
```

[67]:
```
dates = df[(df['DOL'] < df['DOJ'])].shape[0]
print(f'{dates} DOL dates are earlier than DOJ')
```

```
40 DOL dates are earlier than DOJ
```

[68]:
```
df = df.drop(df[~(df['DOL'] > df['DOJ'])].index)
```

[69]:
```
(df['10percentage'] <= 10).sum()
```

[69]: 0

[70]:
```
(df['12percentage'] <= 10).sum()
```

[70]: 0

[71]:
```
(df['collegeGPA'] <= 10).sum()
```

[71]: 12

[72]:
```
df.loc[df['collegeGPA']<=10,'collegeGPA'].index
```

[72]:
```
Index([7, 138, 788, 1419, 1439, 1767, 2151, 2229, 2293, 2662, 2691, 3308],
dtype='int64')
```

[73]:
```
df.loc[df['collegeGPA']<=10,'collegeGPA'] = (df.
 ↪loc[df['collegeGPA']<=10,'collegeGPA']/10)*100
```

[74]:
```
(df['collegeGPA'] <= 10).sum()
```

[74]: 0

[79]:
```
import numpy as np
```

[86]:
```
df = df.drop(columns␣
 ↪=['MechanicalEngg','ElectricalEngg','TelecomEngg','CivilEngg'])
```

[87]:
```
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})
```

[88]:
```
df.head()
```

[88]:
```
      Salary        DOJ        DOL                Designation    JobCity  \
0   420000.0 2012-06-01 2015-12-31   senior quality engineer  Bangalore
1   500000.0 2013-09-01 2015-12-31         assistant manager     Indore
2   325000.0 2014-06-01 2015-12-31          systems engineer    Chennai
3  1100000.0 2011-07-01 2015-12-31  senior software engineer    Gurgaon
4   200000.0 2014-03-01 2015-03-01                       get    Manesar

  Gender           DOB  10percentage                          10board  \
0      f  2/19/90 0:00          84.3  board ofsecondary education,ap
1      m  10/4/89 0:00          85.4                              cbse
2      f   8/3/92 0:00          85.0                              cbse
3      m  12/5/89 0:00          85.6                              cbse
4      m  2/27/91 0:00          78.0                              cbse

   12graduation  …  Quant    Domain  ComputerProgramming  \
0          2007  …    525  0.635979                  NaN
1          2007  …    780  0.960603                  NaN
2          2010  …    370  0.450877                  NaN
3          2007  …    625  0.974396                  NaN
4          2008  …    465  0.124502                  NaN

  ElectronicsAndSemicon ComputerScience  conscientiousness  agreeableness  \
0                     0                0             0.9737         0.8128
1                   466                0            -0.7335         0.3789
2                     0                0             0.2718         1.7109
3                     0                0             0.0464         0.3448
4                   233                0            -0.8810        -0.2793

  extraversion  nueroticism  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 31 columns]
```

[92]:
```
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)
```

[93]:
```
df['Domain'].fillna(df['Domain'].median(), inplace = True)
df['ComputerProgramming'].fillna(df['ComputerProgramming'].median(), inplace =␣
 ↪True)
```

[94]:
```
textual_columns =␣
 ↪['Designation','JobCity','10board','12board','Specialization','CollegeState']
```

[98]:
```
for cols in textual_columns:
    print('Top 10 categories in:', cols)
    print('-'*30)
    print(df[cols].value_counts())
    print('*'*100)
```

```
Top 10 categories in: Designation
------------------------------
Designation
software engineer                    535
software developer                   262
system engineer                      202
programmer analyst                   139
systems engineer                     117
                                    ...
human resources intern                 1
senior quality assurance engineer      1
clerical assistant                     1
delivery software engineer             1
jr. software developer                 1
Name: count, Length: 416, dtype: int64
****************************************************************************************
********************
Top 10 categories in: JobCity
------------------------------
JobCity
Bangalore           1071
Noida                361
Hyderabad            329
Pune                 285
Chennai              269
                    ...
Asansol                1
Tirunelvelli           1
Ernakulam              1
Nanded                 1
Asifabadbanglore       1
Name: count, Length: 337, dtype: int64
****************************************************************************************
********************
```

```
Top 10 categories in: 10board
------------------------------
10board
cbse                          1725
state board                   1139
icse                           276
ssc                            121
up board                        85
                               …
hse,orissa                       1
national public school           1
nagpur board                     1
jharkhand academic council       1
bse,odisha                       1
Name: count, Length: 274, dtype: int64
**********************************************************************************
********************
Top 10 categories in: 12board
------------------------------
12board
cbse                                1737
state board                         1228
icse                                 128
up board                              87
isc                                   45
                                      …
jawahar higher secondary school        1
nagpur board                           1
bsemp                                  1
board of higher secondary orissa       1
boardofintermediate                    1
Name: count, Length: 339, dtype: int64
**********************************************************************************
********************
Top 10 categories in: Specialization
------------------------------
Specialization
electronics and communication engineering      865
computer science & engineering                 731
information technology                         654
computer engineering                           593
computer application                           241
mechanical engineering                         201
electronics and electrical engineering         191
electronics & telecommunications               120
electrical engineering                          79
electronics & instrumentation eng               32
civil engineering                               29
```

```
electronics and instrumentation engineering     27
information science engineering                 27
instrumentation and control engineering         20
electronics engineering                         19
biotechnology                                   15
other                                           13
industrial & production engineering             10
applied electronics and instrumentation          9
chemical engineering                             8
telecommunication engineering                    6
mechanical and automation                        5
computer science and technology                  5
automobile/automotive engineering                5
mechatronics                                     4
instrumentation engineering                      4
aeronautical engineering                         3
electronics and computer engineering             3
electrical and power engineering                 2
metallurgical engineering                        2
biomedical engineering                           2
information & communication technology           2
industrial engineering                           2
computer science                                 2
control and instrumentation engineering          1
power systems and automation                     1
embedded systems technology                      1
mechanical & production engineering              1
computer and communication engineering           1
polymer technology                               1
information science                              1
internal combustion engine                       1
computer networking                              1
ceramic engineering                              1
electronics                                      1
industrial & management engineering              1
Name: count, dtype: int64
********************************************************************************
********************
Top 10 categories in: CollegeState
------------------------------
CollegeState
Uttar Pradesh        902
Karnataka            369
Tamil Nadu           363
Telangana            312
Maharashtra          257
Andhra Pradesh       222
West Bengal          192
```

```
Madhya Pradesh       189
Punjab               188
Haryana              177
Orissa               172
Rajasthan            168
Delhi                161
Uttarakhand          112
Kerala                33
Jharkhand             27
Chhattisgarh          27
Gujarat               24
Himachal Pradesh      16
Bihar                 10
Jammu and Kashmir      7
Assam                  5
Union Territory        5
Sikkim                 3
Goa                    1
Meghalaya              1
Name: count, dtype: int64
********************************************************************************
********************
```

[99]:
```
df['DOB'] = pd.to_datetime(df['DOB'])
df['Age'] = 2015 - df['DOB'].dt.year
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\2421441193.py:1: UserWarning:
Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df['DOB'] = pd.to_datetime(df['DOB'])
```

[100]:
```
df.columns
```

[100]:
```
Index(['Salary', 'DOJ', 'DOL', 'Designation', 'JobCity', 'Gender', 'DOB',
       '10percentage', '10board', '12graduation', '12percentage', '12board',
       'CollegeTier', 'Degree', 'Specialization', 'collegeGPA',
       'CollegeCityTier', 'CollegeState', 'GraduationYear', 'English',
       'Logical', 'Quant', 'Domain', 'ComputerProgramming',
       'ElectronicsAndSemicon', 'ComputerScience', 'conscientiousness',
       'agreeableness', 'extraversion', 'nueroticism', 'openess_to_experience',
       'Age'],
      dtype='object')
```

[101]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3943 entries, 0 to 3997
```

```
Data columns (total 32 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Salary                 3943 non-null   float64
 1   DOJ                    3943 non-null   datetime64[ns]
 2   DOL                    3943 non-null   datetime64[ns]
 3   Designation            3943 non-null   object
 4   JobCity                3943 non-null   object
 5   Gender                 3943 non-null   object
 6   DOB                    3943 non-null   datetime64[ns]
 7   10percentage           3943 non-null   float64
 8   10board                3943 non-null   object
 9   12graduation           3943 non-null   int64
 10  12percentage           3943 non-null   float64
 11  12board                3943 non-null   object
 12  CollegeTier            3943 non-null   int64
 13  Degree                 3943 non-null   object
 14  Specialization         3943 non-null   object
 15  collegeGPA             3943 non-null   float64
 16  CollegeCityTier        3943 non-null   int64
 17  CollegeState           3943 non-null   object
 18  GraduationYear         3943 non-null   float64
 19  English                3943 non-null   int64
 20  Logical                3943 non-null   int64
 21  Quant                  3943 non-null   int64
 22  Domain                 3943 non-null   float64
 23  ComputerProgramming    3943 non-null   float64
 24  ElectronicsAndSemicon  3943 non-null   int64
 25  ComputerScience        3943 non-null   int64
 26  conscientiousness      3943 non-null   float64
 27  agreeableness          3943 non-null   float64
 28  extraversion           3943 non-null   float64
 29  nueroticism            3943 non-null   float64
 30  openess_to_experience  3943 non-null   float64
 31  Age                    3943 non-null   int32
dtypes: datetime64[ns](3), float64(12), int32(1), int64(8), object(8)
memory usage: 1001.2+ KB
```

[102]:
```
numerical_cols = ['Salary','10percentage','12percentage','collegeGPA']
```

[119]:
```
def univariate_analysis(numerical_cols):
    for col_name in numerical_cols:
        print('')
        print(col_name)
        print('-'*20)
        print("Min: ",df[col_name].min())
        print("Max: ",df[col_name].max())
```

```
        print("Mean: ",df[col_name].mean())
        print("Std: ",df[col_name].std())
        print("Skew: ",df[col_name].skew())
        print("Kurt: ",df[col_name].kurt())
        print('')
        print('*'*40)
```

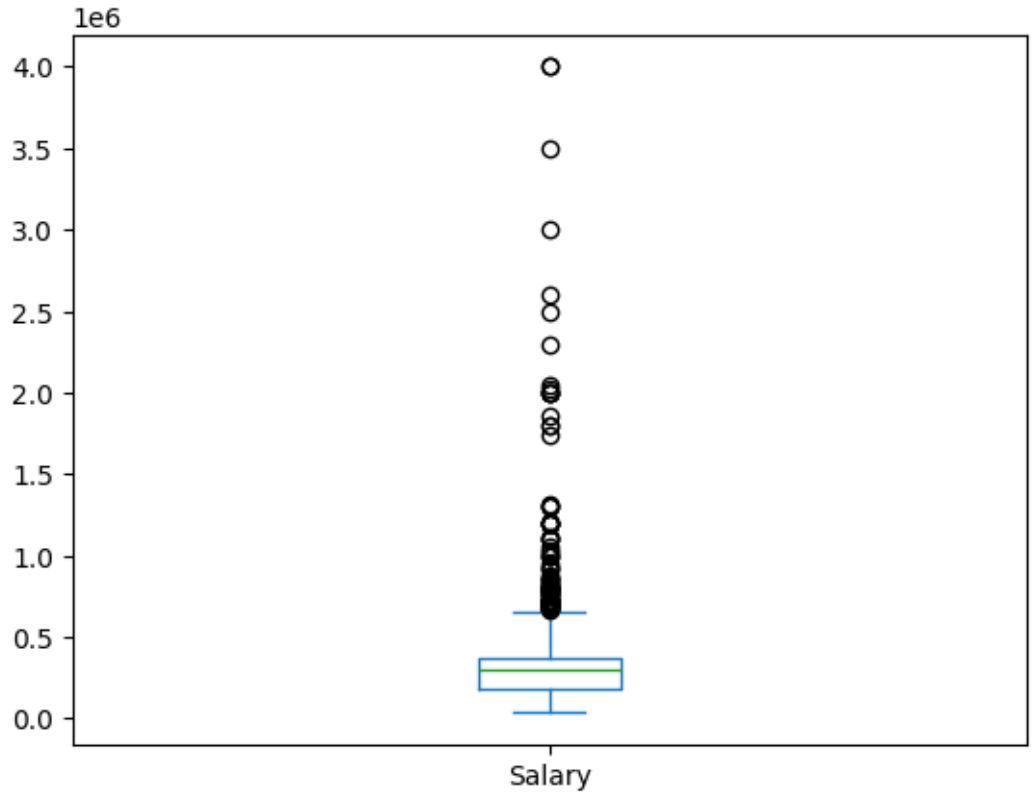
[120]:
```
univariate_analysis(numerical_cols)
```

```
Salary
--------------------
Min:  35000.0
Max:  4000000.0
Mean:  308256.1501394877
Std:  211763.10156460587
Skew:  6.5321030556113815
Kurt:  83.08114863205216

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

10percentage
--------------------
Min:  43.0
Max:  97.76
Mean:  77.9465508496069
Std:  9.839516824896757
Skew:  -0.5970747013477944
Kurt:  -0.09272276610232844

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

12percentage
--------------------
Min:  40.0
Max:  98.7
Mean:  74.45995434948009
Std:  11.00189441036818
Skew:  -0.03614997975718611
Kurt:  -0.6244313845576746

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

collegeGPA
--------------------
Min:  49.07
Max:  99.93
```

```
Mean:  71.69941922394115
Std:  7.417354992942565
Skew:  0.17141439286993057
Kurt:  0.07757400871909503

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

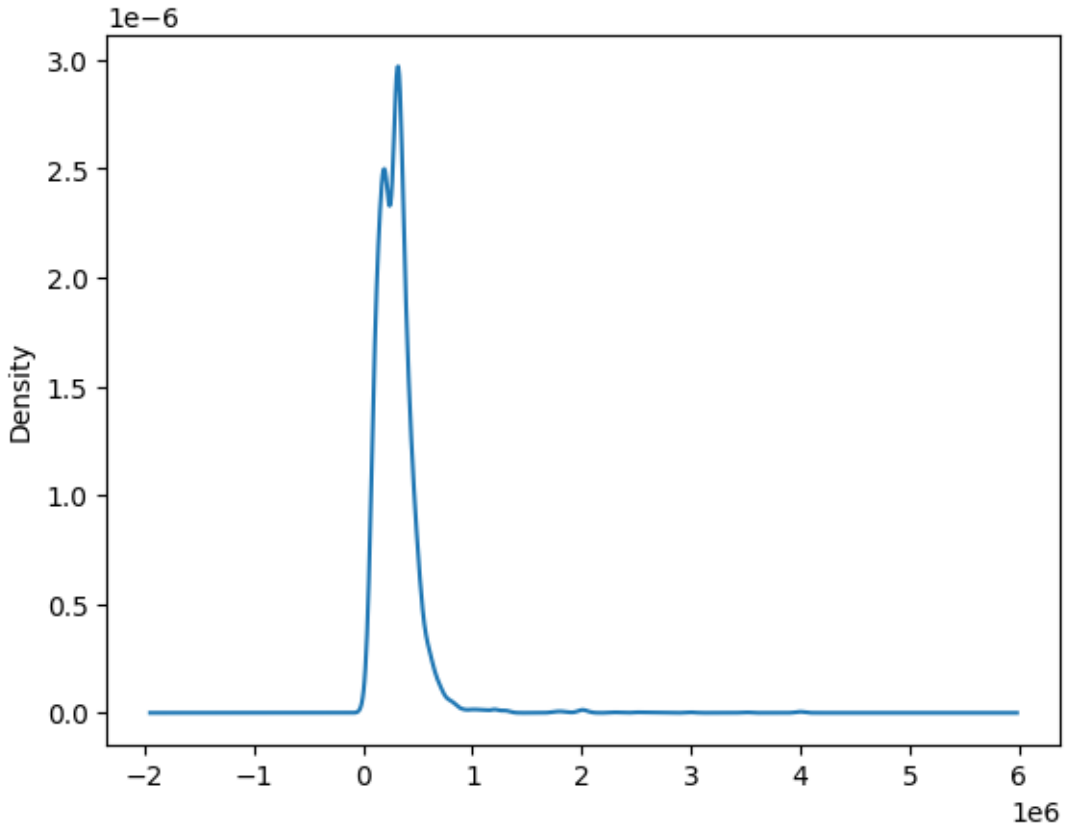
[129]:
```
df['Salary'].plot(kind='box')
```

[129]: <Axes: >



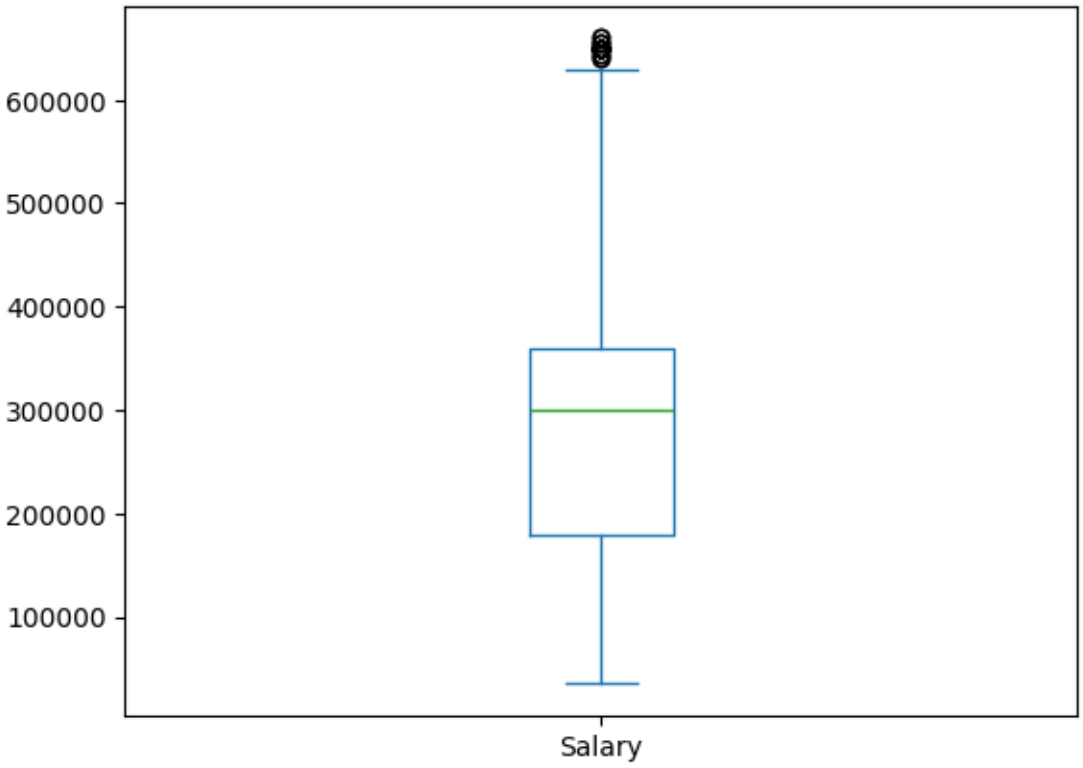
[130]:
```
df['Salary'].plot(kind='kde')
```

[130]: <Axes: ylabel='Density'>



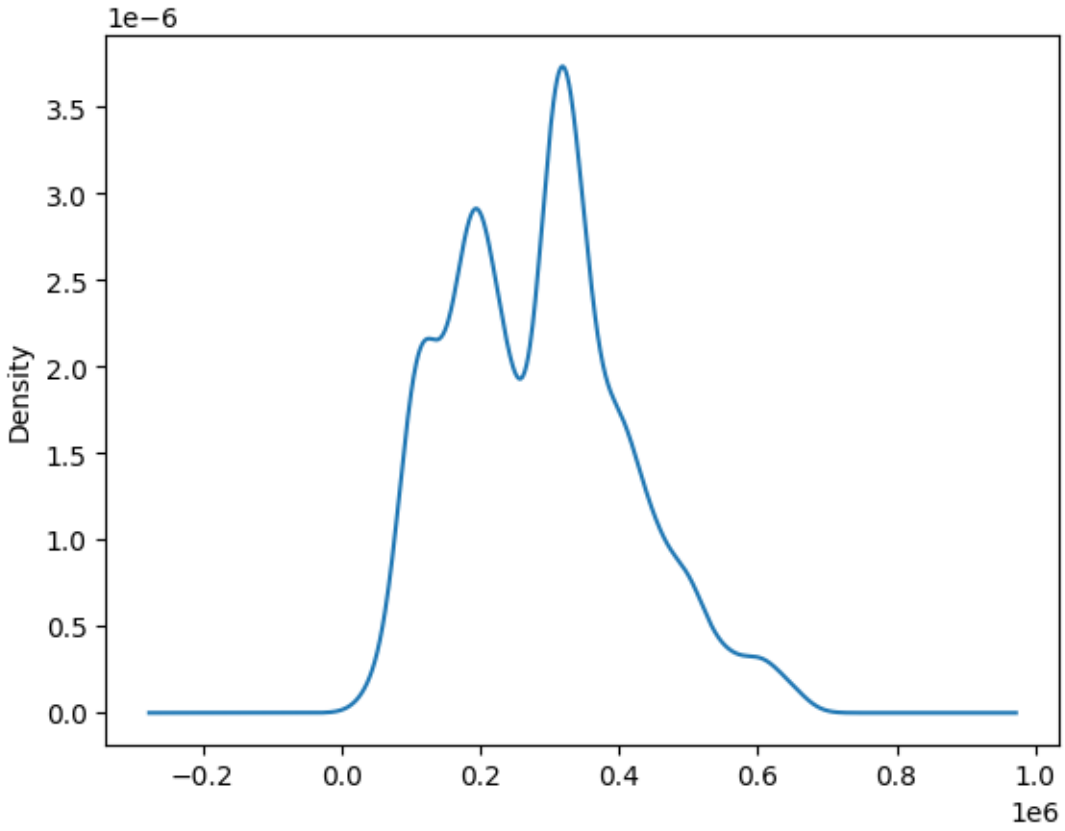
[226]:
```
q1 = df['Salary'].quantile(0.25)
q3 = df['Salary'].quantile(0.75)

IQR = q3 - q1
lower_bound = q1 - 1.5 * IQR
upper_bound = q3 + 1.5 * IQR

cleaned_df = df[(df['Salary'] >= lower_bound)& (df['Salary'] <= upper_bound)]
df = df[(df['Salary'] >= lower_bound)& (df['Salary'] <= upper_bound)]

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

[226]: <Axes: >



[132]:
```
cleaned_df['Salary'].plot(kind='kde')
```

[132]: <Axes: ylabel='Density'>



[133]:
```
high_salary_by_designation = cleaned_df.groupby('Designation')['Salary'].max()

high_salary_by_designation
```

[133]:
```
Designation
.net developer                  470000.0
.net web developer              305000.0
account executive               445000.0
account manager                 350000.0
admin assistant                 105000.0
                                   ...
web designer and seo            200000.0
web developer                   340000.0
web intern                      205000.0
website developer/tester        200000.0
windows systems administrator   200000.0
Name: Salary, Length: 413, dtype: float64
```

[134]:
```
print('Designation :',high_salary_by_designation.idxmax(),"&" ,"Salary:
 ↪",high_salary_by_designation.max())
```

```
Designation : assistant manager & Salary: 660000.0
```

[135]:
```
less_salary_by_designation = cleaned_df.groupby('Designation')['Salary'].min()

less_salary_by_designation
```

[135]:
```
Designation
.net developer                      35000.0
.net web developer                 180000.0
account executive                   85000.0
account manager                    350000.0
admin assistant                    100000.0
                                       ...
web designer and seo               200000.0
web developer                       60000.0
web intern                         205000.0
website developer/tester           200000.0
windows systems administrator      200000.0
Name: Salary, Length: 413, dtype: float64
```

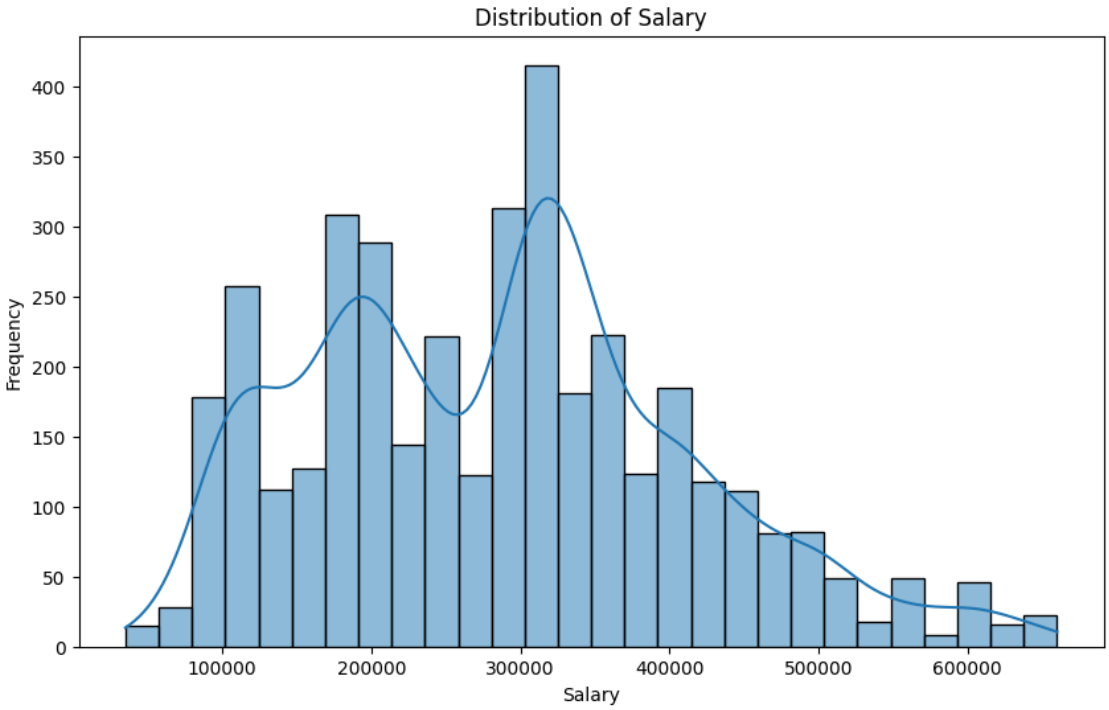
[136]:
```
print('Designation :',less_salary_by_designation.idxmin(),"&" ,"Salary:
 ↪",less_salary_by_designation.min())
```

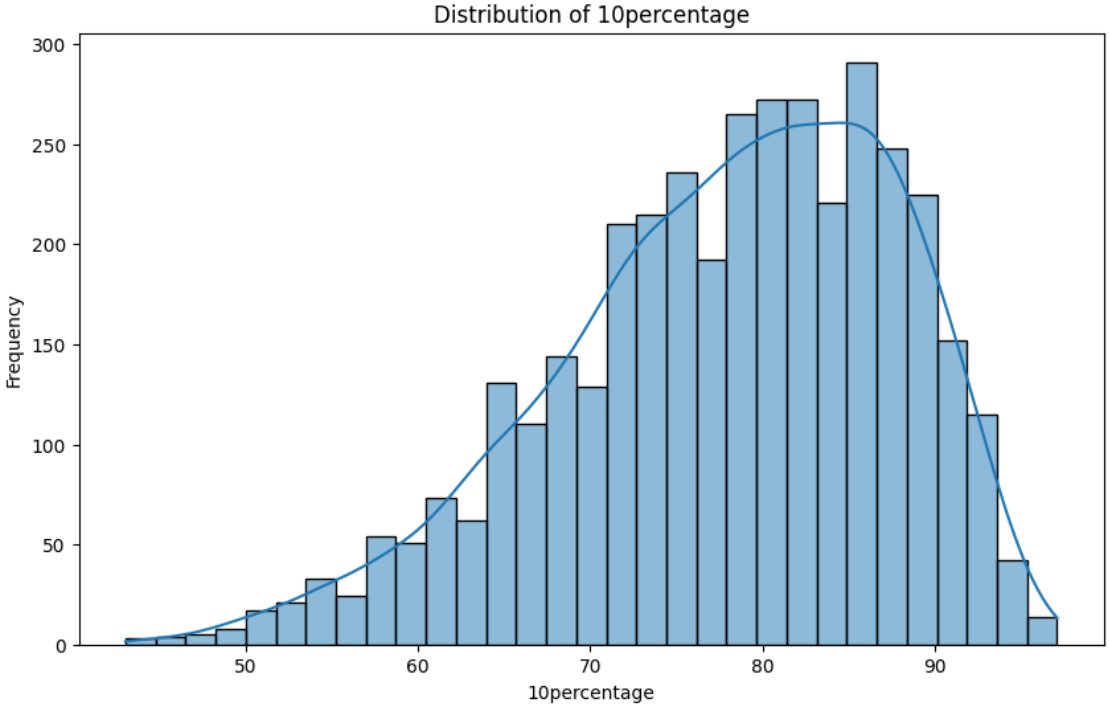
```
Designation : .net developer & Salary: 35000.0
```

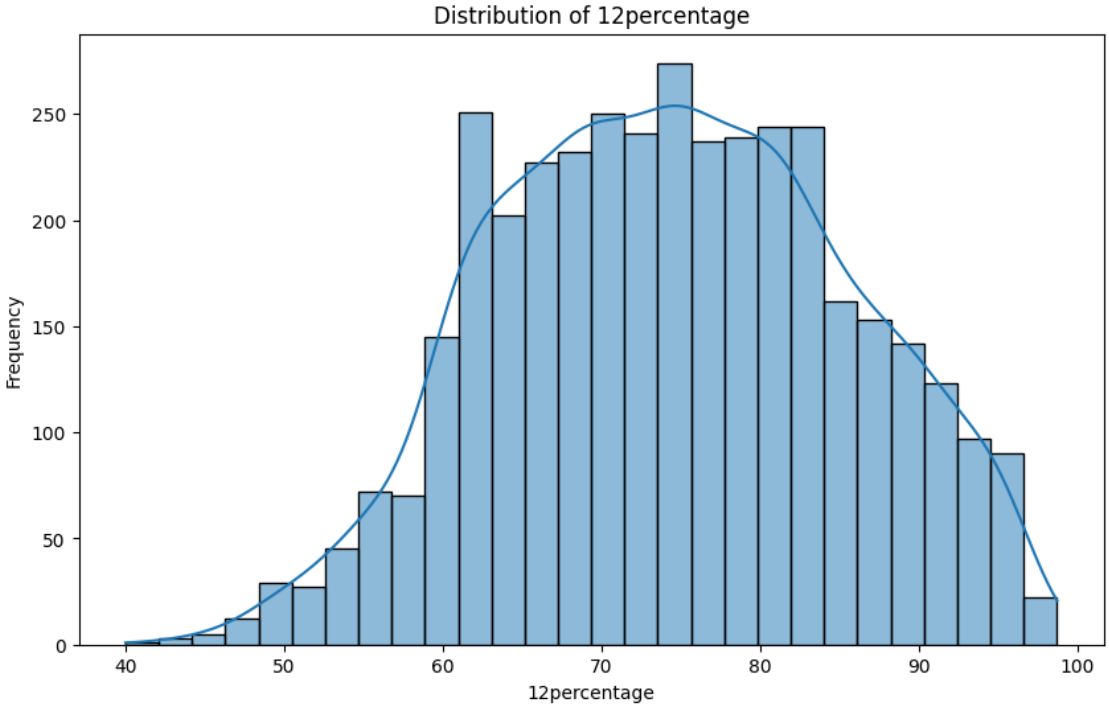
[138]:
```
salary_gender_analysis = df.groupby(['Gender', 'Designation'])['Salary'].
 ↪agg(['max', 'min', 'mean']).reset_index()
```

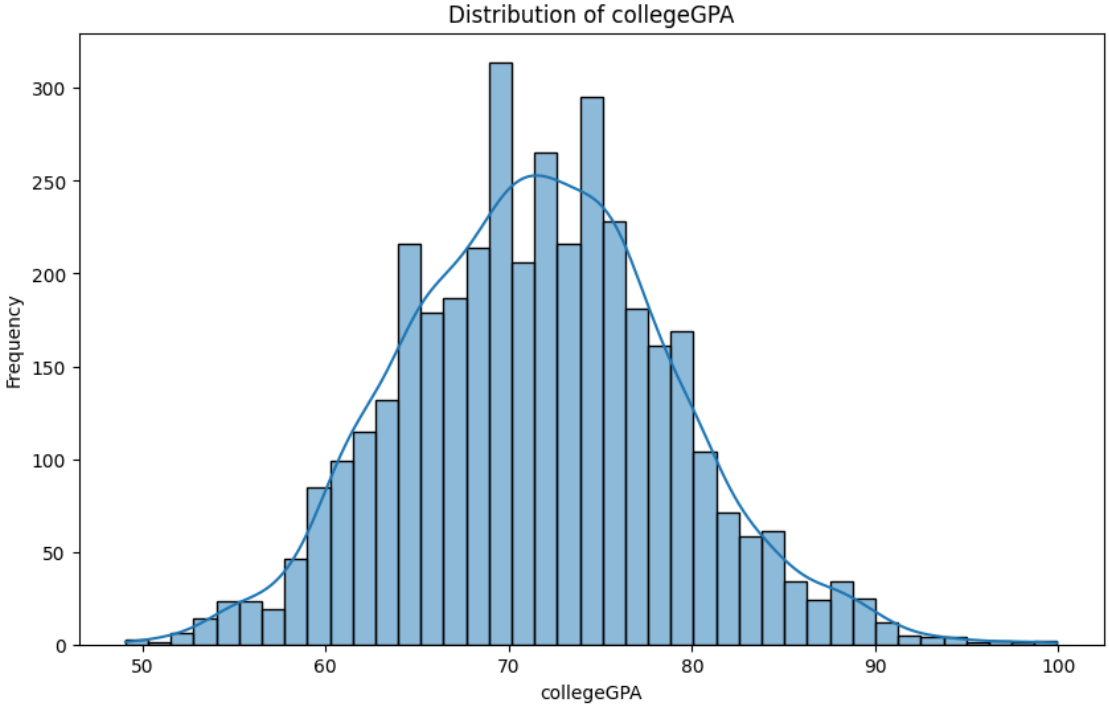
[142]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[147]:
```
for col_name in numerical_cols:
    plt.figure(figsize=(10, 6))
    sns.histplot(cleaned_df[col_name], kde=True)
    plt.title(f'Distribution of {col_name}')
    plt.xlabel(col_name)
    plt.ylabel('Frequency')
    plt.show()
```


Distribution of Salary
Frequency
Salary
0
50
100
150
200
250
300
350
400
100000
200000
300000
400000
500000
600000


Distribution of 10percentage
Frequency
10percentage
0
50
100
150
200
250
300
50
60
70
80
90


Distribution of 12percentage
Frequency
0
50
100
150
200
250
40
50
60
70
80
90
100
12percentage


Distribution of collegeGPA
Frequency
0
50
100
150
200
250
300
50
60
70
80
90
100
collegeGPA

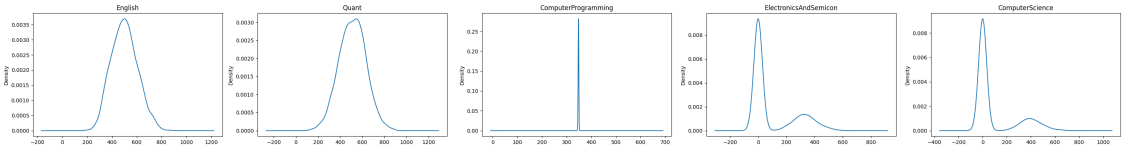
[150]:
```
AMCAT_Scores =␣
↪['English','Quant','ComputerProgramming','ElectronicsAndSemicon','ComputerScience']
```

[166]:
```
for col_name in AMCAT_Scores:
        print('')
        print(col_name)
        print('-'*20)
        print("Min: ",cleaned_df[col_name].min())
        print("Max: ",cleaned_df[col_name].max())
        print("Mean: ",cleaned_df[col_name].mean())
        print("Std: ",cleaned_df[col_name].std())
        print("Skew: ",cleaned_df[col_name].skew())
        print("Kurt: ",cleaned_df[col_name].kurt())
        print('')
        print('*'*40)
```

```
English
--------------------
Min:  180
Max:  875
Mean:  500.7501953633759
Std:  104.61922500219944
Skew:  0.20055135629872003
Kurt:  -0.23981310131506062

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

Quant
--------------------
Min:  120
Max:  900
Mean:  511.56994008856475
Std:  121.43862652924989
Skew:  -0.0154861851846843
Kurt:  -0.08209425112899105

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

ComputerProgramming
--------------------
Min:  166.0
Max:  516.0
Mean:  347.921333680646
Std:  7.084288239272976
Skew:  -1.523512840998529
Kurt:  348.5716746402922
```

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

ElectronicsAndSemicon
--------------------
Min:  0
Max:  612
Mean:  96.56889815056005
Std:  158.11993351005881
Skew:  1.1915391331022904
Kurt:  -0.21567185638617348

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

ComputerScience
--------------------
Min:  0
Max:  715
Mean:  92.2565772336546
Std:  175.29071086969478
Skew:  1.5180615078891895
Kurt:  0.6626747739669763

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

[163]:
```
fig, axs = plt.subplots(nrows=1, ncols=len(AMCAT_Scores), figsize=(30, 4))


for i, col_name in enumerate(AMCAT_Scores):
    cleaned_df[col_name].plot(kind='kde', ax=axs[i])
    axs[i].set_title(col_name)

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



[169]:
```
def collapsing_categories(cleaned_df, data):
    for Designation in cleaned_df[data].unique():
        min_count = cleaned_df[data].value_counts()[:10].min()
        if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0]␣
 ↪< min_count:
```

```
            cleaned_df.loc[cleaned_df[data] == Designation, data] = 'other'
```

[170]:
```
for cols in textual_columns:
    collapsing_categories(cleaned_df, cols)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

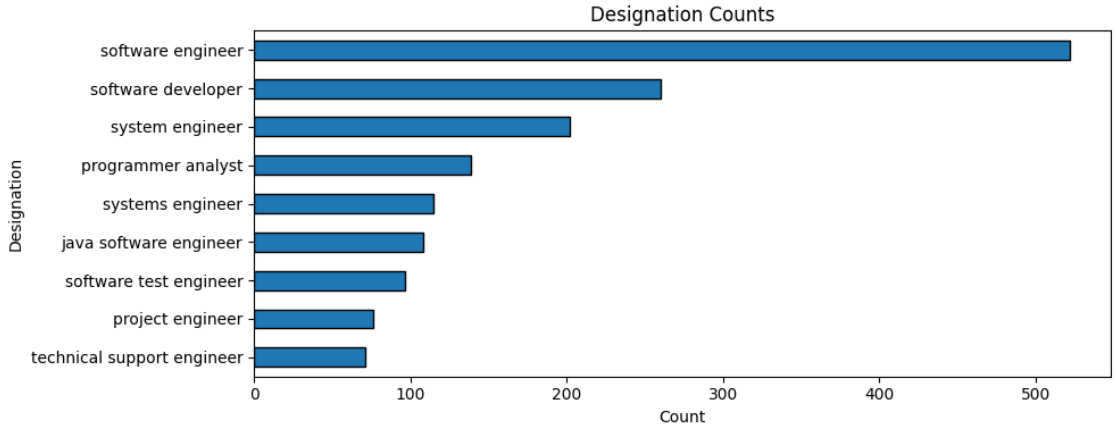
```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

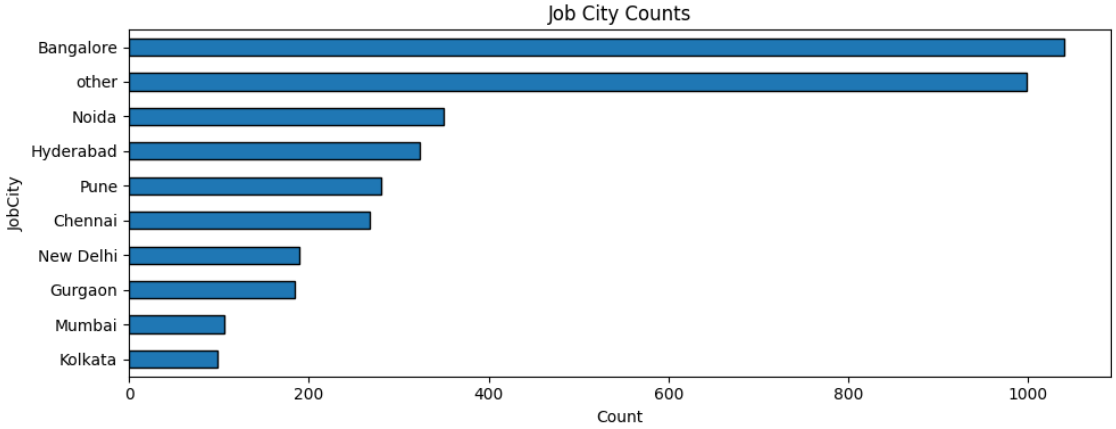
```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
C:\Users\admin\AppData\Local\Temp\ipykernel_11996\725818748.py:4: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  if cleaned_df[cleaned_df[data] == Designation][data].value_counts()[0] <
min_count:
```

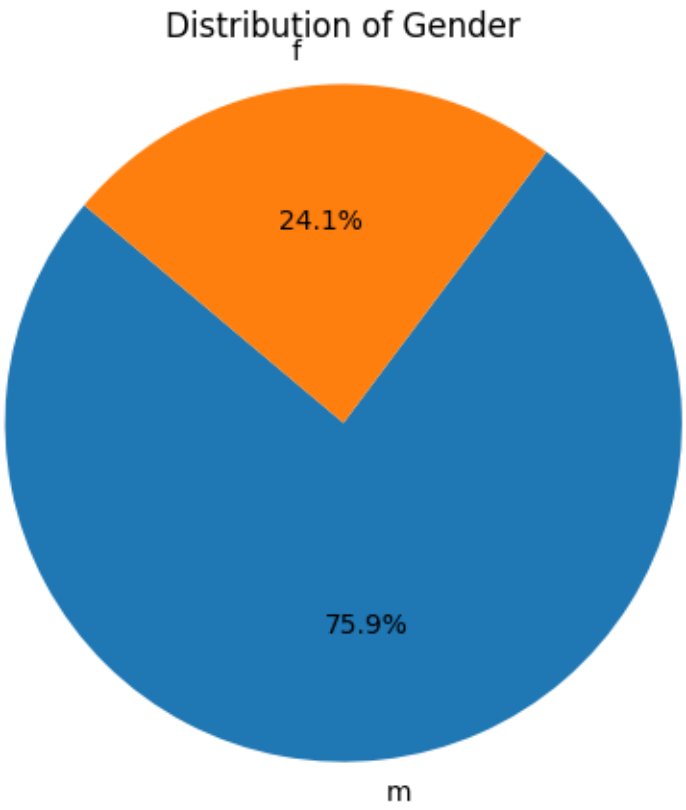
[176]:
```
cleaned_df['Designation'].value_counts()[1:].sort_values(ascending=True).
 ↪plot(kind='barh',title='Designation Counts',figsize=(10, 4),ec='k')
plt.ylabel('Designation')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



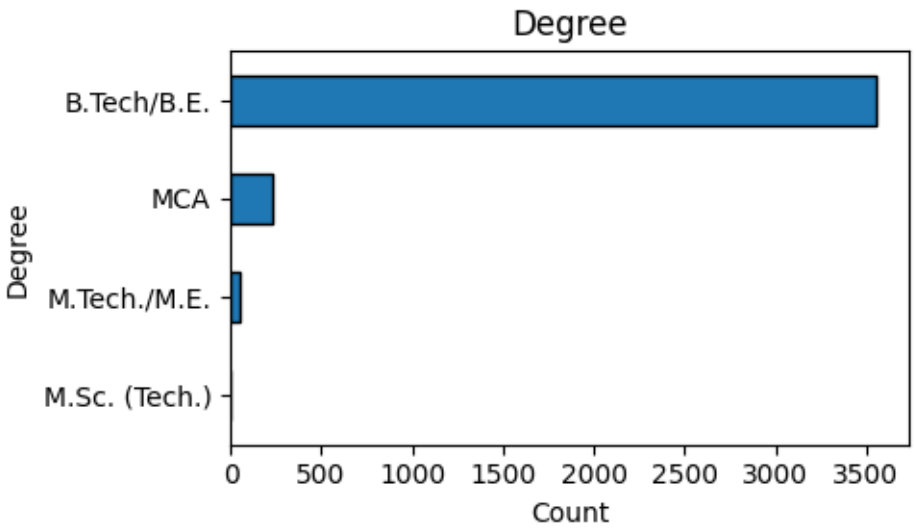
[174]:
```
cleaned_df['JobCity'].value_counts().sort_values(ascending=True).
  ↪plot(kind='barh',title='Job City Counts', figsize=(10,4), ec='k')
plt.ylabel('JobCity')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



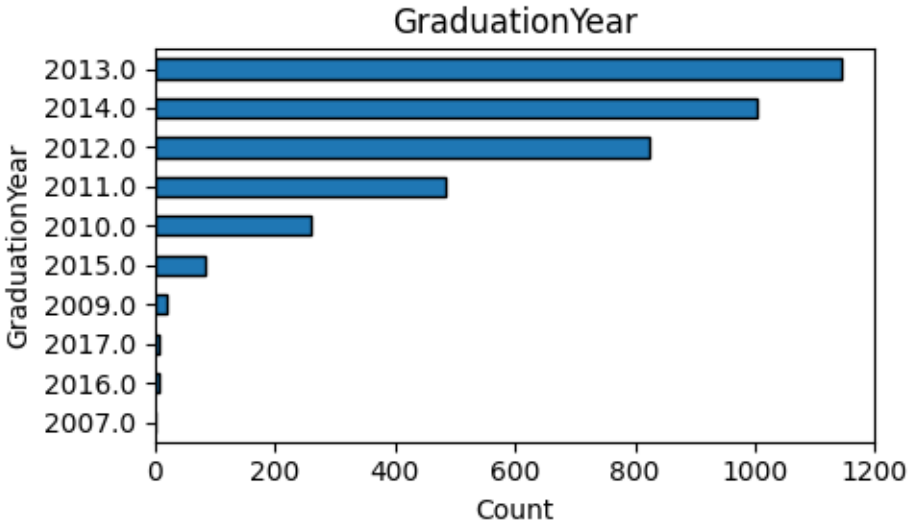
[183]:
```
plt.figure(figsize=(5, 5))
plt.pie(cleaned_df['Gender'].value_counts(), labels=cleaned_df['Gender'].
  ↪value_counts().index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Gender')
plt.axis('equal')
plt.show()
```



[187]:
```
cleaned_df['Degree'].value_counts().sort_values(ascending=True).
  ↪plot(kind='barh',title='Degree', figsize=(5,3), ec='k')
plt.ylabel('Degree')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```

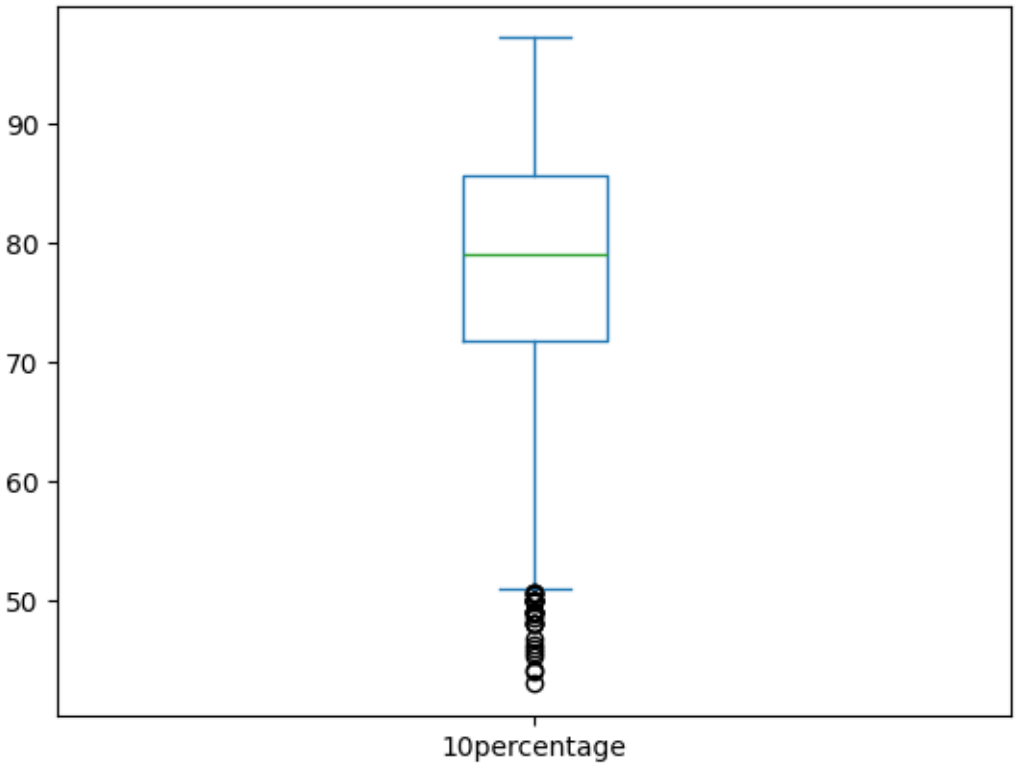


[188]:
```
cleaned_df['GraduationYear'].value_counts().sort_values(ascending=True).
↪plot(kind='barh',title='GraduationYear', figsize=(5,3), ec='k')
plt.ylabel('GraduationYear')
plt.xlabel('Count')
plt.tight_layout()
plt.show()
```



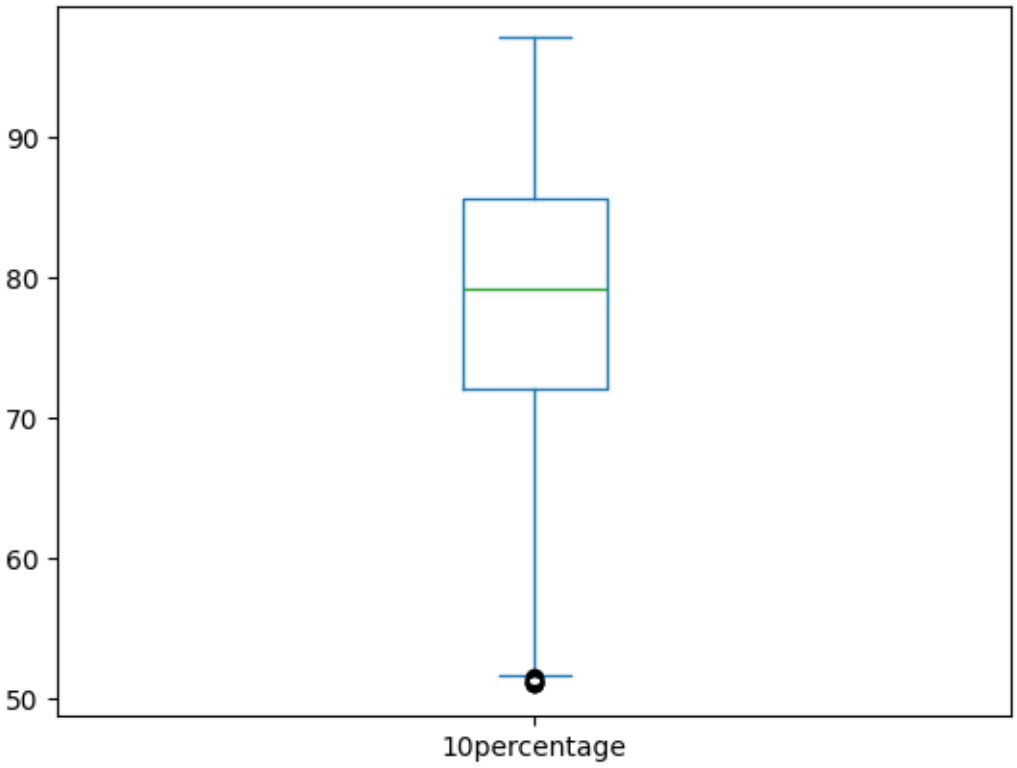
[193]:
```
cleaned_df['10percentage'].plot(kind='box')
```

[193]: <Axes: >



[194]:
```
q1 = cleaned_df['10percentage'].quantile(0.25)
q3 = cleaned_df['10percentage'].quantile(0.75)

IQR = q3 - q1
lower_bound = q1 - 1.5 * IQR
upper_bound = q3 + 1.5 * IQR

cleaned_df = cleaned_df[(cleaned_df['10percentage'] >= lower_bound)&␣
 ↪(cleaned_df['10percentage'] <= upper_bound)]

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

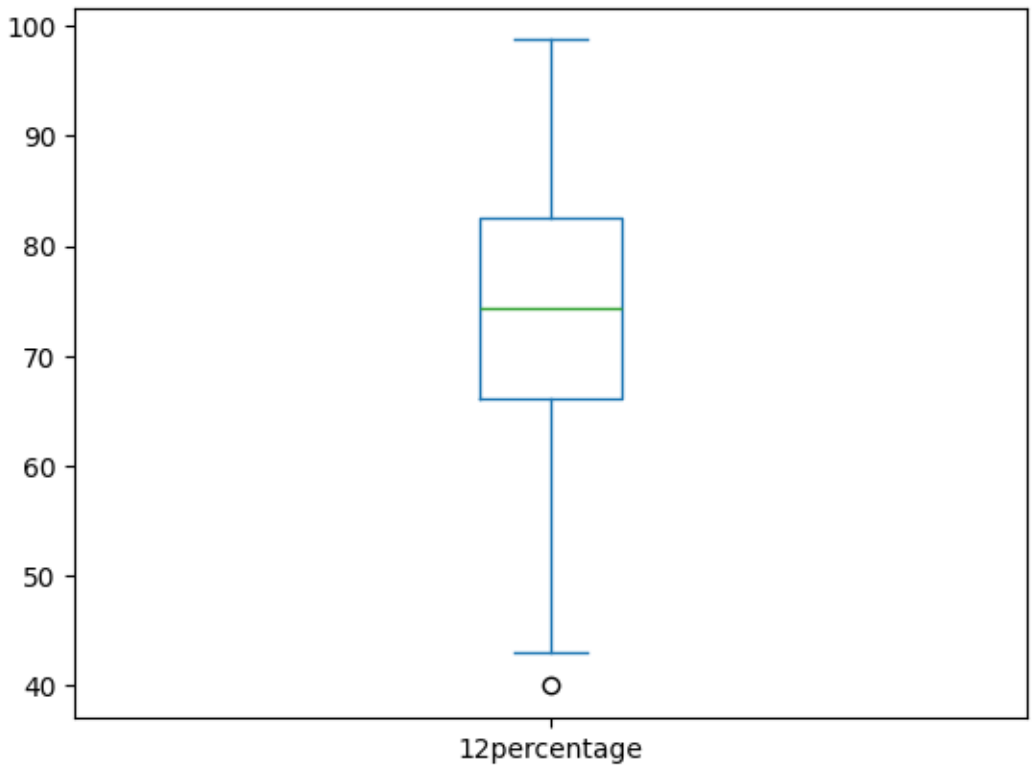
[194]: <Axes: >



[195]:
```
plt.figure(figsize=(8, 6))
plt.scatter(cleaned_df['10percentage'], cleaned_df['Salary'], alpha=0.5)
plt.title('Correlation between 10percentage and Salary')
plt.xlabel('10Percentage')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```

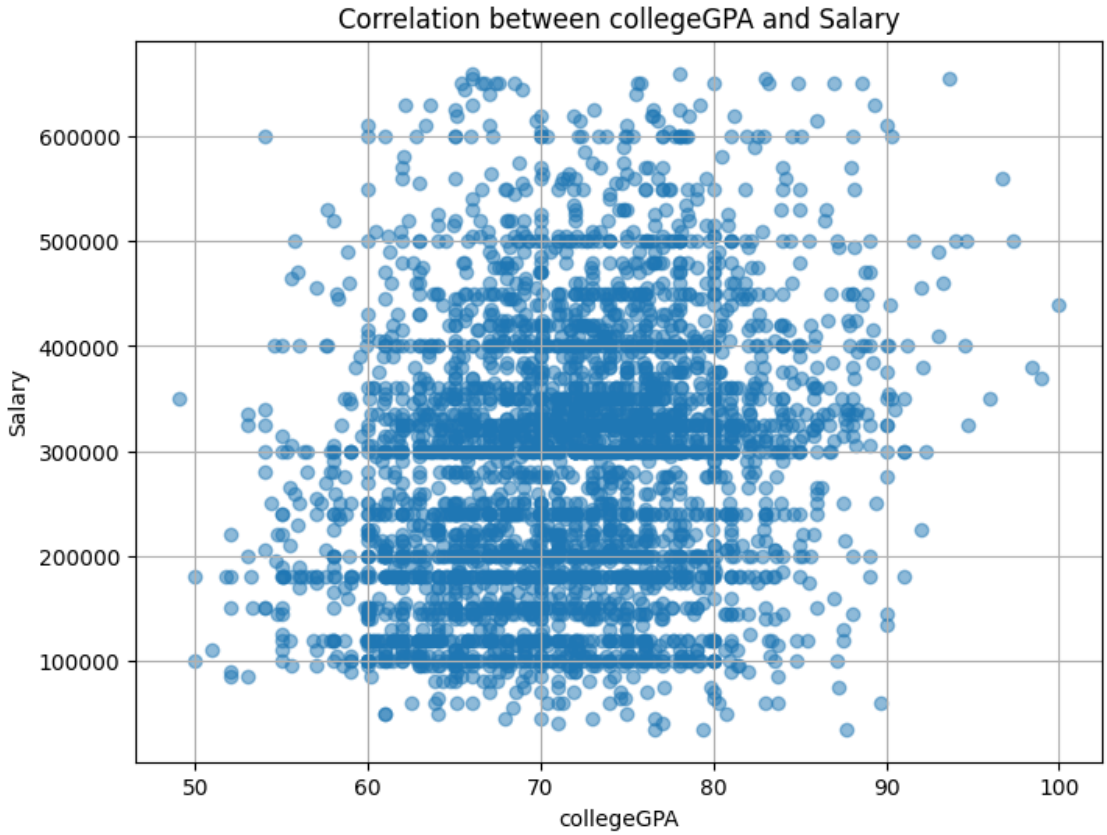
[196]:
```
cleaned_df['12percentage'].plot(kind='box')
```

[196]: <Axes: >

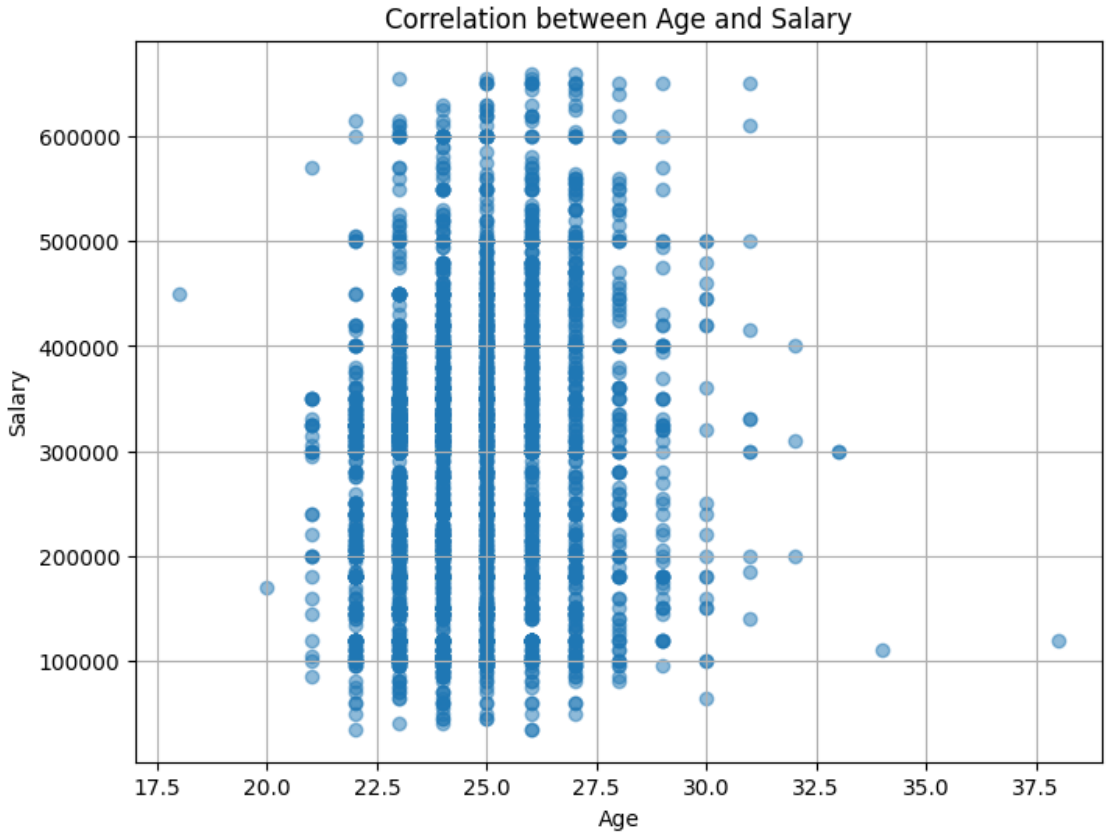


[198]:
```
plt.figure(figsize=(8, 6))
plt.scatter(cleaned_df['12percentage'], cleaned_df['Salary'], alpha=0.5)
plt.title('Correlation between 12percentage and Salary')
plt.xlabel('12Percentage')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```

[199]:
```
plt.figure(figsize=(8, 6))
plt.scatter(cleaned_df['collegeGPA'], cleaned_df['Salary'], alpha=0.5)
plt.title('Correlation between collegeGPA and Salary')
plt.xlabel('collegeGPA')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```

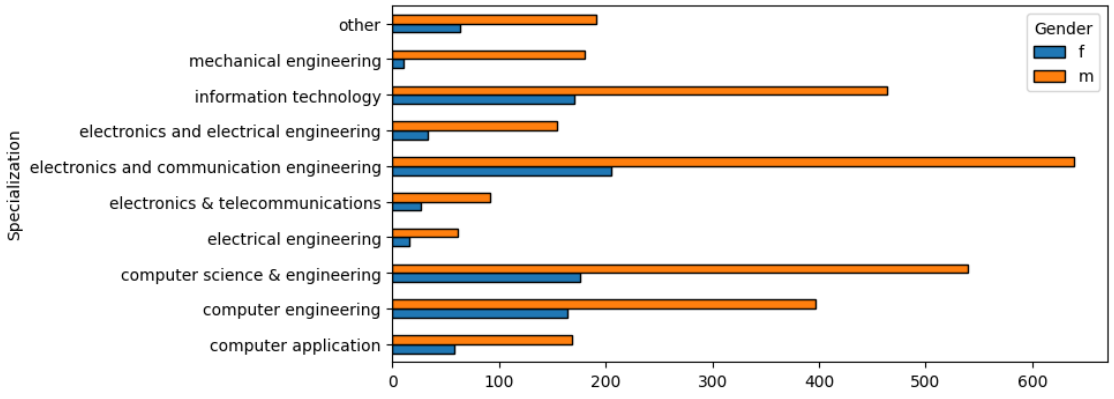


[204]:
```
plt.figure(figsize=(8, 6))
plt.scatter(cleaned_df['Age'], cleaned_df['Salary'], alpha=0.5)
plt.title('Correlation between Age and Salary')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.grid(True)
plt.show()
```



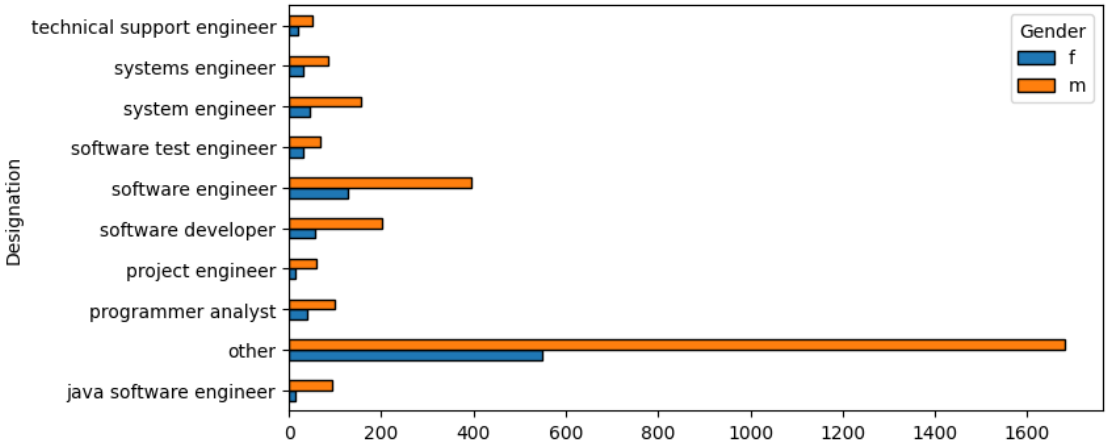
[206]:
```
pd.crosstab(cleaned_df['Gender'],cleaned_df['Specialization']).T.plot(kind =␣
↪'barh',ec = 'k',figsize = (8,4))
```

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



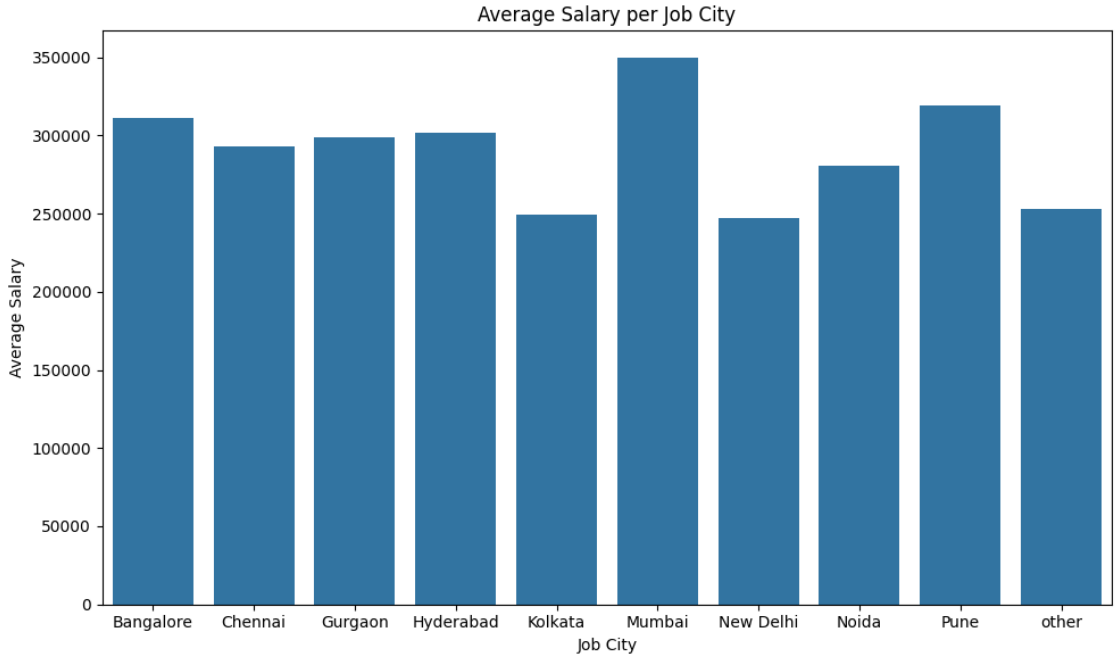
[207]:
```
pd.crosstab(cleaned_df['Gender'],cleaned_df['Designation']).T.plot(kind =␣
↪'barh',ec = 'k',figsize = (8,4))
```

[207]:
```
<Axes: ylabel='Designation'>
```



[208]:
```
avg_salary_per_city = cleaned_df.groupby('JobCity')['Salary'].mean().
↪reset_index()
```

[212]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='JobCity', y='Salary', data=avg_salary_per_city)
plt.title('Average Salary per Job City')
plt.xlabel('Job City')
plt.ylabel('Average Salary')
plt.tight_layout()
plt.show()
```



[227]:
```
Designations = df['Designation'].value_counts().sort_index()
```

[228]:
```
Designations
```

[228]:
```
Designation
.net developer                    34
.net web developer                 3
account executive                  4
account manager                    1
admin assistant                    2
                                  ..
web designer and seo               1
web developer                     52
web intern                         1
website developer/tester           1
windows systems administrator      1
Name: count, Length: 408, dtype: int64
```

[229]:
```
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'
)
```

[230]:
```
designation_analysis = df[(df["Designation"].isin(["programmer analyst",␣
 ↪"software engineer", "hardware engineer", "associate engineer"])) &
                (df["Specialization"].isin(["computer science & engineering",␣
 ↪"computer engineering"]))]
```

[231]:
```
designation_analysis.describe()
```

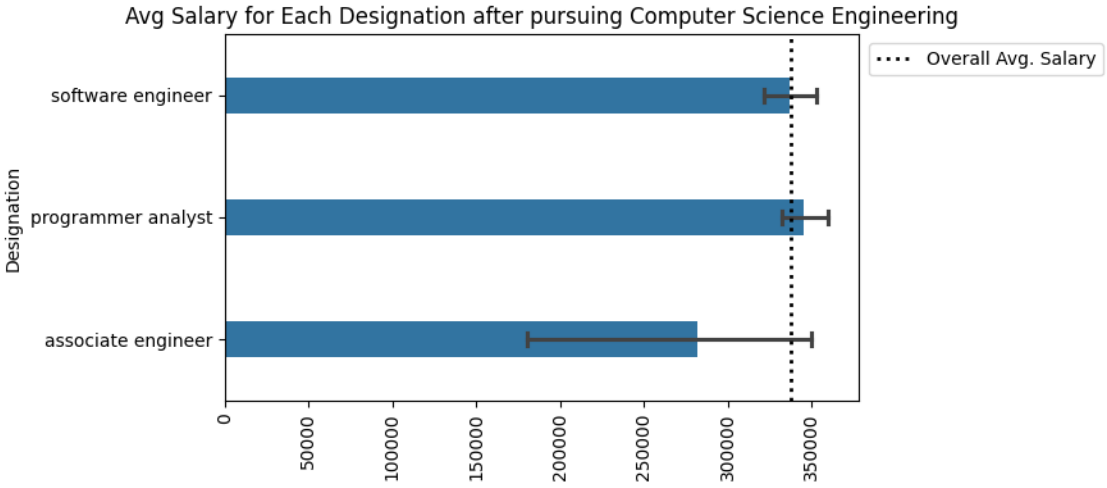
[231]:

```
              Salary                            DOJ  \
count     299.000000                            299
mean   337642.140468  2013-07-12 00:57:47.558528512
min     85000.000000            2009-09-01 00:00:00
25%    300000.000000            2012-09-16 00:00:00
50%    330000.000000            2013-12-01 00:00:00
75%    400000.000000            2014-07-01 00:00:00
max    650000.000000            2015-08-01 00:00:00
std    108677.416710                            NaN

                                DOL                            DOB  \
count                           299                            299
mean   2015-05-21 01:21:52.374582016  1991-01-19 01:41:08.227424768
min              2011-05-01 00:00:00            1977-10-30 00:00:00
25%              2015-04-01 00:00:00            1990-02-18 12:00:00
50%              2015-12-31 00:00:00            1991-04-04 00:00:00
75%              2015-12-31 00:00:00            1992-03-09 12:00:00
max              2015-12-31 00:00:00            1994-09-20 00:00:00
std                              NaN                            NaN

       10percentage  12graduation  12percentage  CollegeTier  collegeGPA  \
count    299.000000    299.000000    299.000000   299.000000  299.000000
mean      79.587625   2008.214047     76.106355     1.906355   72.672140
min       53.400000   1995.000000     49.000000     1.000000   49.070000
25%       73.000000   2007.000000     68.265000     2.000000   68.000000
50%       81.000000   2008.000000     76.000000     2.000000   72.000000
75%       86.940000   2009.000000     84.400000     2.000000   77.000000
max       95.040000   2012.000000     95.700000     2.000000   96.700000
std        9.120887      1.675225     10.287726     0.291823    7.367756

       CollegeCityTier  …      Domain  ComputerProgramming  \
count       299.000000  …  299.000000           299.000000
mean          0.357860  …    0.696373           348.173913
min           0.000000  …    0.040999           348.000000
25%           0.000000  …    0.563268           348.000000
```

```
50%           0.000000  …    0.744758           348.000000
75%           1.000000  …    0.901490           348.000000
max           1.000000  …    0.999910           400.000000
std           0.480174  …    0.240846             3.007238

       ElectronicsAndSemicon  ComputerScience  conscientiousness  \
count             299.000000       299.000000         299.000000
mean                1.003344       165.120401          -0.064367
min                 0.000000         0.000000          -3.199400
25%                 0.000000         0.000000          -0.589900
50%                 0.000000         0.000000          -0.015400
75%                 0.000000       376.000000           0.559100
max               300.000000       715.000000           1.995300
std                17.349448       213.563043           0.951807

       agreeableness  extraversion  nueroticism  openess_to_experience  \
count     299.000000    299.000000   299.000000              299.000000
mean        0.221235      0.025282    -0.255263               -0.170269
min        -2.951100     -2.602800    -2.389500               -3.960500
25%        -0.201200     -0.604800    -0.868200               -0.669200
50%         0.344800      0.010000    -0.260900               -0.050600
75%         0.711900      0.701000     0.272700                0.480500
max         1.904800      2.008000     2.934900                1.630200
std         0.832022      0.924867     0.917230                0.934429

              Age
count  299.000000
mean    24.441472
min     21.000000
25%     23.000000
50%     24.000000
75%     25.000000
max     38.000000
std      1.794591

[8 rows x 24 columns]
```

[235]:
```
plt.figure(figsize=(10, 4))
sns.barplot(x='Salary', y='Designation',
            data=designation_analysis,
            capsize=0.1,
            width=0.3)
plt.axvline(designation_analysis['Salary'].mean(), color='k',
           linestyle=':',
           linewidth=2, label='Overall Avg. Salary')
plt.title('Avg Salary for Each Designation after pursuing Computer Science␣
 ↪Engineering')
```

```
plt.legend(loc='upper right', bbox_to_anchor=(1.4, 1))
plt.xlabel('')
plt.xticks(rotation=90)

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



[ ]: