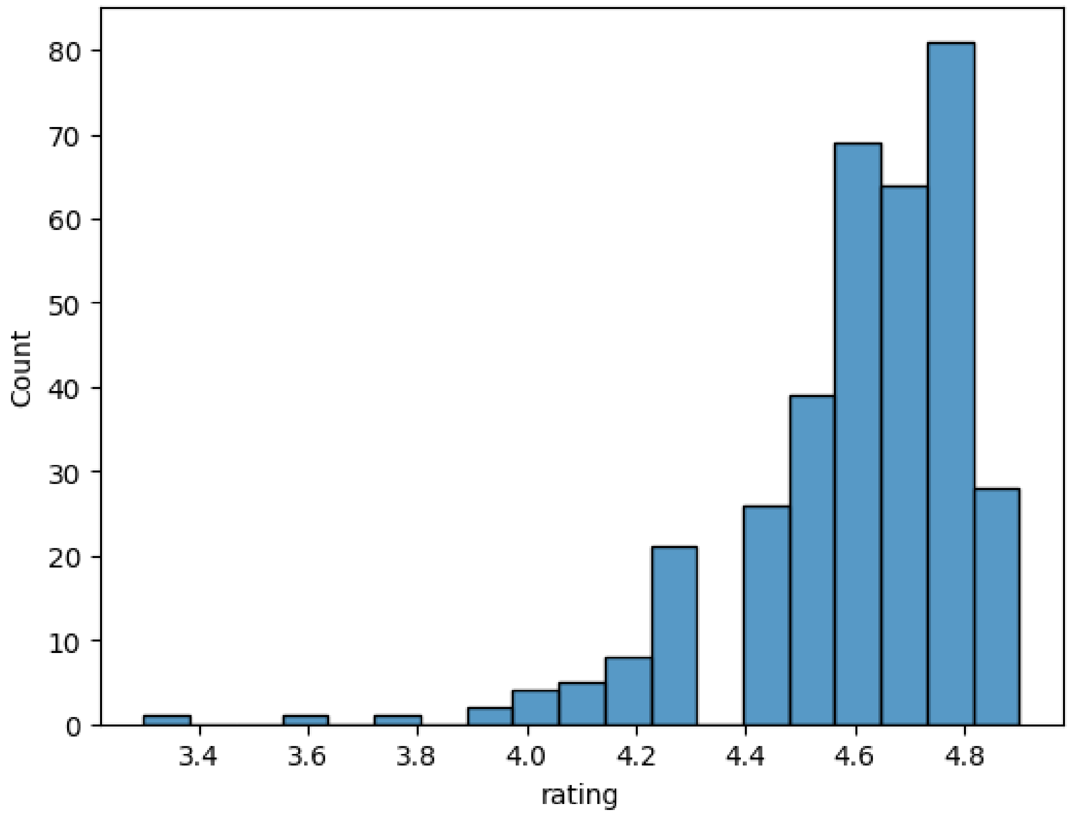
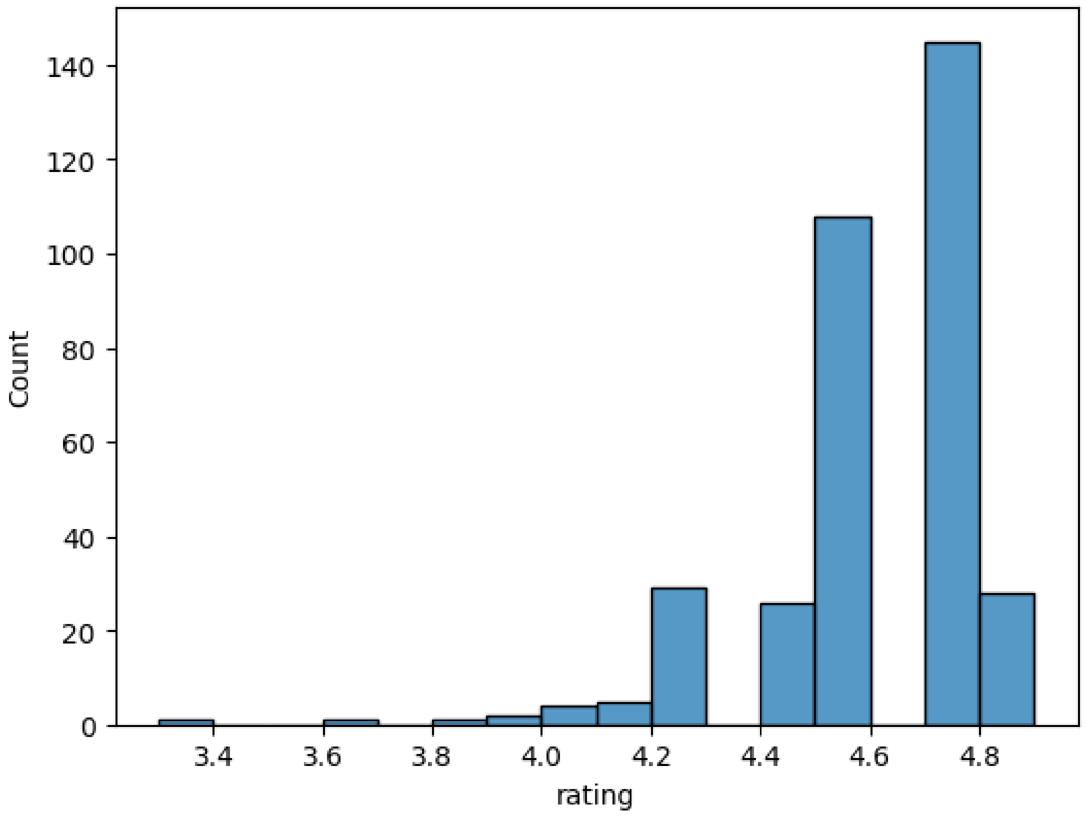
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
books = pd.read_csv('clean_books.csv')
sns.histplot(data=books,x='rating')
plt.show()
```



```
books.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    350 non-null    object
 1   author  350 non-null    object
 2   rating  350 non-null    float64
 3   year    350 non-null    int64
 4   genre   350 non-null    object
dtypes: float64(1), int64(1), object(3)
memory usage: 13.8+ KB
```

```
sns.histplot(data=books,x='rating',binwidth=0.1)
plt.show()
```



```
books.value_counts('genre')
```

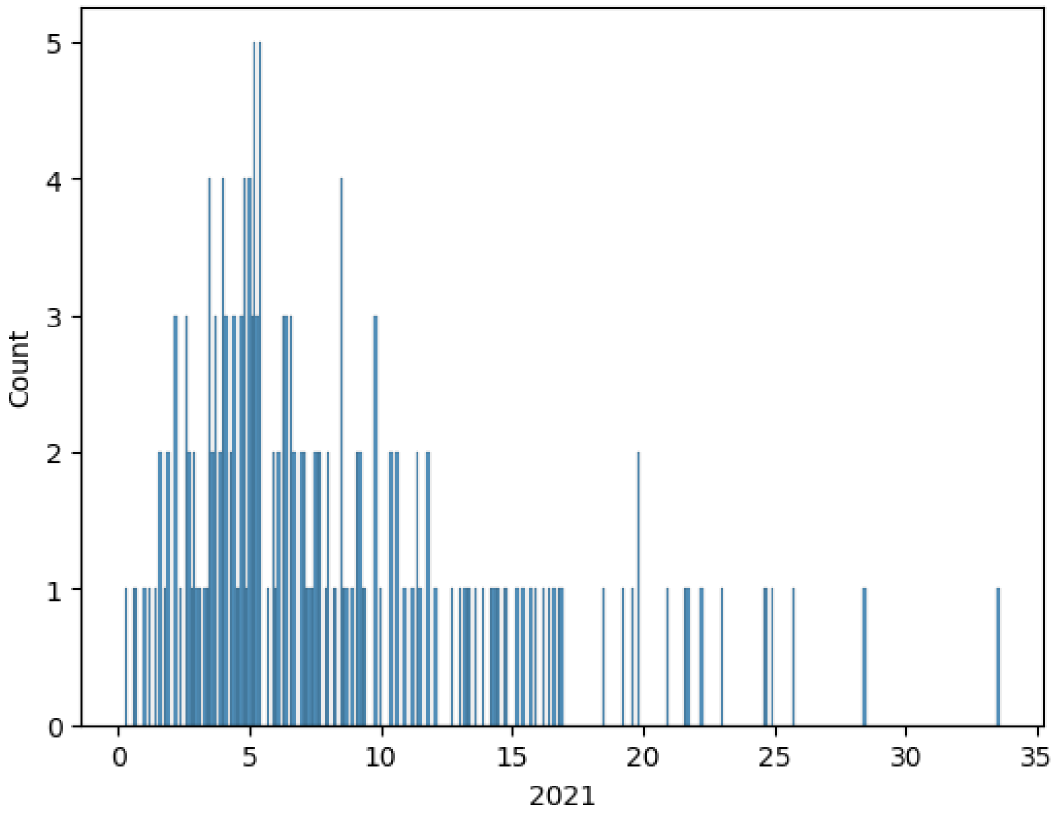
```
genre
Non Fiction    179
Fiction        131
Childrens       40
Name: count, dtype: int64
```

```
books['genre'].value_counts()
```

```
genre
Non Fiction    179
Fiction        131
Childrens       40
Name: count, dtype: int64
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
unemployment = pd.read_csv('clean_unemployment.csv')
unemployment.head()
```

```
  country_code          country_name      continent   2010   2011
2012  \
0          AFG           Afghanistan           Asia  11.35  11.05
11.34
1          AGO                Angola         Africa   9.43   7.36
7.35
2          ALB               Albania         Europe  14.09  13.48
13.38
3          ARE  United Arab Emirates           Asia   2.48   2.30
2.18
4          ARG             Argentina  South America   7.71   7.18
7.22

    2013   2014   2015   2016   2017   2018   2019   2020   2021
0  11.19  11.14  11.13  11.16  11.18  11.15  11.22  11.71  13.28
1   7.37   7.37   7.39   7.41   7.41   7.42   7.42   8.33   8.53
2  15.87  18.05  17.19  15.42  13.62  12.30  11.47  13.33  11.82
3   2.04   1.91   1.77   1.64   2.46   2.35   2.23   3.19   3.36
4   7.10   7.27   7.52   8.11   8.35   9.22   9.84  11.46  10.90

sns.histplot(data=unemployment,x='2021',binwidth=0.1)
plt.show()
```



```
books.dtypes
```

```
name        object
author      object
rating     float64
year         int64
genre       object
dtype: object
```

```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   name    350 non-null    object 
 1   author  350 non-null    object 
 2   rating  350 non-null    float64
 3   year    350 non-null    int64  
 4   genre   350 non-null    object 
dtypes: float64(1), int64(1), object(3)
memory usage: 13.8+ KB
```

```
books['genre'].isin(["Fiction","Non Fiction"])
```

```
0       True
1       True
2       True
3       True
4      False
       ...  
345     True
346     True
347     True
348     True
349    False
Name: genre, Length: 350, dtype: bool
```

```
~books['genre'].isin(["Fiction", "Non Fiction"])
```

```
0      False
1      False
2      False
3      False
4       True
       ...  
345    False
346    False
347    False
348    False
349     True
Name: genre, Length: 350, dtype: bool
```

```
books[books['genre'].isin(["Fiction", "Non Fiction"])].head()
```

```
                                                name              author 
rating  \
0                    10-Day Green Smoothie Cleanse            JJ Smith 
4.7   
1                                2011/22/63: A Novel        Stephen King 
4.6   
2          12 Rules for Life: An Antidote to Chaos  Jordan B. Peterson 
4.7   
3                           1984 (Signet Classics)       George Orwell 
4.7   
5  A Dance with Dragons (A Song of Ice and Fire)  George R. R. Martin 
4.4   

   year        genre  
0  2016  Non Fiction  
1  2011      Fiction  
2  2018  Non Fiction  
3  2017      Fiction  
5  2011      Fiction  
```

```
books.head()
```

```
                                                name  \
0                      10-Day Green Smoothie Cleanse   
1                                  11/22/63: A Novel   
2            12 Rules for Life: An Antidote to Chaos   
3                             1984 (Signet Classics)   
4  5,000 Awesome Facts (About Everything!) (Natio...   

                     author  rating  year        genre  
0                  JJ Smith     4.7  2016  Non Fiction  
1              Stephen King     4.6  2011      Fiction  
2        Jordan B. Peterson     4.7  2018  Non Fiction  
3             George Orwell     4.7  2017      Fiction  
4  National Geographic Kids     4.8  2019    Childrens  
```

```
books.select_dtypes('number').head()
```

```
   rating  year
0     4.7  2016
1     4.6  2011
2     4.7  2018
3     4.7  2017
4     4.8  2019
```

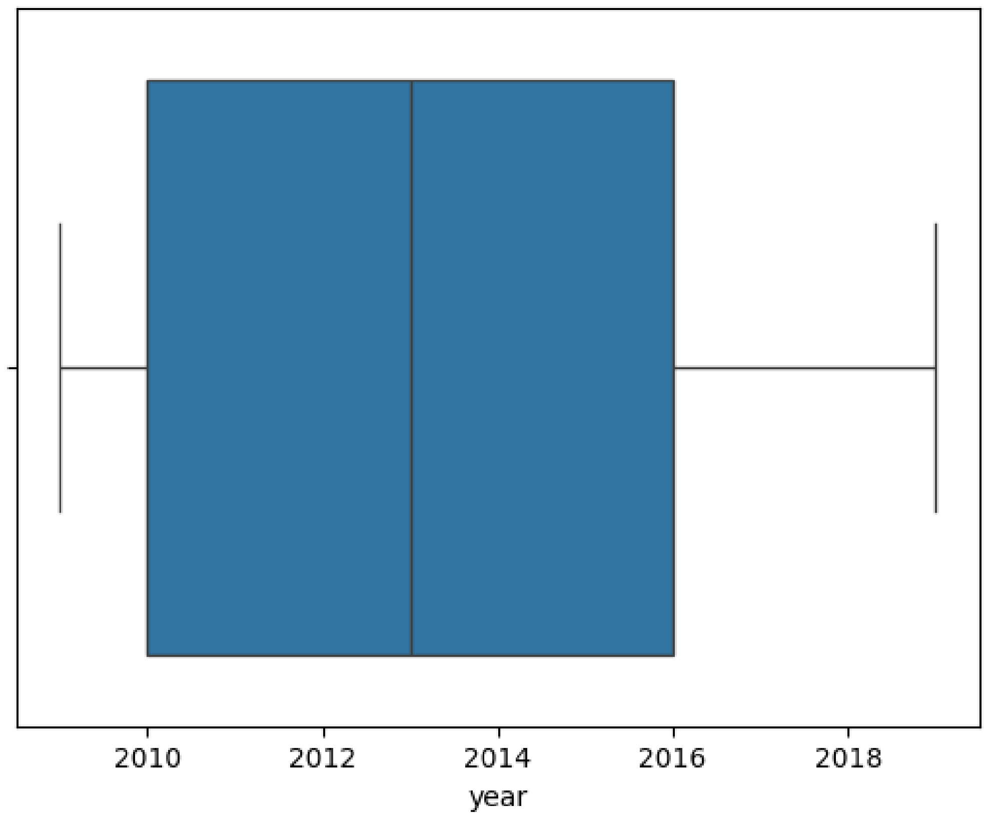
```
books["year"].min()
```

```
np.int64(2009)
```

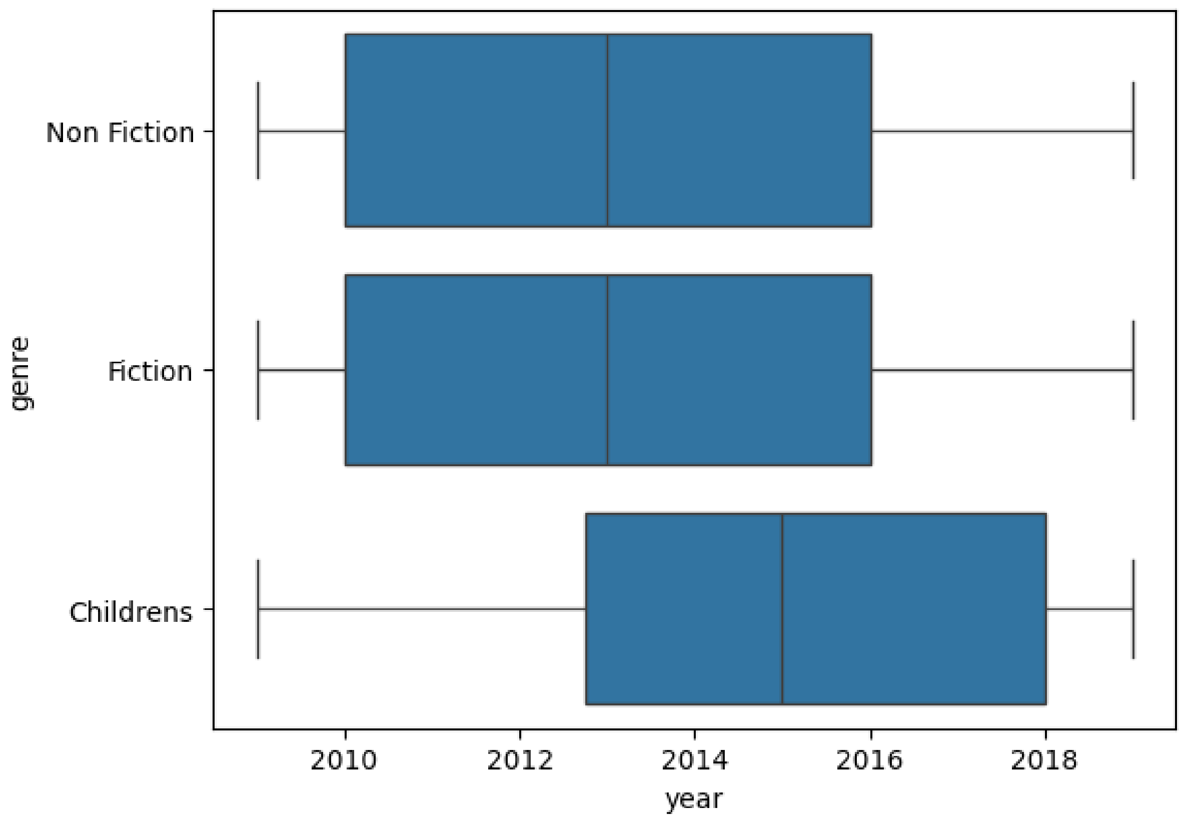
```
books["year"].max()
```

```
np.int64(2019)
```

```
sns.boxplot(data=books,x='year')
plt.show()
```



```
sns.boxplot(data=books,x='year',y='genre')
plt.show()
```



```
unemployment = pd.read_csv('clean_unemployment.csv')
unemployment.head()

  country_code          country_name      continent   2010   2011
2012  \
0          AFG           Afghanistan           Asia  11.35  11.05
11.34
1          AGO                Angola         Africa   9.43   7.36
7.35
2          ALB               Albania         Europe  14.09  13.48
13.38
3          ARE  United Arab Emirates           Asia   2.48   2.30
2.18
4          ARG             Argentina  South America   7.71   7.18
7.22

    2013   2014   2015   2016   2017   2018   2019   2020   2021
0  11.19  11.14  11.13  11.16  11.18  11.15  11.22  11.71  13.28
1   7.37   7.37   7.39   7.41   7.41   7.42   7.42   8.33   8.53
2  15.87  18.05  17.19  15.42  13.62  12.30  11.47  13.33  11.82
3   2.04   1.91   1.77   1.64   2.46   2.35   2.23   3.19   3.36
4   7.10   7.27   7.52   8.11   8.35   9.22   9.84  11.46  10.90

not_oceania = ~unemployment['country_name'].isin(['Oceania'])
print(not_oceania)
```

```
0      True
1      True
2      True
3      True
4      True
       ...
177    True
178    True
179    True
180    True
181    True
Name: country_name, Length: 182, dtype: bool
```

```
unemployment[not_oceania].head()
```

```
  country_code          country_name      continent   2010   2011
2012  \
0          AFG           Afghanistan           Asia  11.35  11.05
11.34
1          AGO                Angola         Africa   9.43   7.36
7.35
2          ALB               Albania         Europe  14.09  13.48
13.38
3          ARE  United Arab Emirates           Asia   2.48   2.30
2.18
4          ARG             Argentina  South America   7.71   7.18
7.22

    2013   2014   2015   2016   2017   2018   2019   2020   2021
0  11.19  11.14  11.13  11.16  11.18  11.15  11.22  11.71  13.28
1   7.37   7.37   7.39   7.41   7.41   7.42   7.42   8.33   8.53
2  15.87  18.05  17.19  15.42  13.62  12.30  11.47  13.33  11.82
3   2.04   1.91   1.77   1.64   2.46   2.35   2.23   3.19   3.36
4   7.10   7.27   7.52   8.11   8.35   9.22   9.84  11.46  10.90
```

```
unemployment['2021'].max()
```

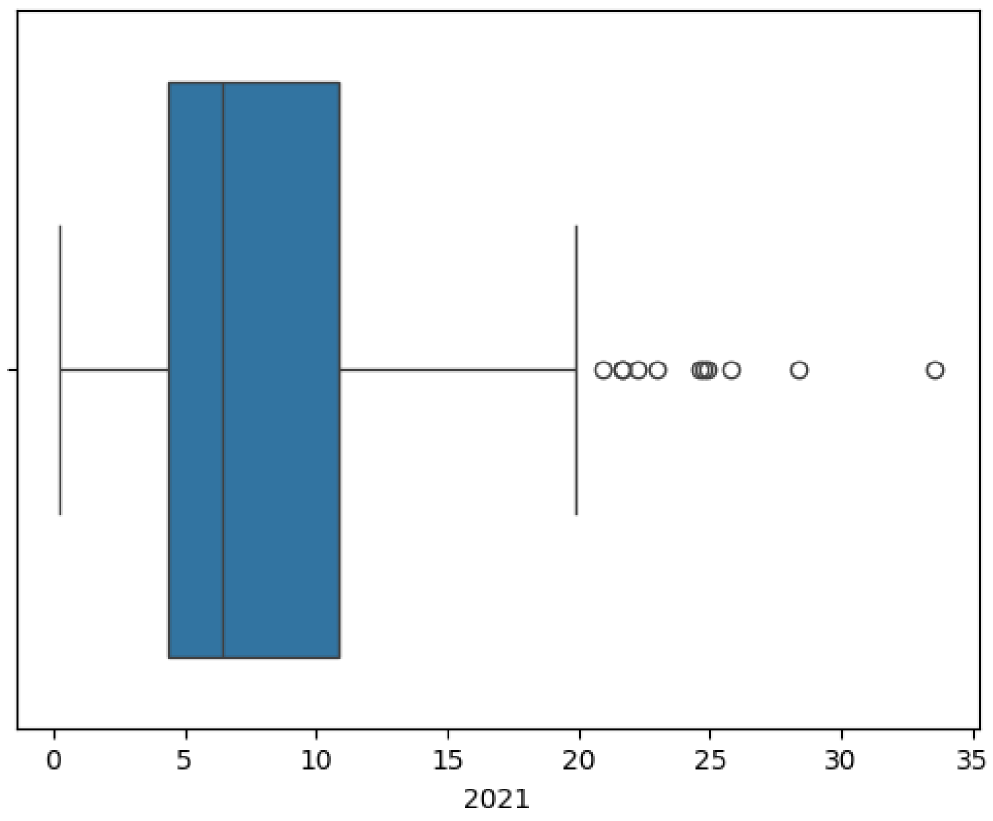
```
np.float64(33.56)
```

```
unemployment['2021'].min()
```

```
np.float64(0.26)
```

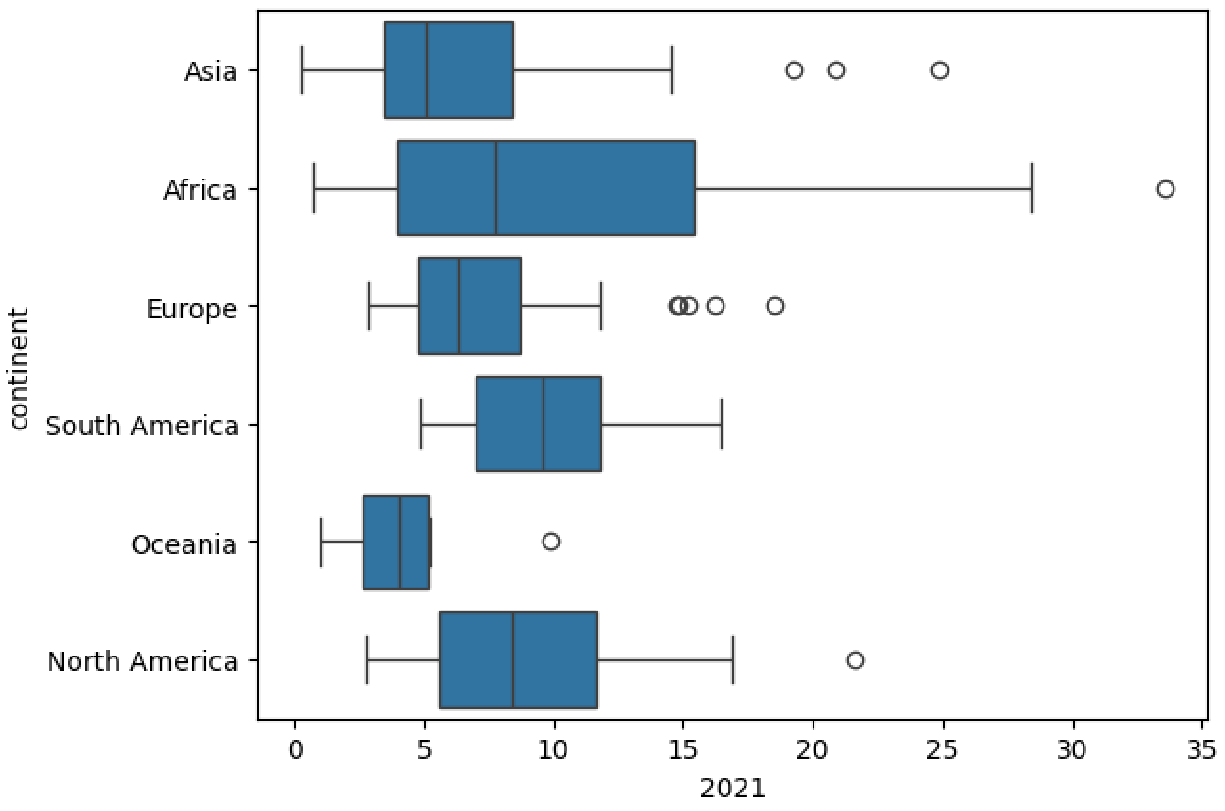
```
sns.boxplot(data=unemployment,x="2021")
```

```
<Axes: xlabel='2021'>
```



```
sns.boxplot(data=unemployment,x="2021",y='continent')
```

```
<Axes: xlabel='2021', ylabel='continent'>
```



```
books.groupby("genre")[['rating','year']].mean()

                rating         year
genre
Childrens     4.780000  2015.075000
Fiction       4.570229  2013.022901
Non Fiction   4.598324  2013.513966

books.groupby("genre").mean(numeric_only=True)

                rating         year
genre
Childrens     4.780000  2015.075000
Fiction       4.570229  2013.022901
Non Fiction   4.598324  2013.513966

books[['rating','year']].agg(['std','mean'])

        rating         year
std   0.226941     3.284711
mean  4.608571  2013.508571

books.agg({"rating": ["mean","std"] , "year": ["median"]})

          rating    year
mean    4.608571     NaN
```

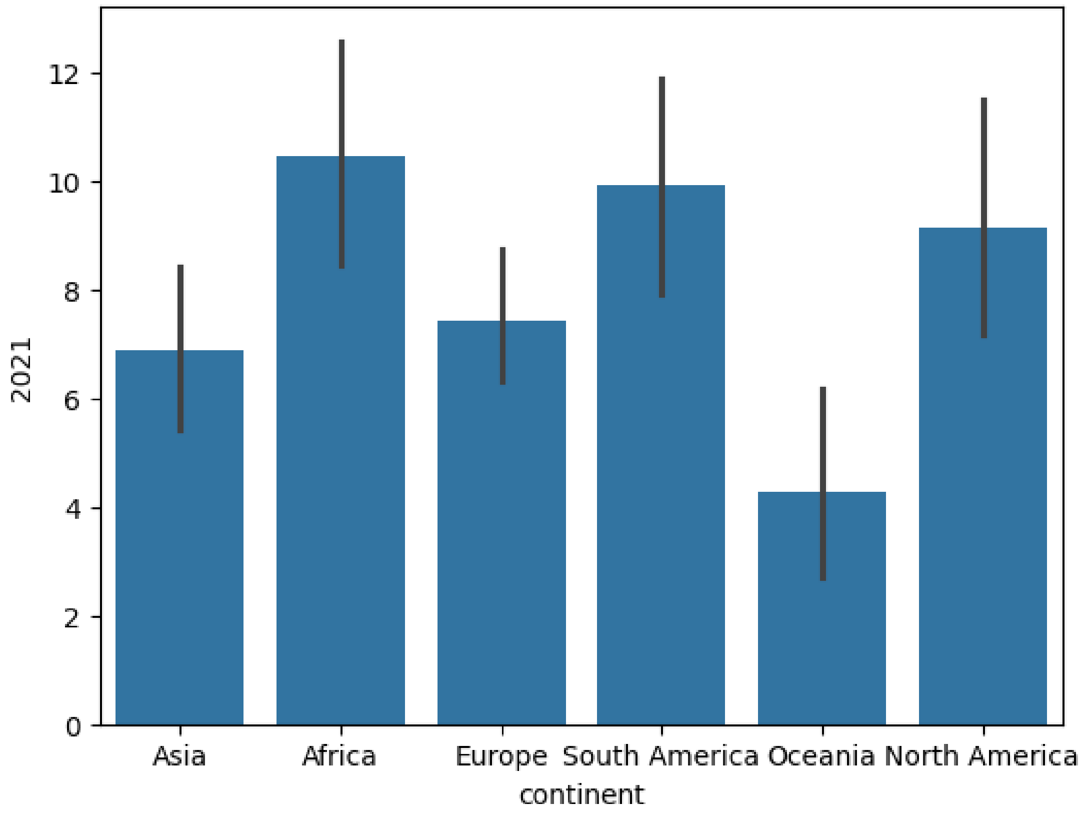
```
std     0.226941     NaN
median       NaN  2013.0


numeric_col =
list(unemployment.select_dtypes(include=np.number).columns)
unemployment[numeric_col].agg(["mean","std"])

          2010      2011      2012      2013      2014      2015
2016  \
mean  8.409286  8.315440  8.317967  8.344780  8.179670  8.058901
7.925879
std   6.248887  6.266795  6.367270  6.416041  6.284241  6.161170
6.045439

          2017      2018      2019      2020      2021
mean  7.668626  7.426429  7.243736  8.420934  8.390879
std   5.902152  5.818915  5.696573  6.040915  6.067192

unemployment.groupby('continent')[numeric_col].agg(["mean","std"])

                    2010                2011                2012
\
                    mean       std       mean       std       mean
std
continent

Africa          9.343585  7.411259   9.369245  7.401556   9.240755
7.264542
Asia            6.240638  5.146175   5.942128  4.779575   5.835319
4.756904
Europe         11.008205  6.392063  10.947949  6.539538  11.325641
7.003527
North America   8.663333  5.115805   8.563333  5.377041   8.448889
5.495819
Oceania         3.622500  2.054721   3.647500  2.008466   4.103750
2.723118
South America   6.870833  2.807058   6.518333  2.801577   6.410833
2.936508

                    2013                2014            ...      2017
\
                    mean       std       mean       std  ...      mean

continent                                                ...

Africa          9.132453  7.309285   9.121321  7.291359  ...  9.284528

Asia            5.852128  4.668405   5.853191  4.681301  ...  6.171277

Europe         11.466667  6.969209  10.971282  6.759765  ...  8.359744
```

```
North America   8.840556  6.081829   8.512222  5.801927  ...  7.391111

Oceania         3.980000  2.640119   3.976250  2.659205  ...  3.872500

South America   6.335000  2.808780   6.347500  2.834332  ...  7.281667


                             2018                2019
2020  \
                    std       mean       std      mean       std
mean
continent

Africa         7.407620  9.237925  7.358425  9.264340  7.455293
10.307736
Asia           5.277201  6.090213  5.409128  5.949149  5.254008
7.012340
Europe         5.177845  7.427436  4.738206  6.764359  4.124734
7.470513
North America  5.326446  7.281111  5.253180  7.095000  4.770490
9.297778
Oceania        2.492834  3.851250  2.455893  3.773750  2.369068
4.273750
South America  3.398994  7.496667  3.408856  7.719167  3.379845
10.275000

                              2021
                    std       mean       std
continent
Africa         7.928166  10.473585  8.131636
Asia           5.699609   6.906170  5.414745
Europe         4.071218   7.414872  3.947825
North America  4.963045   9.155000  5.076482
Oceania        2.617490   4.280000  2.671522
South America  3.411263   9.924167  3.611624

[6 rows x 24 columns]
```

```
continent_summary = unemployment.groupby("continent").agg(
    max_rate_2021 = ('2021','mean'),
    std_rate_2021 = ('2021','std')
)
continent_summary
```

```
               max_rate_2021  std_rate_2021
continent
Africa             10.473585       8.131636
Asia                6.906170       5.414745
```

```
Europe              7.414872      3.947825
North America       9.155000      5.076482
Oceania             4.280000      2.671522
South America       9.924167      3.611624

sns.barplot(data=unemployment,x='continent',y='2021')
plt.show()
```



```
salaries = pd.read_csv("ds_salaries.csv")
print(salaries.isna().sum())

Unnamed: 0            0
work_year             0
experience_level      0
employment_type       0
job_title             0
salary                0
salary_currency       0
salary_in_usd         0
employee_residence    0
remote_ratio          0
company_location      0
company_size          0
dtype: int64
```

```python
planes = pd.read_csv('Airlines_unclean.csv')
planes.isna().sum()
```

```
Unnamed: 0           0
Airline            427
Date_of_Journey    322
Source             187
Destination        347
Route              256
Dep_Time           260
Arrival_Time       194
Duration           214
Total_Stops        212
Additional_Info    589
Price              616
dtype: int64
```

```python
threshold = len(planes) * 0.05
threshold
```

```
533.0
```

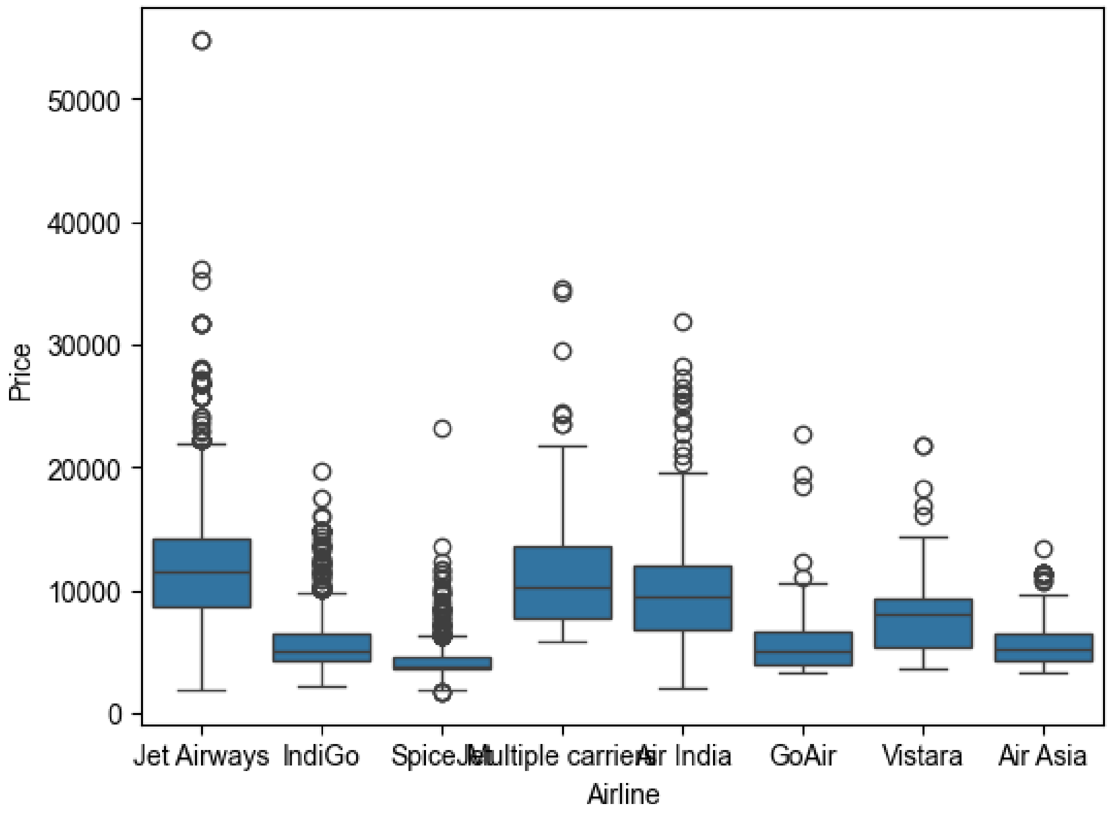
```python
cols_to_drop = planes.columns[planes.isna().sum() <= threshold]
planes.dropna(subset = cols_to_drop,inplace=True)
planes.isna().sum()
```

```
Unnamed: 0           0
Airline              0
Date_of_Journey      0
Source               0
Destination          0
Route                0
Dep_Time             0
Arrival_Time         0
Duration             0
Total_Stops          0
Additional_Info    300
Price              368
dtype: int64
```

```python
# Check the values of the Additional_Info column
print(planes["Additional_Info"].value_counts())

# Create a box plot of Price by Airline
sns.boxplot(data=planes, x='Airline', y='Price')
sns.set(rc={"figure.figsize":(8, 6)}) #width=8, #height=6
plt.show()
```

```
Additional_Info
No info                         6399
In-flight meal not included     1525
```

```
No check-in baggage included     258
1 Long layover                    14
Change airports                    7
No Info                            2
Business class                     1
Red-eye flight                     1
2 Long layover                     1
Name: count, dtype: int64
```



```
planes = planes.drop(columns="Additional_Info")
price_dict = planes.groupby("Airline")["Price"].median().to_dict()
price_dict
```

```
{'Air Asia': 5192.0,
 'Air India': 9443.0,
 'GoAir': 5003.5,
 'IndiGo': 5054.0,
 'Jet Airways': 11507.0,
 'Multiple carriers': 10197.0,
 'SpiceJet': 3873.0,
 'Vistara': 8028.0}
```

```
planes["Price"] =
planes["Price"].fillna(planes["Airline"].map(price_dict))
planes.isna().sum()
```

```
Unnamed: 0         0
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Price              0
dtype: int64
```

```
print(salaries.select_dtypes("object").head())
```

```
  experience_level employment_type                   job_title  \
0               MI              FT              Data Scientist
1               SE              FT  Machine Learning Scientist
2               SE              FT           Big Data Engineer
3               MI              FT        Product Data Analyst
4               SE              FT   Machine Learning Engineer

  salary_currency employee_residence company_location company_size
0             EUR                 DE               DE            L
1             USD                 JP               JP            S
2             GBP                 GB               GB            M
3             USD                 HN               HN            S
4             USD                 US               US            L
```

```
print(salaries["Designation"].value_counts())
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/indexes/base.py:3805, in Index.get_loc(self,
key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()
File pandas/_libs/hashtable_class_helper.pxi:7081, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7089, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Designation'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
Cell In[43], line 1
----> 1 print(salaries["Designation"].value_counts())

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/frame.py:4102, in
DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/indexes/base.py:3812, in Index.get_loc(self,
key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will
raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-
raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'Designation'

# Filter the DataFrame for object columns
non_numeric = planes.select_dtypes("object")

# Loop through columns
for col in non_numeric.columns:
  # Print the number of unique values
  print(f"Number of unique values in {col} column: ",
non_numeric[col].nunique())
```