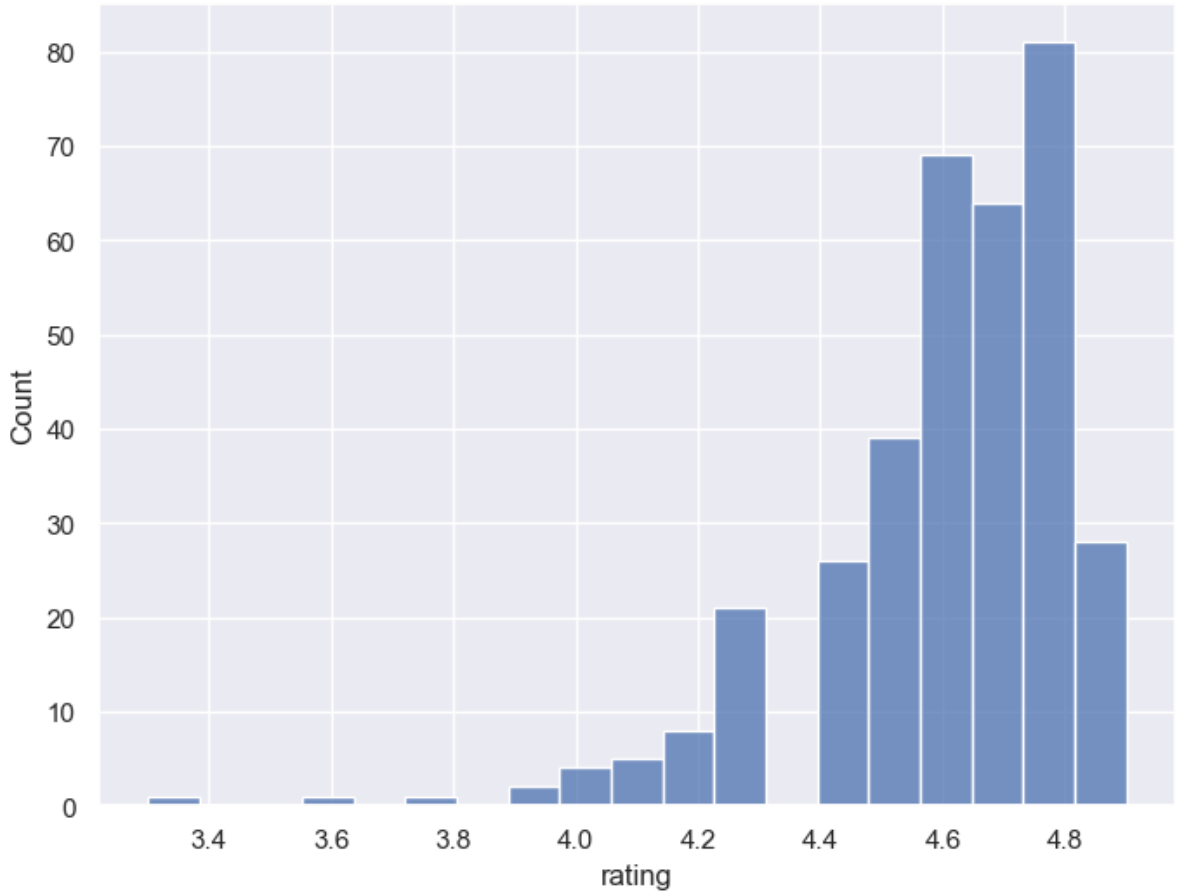
In [538…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
books = pd.read_csv("clean_books.csv")
```

In [539…
```
sns.histplot(data=books , x='rating')
plt.show()
```



In [540…
```
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
```

In [541…
```
books.value_counts("genre")
```

Out[541]:
```
genre
Non Fiction    179
Fiction        131
Childrens       40
dtype: int64
```

In [542…
```
books['genre'].value_counts()
```

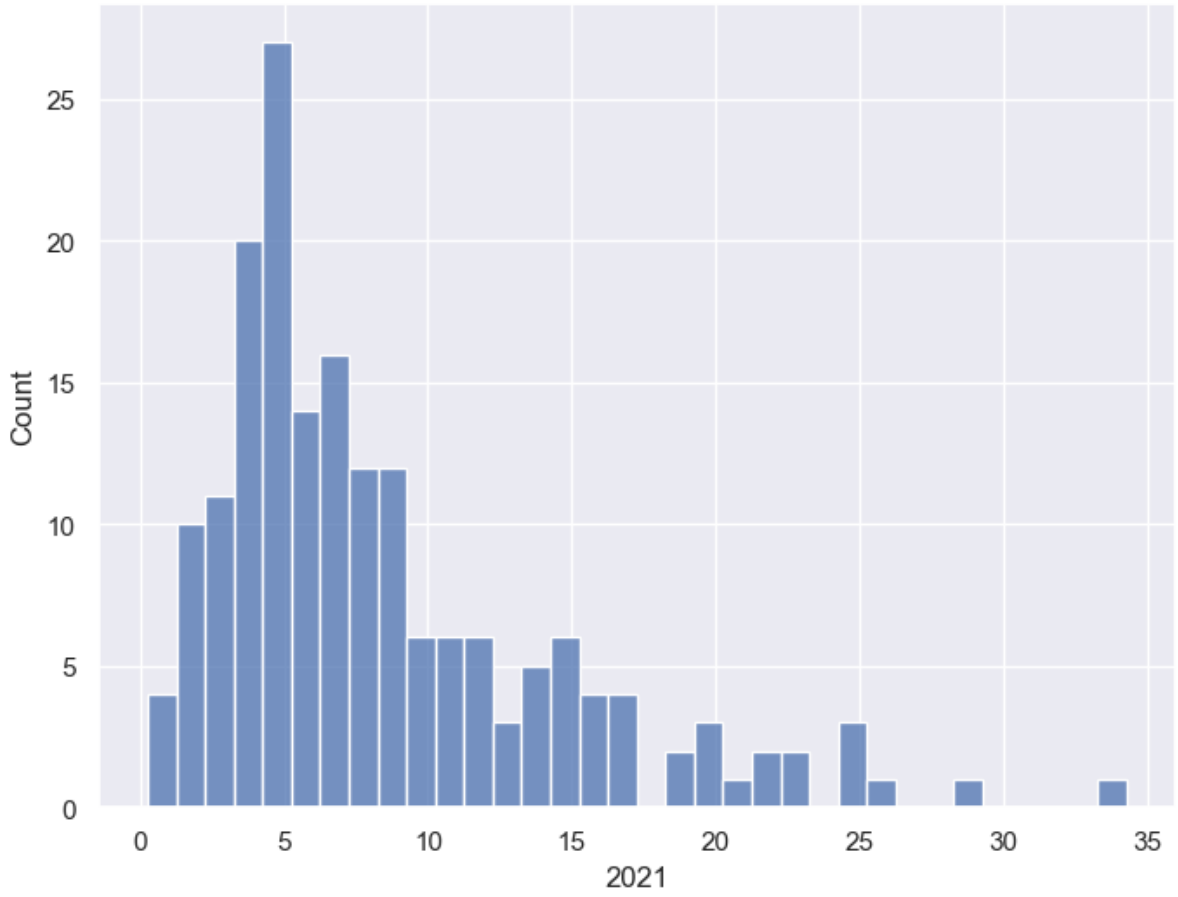
Out[542]:
```
Non Fiction    179
Fiction        131
Childrens       40
Name: genre, dtype: int64
```

In [543…
```
books.value_counts("genre")
```

Out[543]:
```
genre
Non Fiction    179
Fiction        131
Childrens       40
dtype: int64
```

1

In [544…
```
unemployment = pd.read_csv("clean_unemployment.csv")
sns.histplot(data=unemployment , x='2021' , binwidth = 1)
plt.show()
```



In [545…
```
books.dtypes
```

Out[545]:
```
name       object
author     object
rating    float64
year        int64
genre      object
dtype: object
```

In [546…
```
books["year"] = books["year"].astype(int)
books.dtypes
```

Out[546]:
```
name       object
author     object
rating    float64
year        int64
genre      object
dtype: object
```

Validating categorical data

In [547…
```
books["genre"].isin(["Fiction" , "Non Fiction"])
```

Out[547]:
```
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
```

In [548…
```
books[books["genre"].isin(["Fiction", "Non Fiction"])].head()
```

Out[548]:

	name	author	rating	year	genre
0	10-Day Green Smoothie Cleanse	JJ Smith	4.7	2016	Non Fiction
1	11/22/63: A Novel	Stephen King	4.6	2011	Fiction
2	12 Rules for Life: An Antidote to Chaos	Jordan B. Peterson	4.7	2018	Non Fiction
3	1984 (Signet Classics)	George Orwell	4.7	2017	Fiction
5	A Dance with Dragons (A Song of Ice and Fire)	George R. R. Martin	4.4	2011	Fiction

In [549…
```
books.select_dtypes("number").head()
```

Out[549]:

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

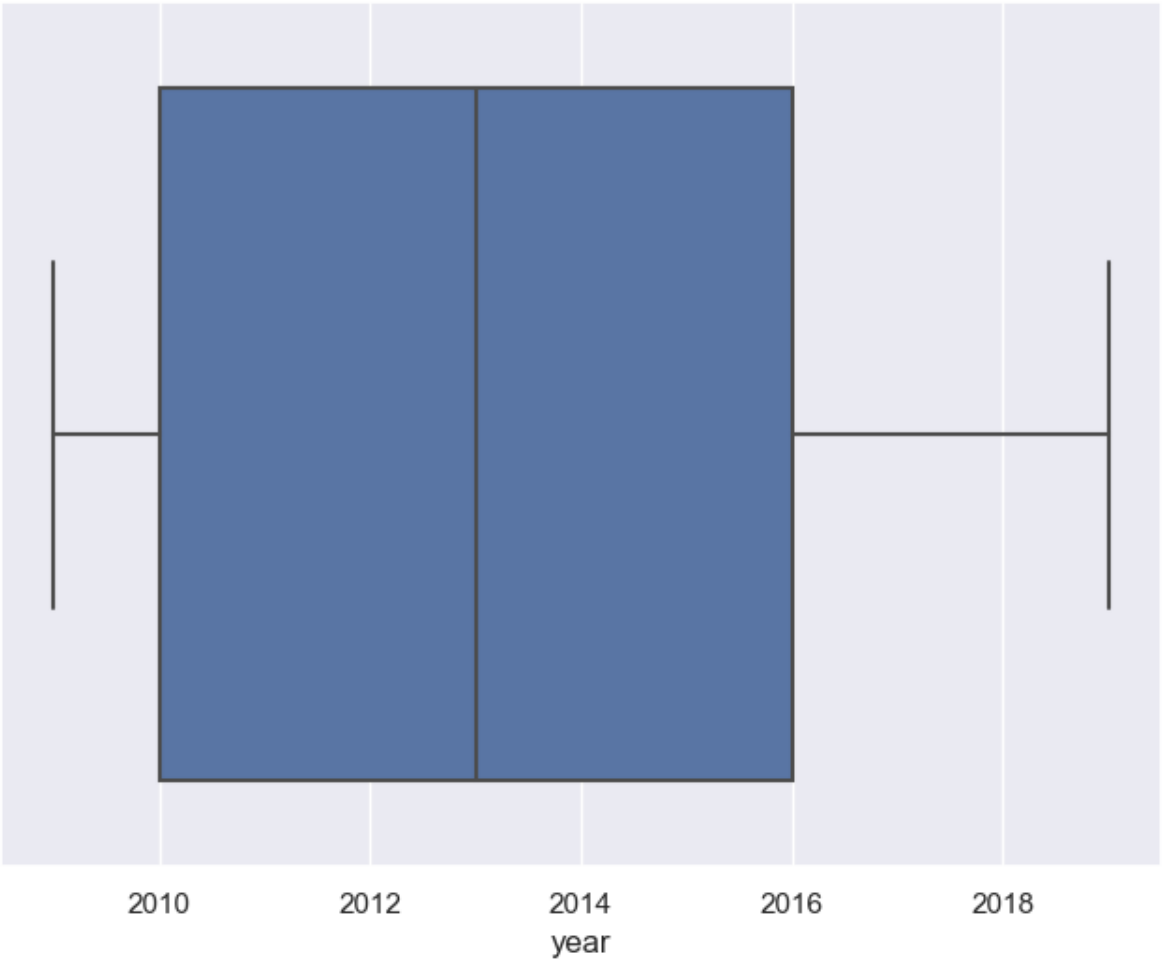
In [550…
```
books["year"].min()
```

Out[550]:
```
2009
```

In [551…
```
books["year"].max()
```

Out[551]:
```
2019
```

In [552…
```
sns.boxplot(data=books , x = "year")
plt.show()
```



In [553…
```
sns.boxplot(data = books , x="year" , y ="genre")
```

Out[553]:
```
<AxesSubplot:xlabel='year', ylabel='genre'>
```

2

In [554…
```
unemployment = pd.read_csv("clean_unemployment.csv")
not_oceania = ~unemployment['continent'].isin(["Oceania"])
not_oceania
```

Out[554]:
```
0       True
1       True
2       True
3       True
4       True
       ...
177    False
178     True
179     True
180     True
181     True
Name: continent, Length: 182, dtype: bool
```

3

In [555…
```
unemployment["2021"].min()
```

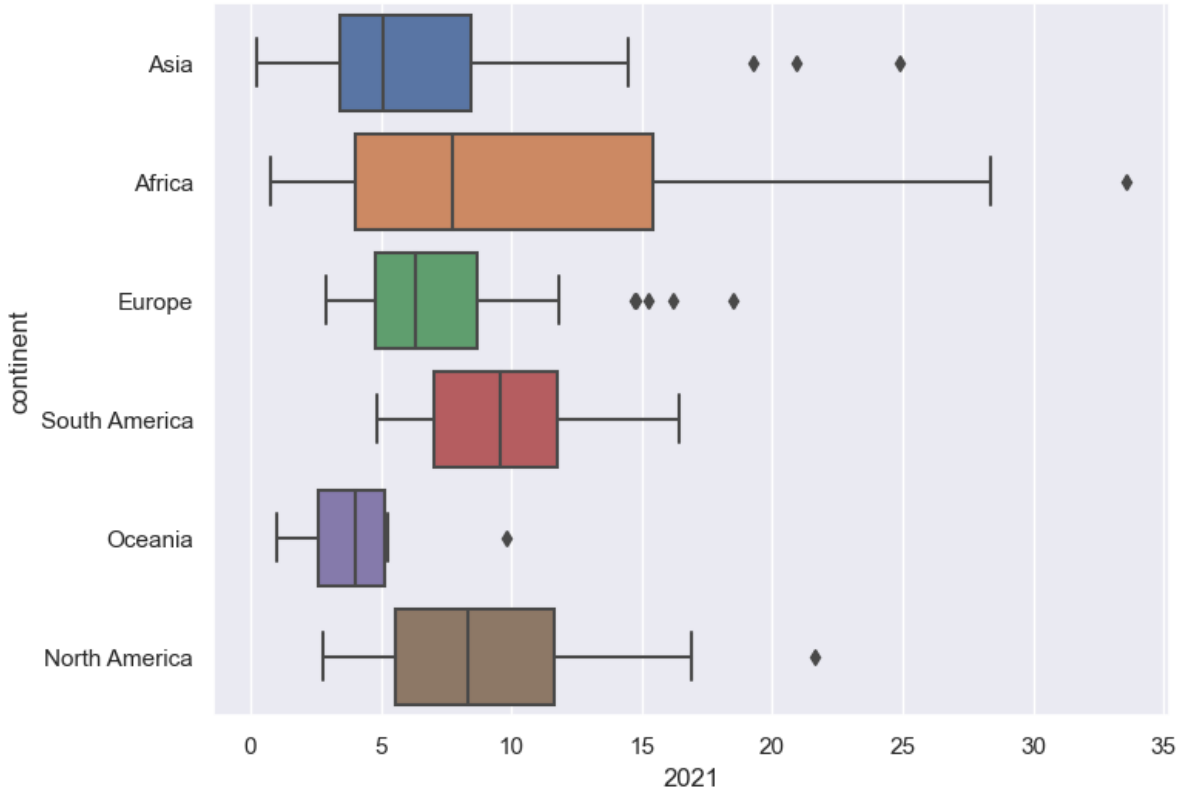
Out[555]:
```
0.26
```

In [556…
```
unemployment["2021"].max()
```

Out[556]:
```
33.56
```

In [557…
```
sns.boxplot(data=unemployment , x = "2021" , y="continent")
```

Out[557]: <AxesSubplot:xlabel='2021', ylabel='continent'>



Exploring groups of data

In [558…
```
books.groupby("genre").mean()
```

Out[558]:

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

In [559…
```
books.agg(["mean" , "std"])
```

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97859/1965544463.
py:1: FutureWarning: ['name', 'author', 'genre'] did not aggregate successfu
lly. If any error is raised this will raise in a future version of pandas. D
rop these columns/ops to avoid this warning.
  books.agg(["mean" , "std"])
```

Out[559]:

	rating	year
mean	4.608571	2013.508571
std	0.226941	3.284711

In [560…
```
books.agg({"rating": ["mean", "std"] , "year":["median"]})
```

Out[560]:

	rating	year
mean	4.608571	NaN
std	0.226941	NaN
median	NaN	2013.0

In [561…

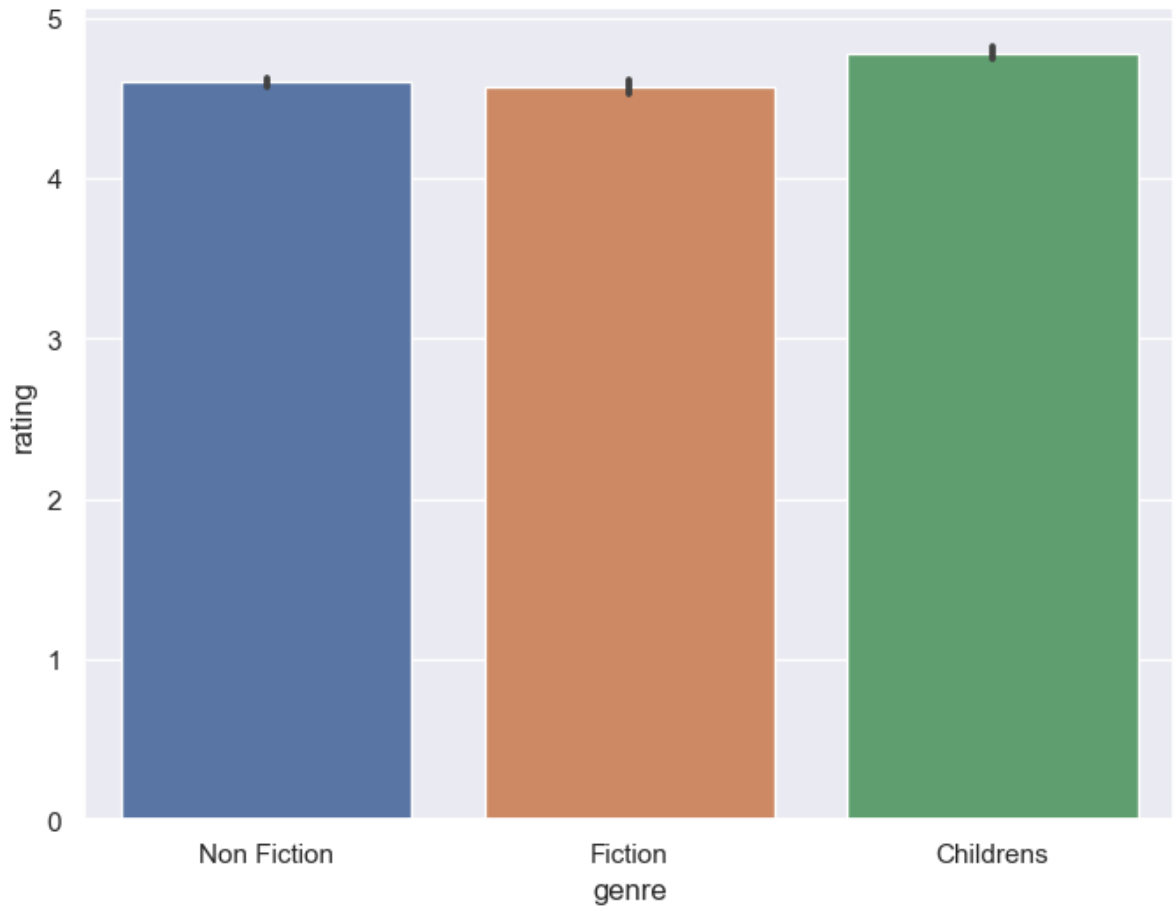
```
books.groupby("genre").agg(mean_rating=("rating", "mean"),
 std_rating = ("rating" , "std"),
 median_year =("year" , "median")
                            )
```

Out[561]:

genre	mean_rating	std_rating	median_year
Childrens	4.780000	0.122370	2015.0
Fiction	4.570229	0.281123	2013.0
Non Fiction	4.598324	0.179411	2013.0

In [562…

```
sns.barplot(data=books, x = "genre" ,y = "rating")
plt.show()
```



4

In [563…

```
unemployment.agg(["mean" , "std"])
```

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97859/2501974079.
py:1: FutureWarning: ['country_code', 'country_name', 'continent'] did not a
ggregate successfully. If any error is raised this will raise in a future ve
rsion of pandas. Drop these columns/ops to avoid this warning.
  unemployment.agg(["mean" , "std"])
```

Out[563]:

	2010	2011	2012	2013	2014	2015	2016	2017	
mean	8.409286	8.315440	8.317967	8.344780	8.179670	8.058901	7.925879	7.668626	7.
std	6.248887	6.266795	6.367270	6.416041	6.284241	6.161170	6.045439	5.902152	5

In [564…

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

```
/var/folders/4j/bnvctt7152z6l5l6szd4m7wh0000gn/T/ipykernel_97859/564087925.p
y:1: FutureWarning: ['country_code', 'country_name'] did not aggregate succe
ssfully. If any error is raised this will raise in a future version of panda
s. Drop these columns/ops to avoid this warning.
  unemployment.groupby("continent").agg(["mean" , "std"])
```

Out[564]:

	2010		2011		2012			
	mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	
continent								
Africa	9.343585	7.411259	9.369245	7.401556	9.240755	7.264542	9.132453	7.30
Asia	6.240638	5.146175	5.942128	4.779575	5.835319	4.756904	5.852128	4.66
Europe	11.008205	6.392063	10.947949	6.539538	11.325641	7.003527	11.466667	6.96
North America	8.663333	5.115805	8.563333	5.377041	8.448889	5.495819	8.840556	6.08
Oceania	3.622500	2.054721	3.647500	2.008466	4.103750	2.723118	3.980000	2.64
South America	6.870833	2.807058	6.518333	2.801577	6.410833	2.936508	6.335000	2.80

6 rows × 24 columns

5

In [565…

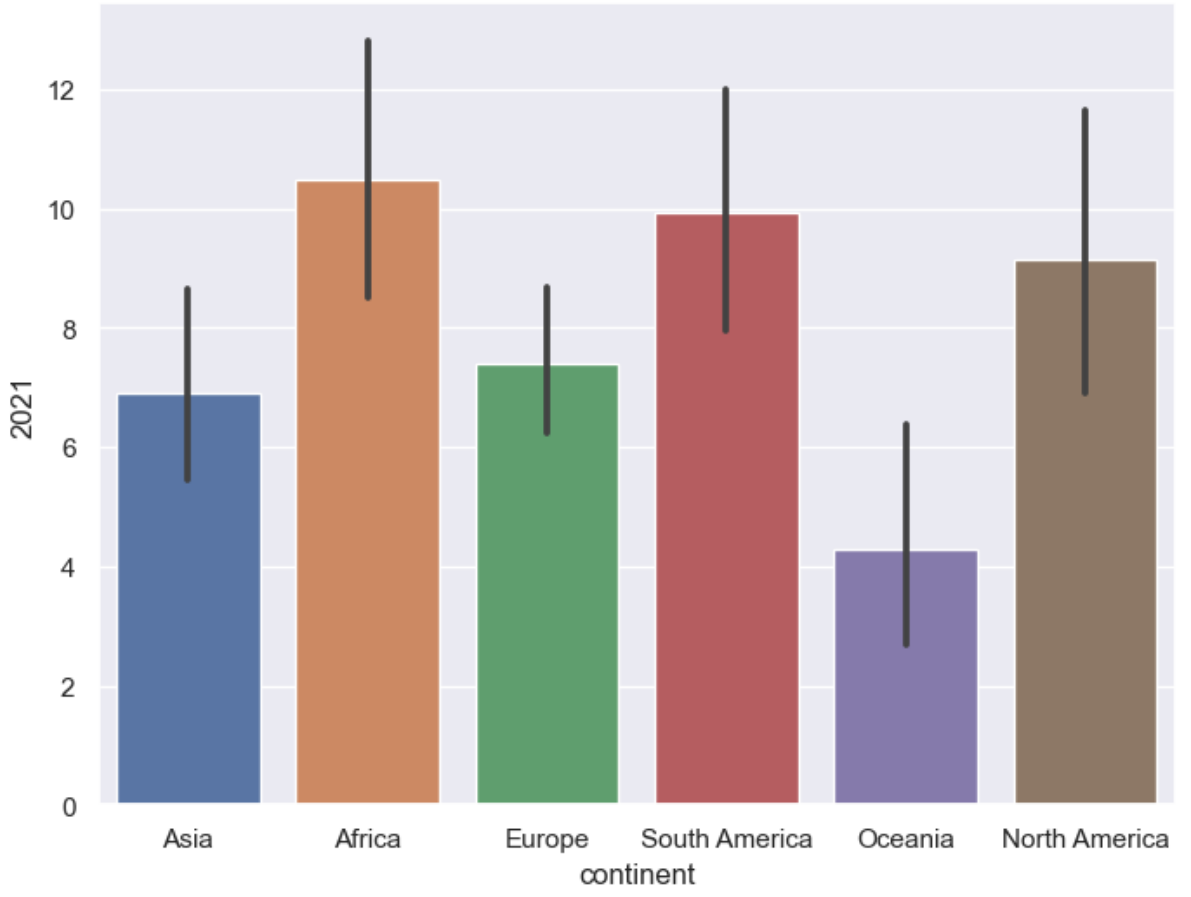
```
continent_summary = unemployment.groupby("continent").agg(mean_rate_2021=("2
                                                          std_rate = ("2021"
continent_summary
```

Out[565]:

continent	mean_rate_2021	std_rate
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

6

In [566…
```
sns.barplot(data=unemployment, x = "continent" ,y = "2021")
plt.show()
```



In [567…
```
salaries = pd.read_csv("ds_salaries_clean.csv")
```

In [568…
```
print(salaries.isna().sum())
```

```
Working_Year            0
Designation             0
Experience              0
Employment_Status       0
Employee_Location       0
Company_Size            0
Remote_Working_Ratio    0
Salary_USD              0
dtype: int64
```

In [569…
```
threshold = len(salaries) * 0.05
print(threshold)
```

```
30.35
```

In [570…
```
cols_to_drop = salaries.columns[salaries.isna().sum() <= threshold]
print(cols_to_drop)
```

```
Index(['Working_Year', 'Designation', 'Experience', 'Employment_Status',
       'Employee_Location', 'Company_Size', 'Remote_Working_Ratio',
       'Salary_USD'],
      dtype='object')
```

In [571…
```
salaries.dropna(subset = cols_to_drop , inplace = True)
```

In [572…
```
cols_with_missing_values = salaries.columns[salaries.isna().sum() > 0]
print(cols_with_missing_values)
```

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

In [573…
```
for col in cols_with_missing_values[:-1]:
    salaries[col].fillna(salaries[col].mode()[0])
```

In [574…
```
print(salaries.isna().sum())
```

```
Working_Year            0
Designation             0
Experience              0
Employment_Status       0
Employee_Location       0
Company_Size            0
Remote_Working_Ratio    0
Salary_USD              0
dtype: int64
```

In [575…
```
salaries_dict = salaries.groupby("Experience")["Salary_USD"].median().to_dic
print(salaries_dict)
```

```
{'Entry': 53948.0, 'Executive': 163694.5, 'Mid': 73465.0, 'Senior': 129380.
0}
```

In [576…
```
salaries["Salary_USD"] = salaries["Salary_USD"].fillna(salaries["Experience"
print(salaries.isna().sum())
```

```
Working_Year            0
Designation             0
Experience              0
Employment_Status       0
Employee_Location       0
Company_Size            0
Remote_Working_Ratio    0
Salary_USD              0
dtype: int64
```

7

In [577…
```
planes = pd.read_csv("Airlines_unclean.csv" , index_col = 0)
print(planes.isna().sum())
```

```
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
```

In [578…
```
threshold = len(planes) * 0.05
print(threshold)
```

```
533.0
```

In [579…
```
cols_to_drop = planes.columns[planes.isna().sum() <= threshold]
print(cols_to_drop)
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops'],
      dtype='object')
```

In [580…
```
planes.dropna(subset = cols_to_drop , inplace = True)
```

In [581…
```
cols_with_missing_values = planes.columns[planes.isna().sum() > 0]
print(cols_with_missing_values)
```

```
Index(['Additional_Info', 'Price'], dtype='object')
```

In [582…
```
for col in cols_with_missing_values[:-1]:
    planes[col].fillna(planes[col].mode()[0])
```

In [583…
```
print(planes.isna().sum())
```

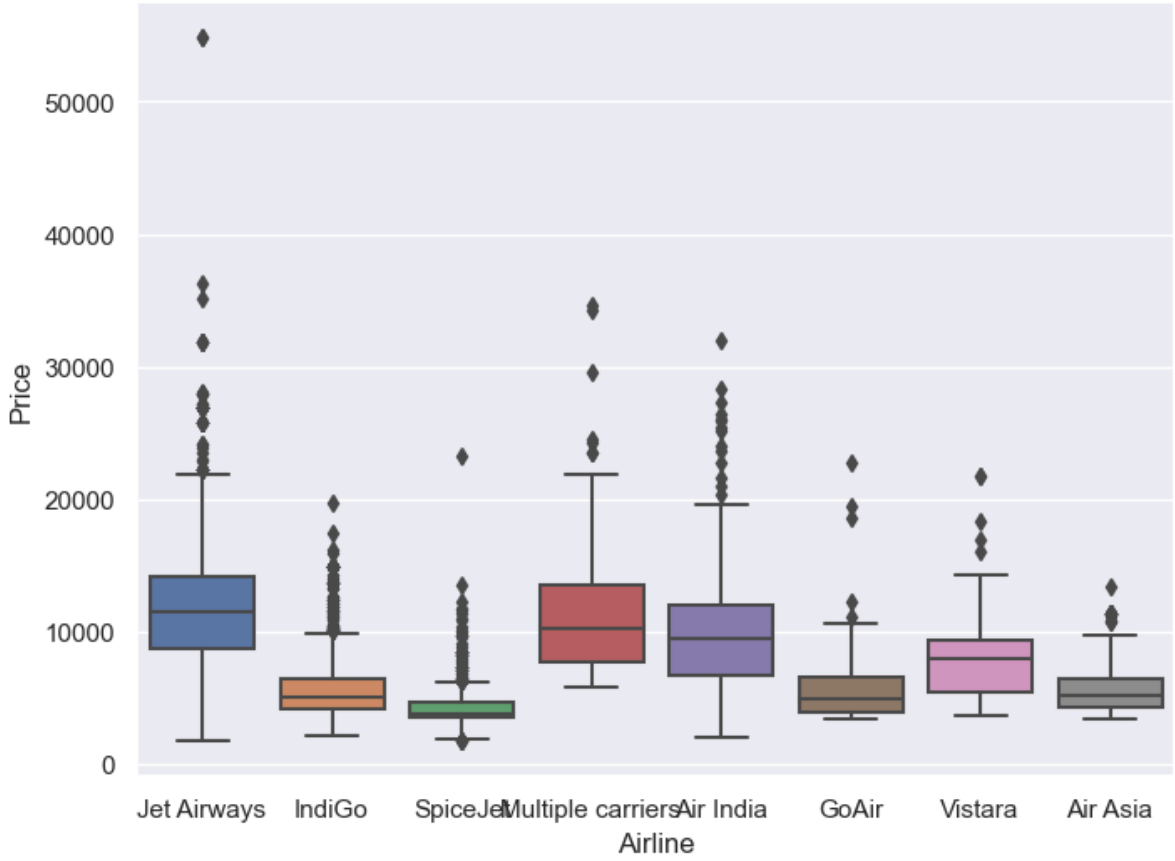
```
Airline              0
Date_of_Journey      0
Source               0
Destination          0
Route                0
Dep_Time             0
Arrival_Time         0
Duration             0
Total_Stops          0
Additional_Info    300
Price              368
dtype: int64
```

Strategies for remaining missing data.

In [584…
```
# 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()
```

```
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: Additional_Info, dtype: int64
```



8

In [585…

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


print(planes_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, 'V
istara': 8028.0}
```

In [586…

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

```
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
```

WORKING WITH CATEGORICAL DATA

In [587…
```
salaries = pd.read_csv("ds_salaries_clean.csv")
print(salaries.select_dtypes("object").head())
```

```
                  Designation Experience Employment_Status Employee_Location
\
0              Data Scientist        Mid                FT                DE
1  Machine Learning Scientist     Senior                FT                JP
2           Big Data Engineer     Senior                FT                GB
3        Product Data Analyst        Mid                FT                HN
4   Machine Learning Engineer     Senior                FT                US

  Company_Size
0            L
1            S
2            M
3            S
4            L
```

In [588…
```
print(salaries["Designation"].value_counts())
```

```
Data Scientist                                143
Data Engineer                                 132
Data Analyst                                   97
Machine Learning Engineer                      41
Research Scientist                             16
Data Science Manager                           12
Data Architect                                 11
Big Data Engineer                               8
Machine Learning Scientist                      8
Principal Data Scientist                        7
AI Scientist                                    7
Data Science Consultant                         7
Director of Data Science                        7
Data Analytics Manager                          7
ML Engineer                                     6
Computer Vision Engineer                        6
BI Data Analyst                                 6
Lead Data Engineer                              6
Data Engineering Manager                        5
Business Data Analyst                           5
Head of Data                                    5
Applied Data Scientist                          5
Applied Machine Learning Scientist              4
Head of Data Science                            4
Analytics Engineer                              4
Data Analytics Engineer                         4
Machine Learning Developer                      3
Machine Learning Infrastructure Engineer        3
Lead Data Scientist                             3
Computer Vision Software Engineer               3
Lead Data Analyst                               3
Data Science Engineer                           3
Principal Data Engineer                         3
Principal Data Analyst                          2
ETL Developer                                   2
Product Data Analyst                            2
Director of Data Engineering                    2
Financial Data Analyst                          2
Cloud Data Engineer                             2
Lead Machine Learning Engineer                  1
NLP Engineer                                    1
Head of Machine Learning                        1
3D Computer Vision Researcher                   1
Data Specialist                                 1
Staff Data Scientist                            1
Big Data Architect                              1
Finance Data Analyst                            1
Marketing Data Analyst                          1
Machine Learning Manager                        1
Data Analytics Lead                             1
Name: Designation, dtype: int64
```

In [589…

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

```
50
```

In [590…

```
designation_counts = salaries["Designation"].value_counts()
top_five_designations = designation_counts.head(5)

# Plot the bar chart
plt.bar(top_five_designations.index, top_five_designations.values)


# colors = ["blue", "orange", "green", "red", "purple"]
```

```
# # Plot the bar chart with colors
# plt.bar(top_five_designations.index, top_five_designations.values, color=c

# Add labels and title
plt.ylabel("Number of professionals")
plt.xlabel("Count")
plt.title("Top Five most common data professionals job titles")

# Display the plot
plt.show()
```

In [591…
```
salaries["Designation"].str.contains("Scientist")
```

Out[591]:
```
0       True
1       True
2      False
3      False
4      False
       ...
602    False
603    False
604    False
605    False
606     True
Name: Designation, Length: 607, dtype: bool
```

In [592…
```
salaries["Designation"].str.contains("Machine Learning|AI")
```

Out[592]:
```
0      False
1       True
2      False
3      False
4       True
       ...
602    False
603    False
604    False
605    False
606     True
Name: Designation, Length: 607, dtype: bool
```

In [593…
```
job_categories = ["Data Science", "Data Analytics" , "Data Engineering" , "M
```

In [594…
```
data_science = "Data Scientist|NLP"
data_analyst = "Ana;yst|Analytics"
data_engineer = "Data Engineer|ETL|Architect|Infrastructure"
ml_engineer = "Machine Learning|ML|Big Data|AI"
manager = "Manager|Head|Directore|Lead|Principal|Staff"
consultant = "Consultant|Freelance"
```

In [595…
```
conditions = [
(salaries ["Designation"].str.contains(data_science)),
    (salaries["Designation"].str.contains(data_analyst)),
    (salaries["Designation"].str.contains(data_engineer)),
    (salaries["Designation"].str.contains(ml_engineer)),
    (salaries["Designation"].str.contains(manager)),
    (salaries["Designation"].str.contains(consultant))
]
conditions
```

Out[595]:
```
[0      True
 1     False
 2     False
 3     False
 4     False
       ...
602    False
603    False
604    False
605    False
606    False
Name: Designation, Length: 607, dtype: bool,
0     False
1     False
2     False
3     False
4     False
      ...
602   False
603   False
604   False
605   False
606   False
Name: Designation, Length: 607, dtype: bool,
0     False
1     False
2      True
3     False
4     False
      ...
602    True
603    True
604   False
605   False
606   False
Name: Designation, Length: 607, dtype: bool,
0     False
1      True
2      True
3     False
4      True
      ...
602   False
603   False
604   False
605   False
606    True
Name: Designation, Length: 607, dtype: bool,
0     False
1     False
2     False
3     False
4     False
      ...
602   False
603   False
604   False
605   False
606   False
Name: Designation, Length: 607, dtype: bool,
0     False
1     False
2     False
3     False
```

```
4      False
       ...
602    False
603    False
604    False
605    False
606    False
Name: Designation, Length: 607, dtype: bool]
```

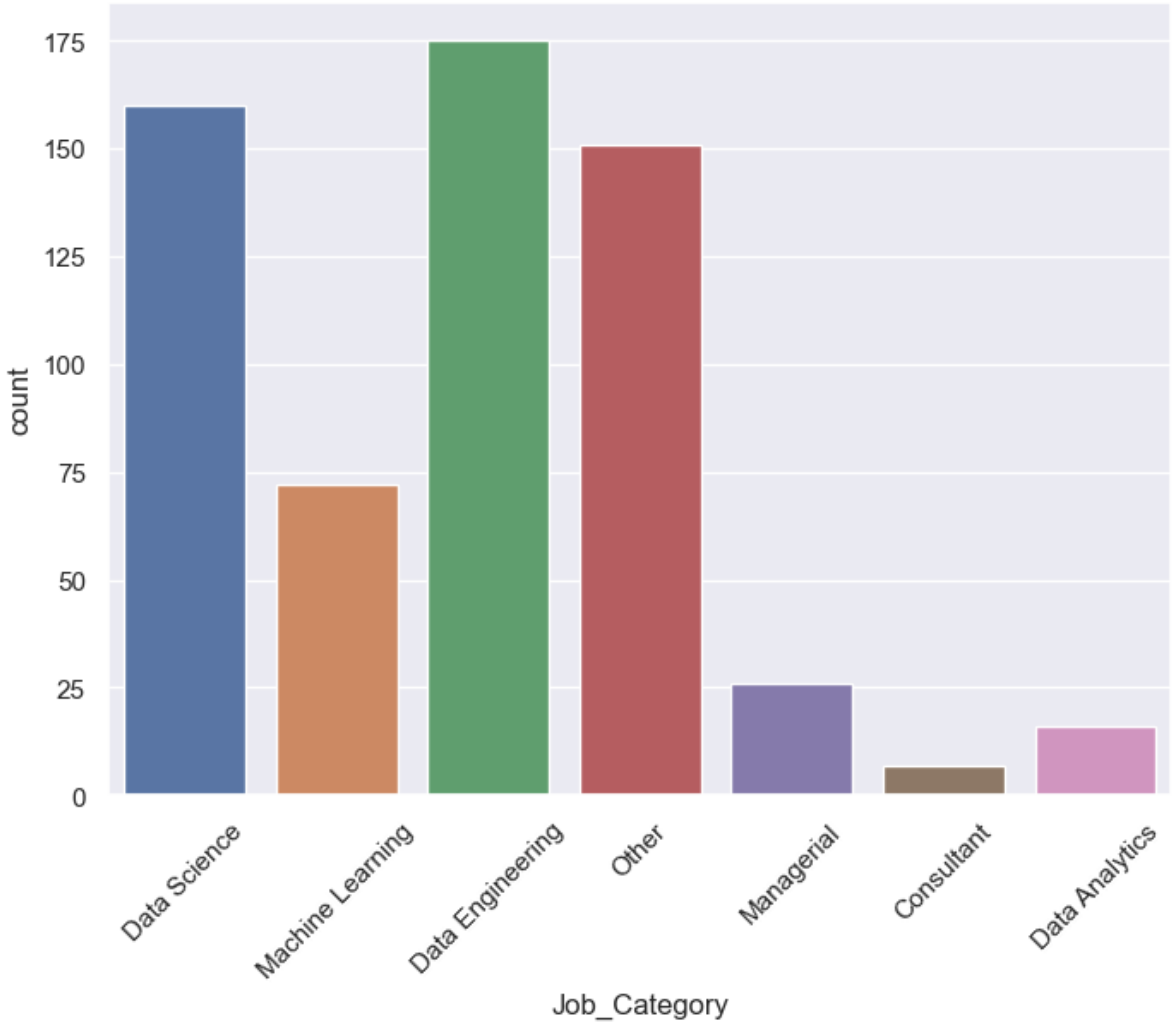
In [596…
```
import numpy as np
salaries["Job_Category"] = np.select(conditions, job_categories, default = "
salaries["Job_Category"]
```

Out[596]:
```
0          Data Science
1      Machine Learning
2      Data Engineering
3                 Other
4      Machine Learning
             ...
602    Data Engineering
603    Data Engineering
604               Other
605               Other
606    Machine Learning
Name: Job_Category, Length: 607, dtype: object
```

In [597…
```
print(salaries[["Designation", "Job_Category"]].head())
```

```
                  Designation      Job_Category
0              Data Scientist      Data Science
1  Machine Learning Scientist  Machine Learning
2           Big Data Engineer  Data Engineering
3        Product Data Analyst             Other
4   Machine Learning Engineer  Machine Learning
```

In [598…
```
sns.countplot(data=salaries, x="Job_Category")
plt.xticks(rotation=45)
plt.show()
```



In [599…
```
# planes = pd.read_csv("Airlines_unclean.csv")
# non_numeric = planes.select_dtypes("object")
# for col in non_numeric.columns:
#     print(f"Number of unique values in {col} column: ", non_numeric[col].n
```

In [600…
```
# planes["Duration"].head()
```

9

In [601…
```
import numpy as np
import pandas as pd


categorie_values = ["short_flights" , "medium_flights" , "long_flights"]


short_flights = "0h|1h|2h|3h|4h"
medium_flights = "5h|6h|7h|8h|9h"
long_flights = "10h|11h|12h|13h|14h|15h|16h"

condition = [
    (planes["Duration"].str.contains(short_flights)),
    (planes["Duration"].str.contains(medium_flights)),
    (planes["Duration"].str.contains(long_flights))
]
```

```
planes['Duration_Category'] = np.select(condition, categorie_values, default

print(planes[["Duration","Duration_Category"]])
```

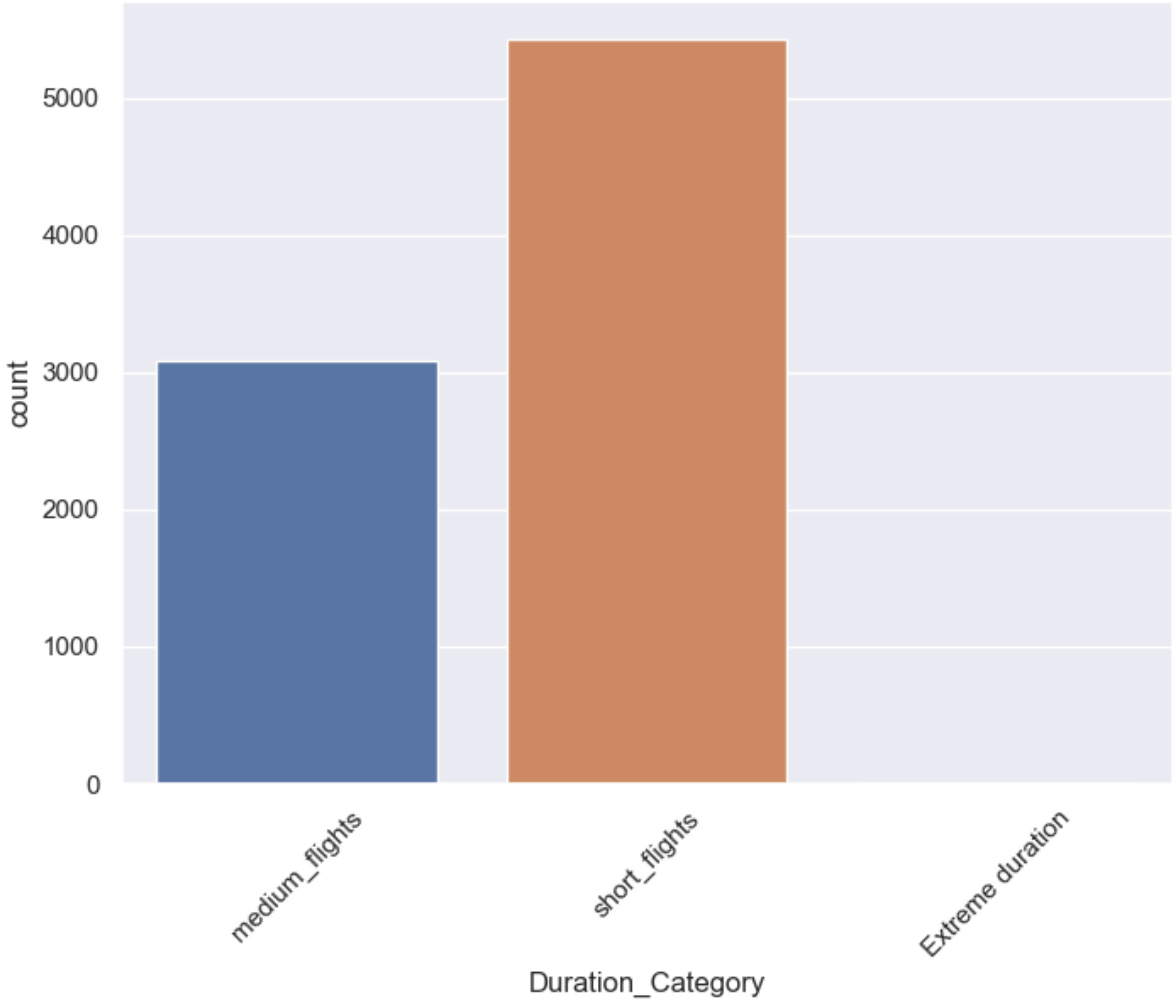
```
      Duration Duration_Category
0          19h    medium_flights
1        5h 25m    medium_flights
2        4h 45m     short_flights
3        2h 25m     short_flights
4       15h 30m    medium_flights
...         ...               ...
10654    2h 40m     short_flights
10655    2h 30m     short_flights
10656    2h 35m     short_flights
10658    2h 40m     short_flights
10659    8h 20m    medium_flights

[8508 rows x 2 columns]
```

10

In [602…

```
sns.countplot(data=planes, x="Duration_Category")
plt.xticks(rotation=45)
plt.show()
```



WORKING WITH NUMERICAL DATA

In [603…
```
# import pandas as pd
# salaries = pd.read_csv("Salary_Rupee_USD.csv", index_col = 0)
# salaries["Salary_In_Rupees"] = salaries["Salary_In_Rupees"].str.replace(",
# print(salaries["Salary_In_Rupees" ,"Salary_USD"].head())
```

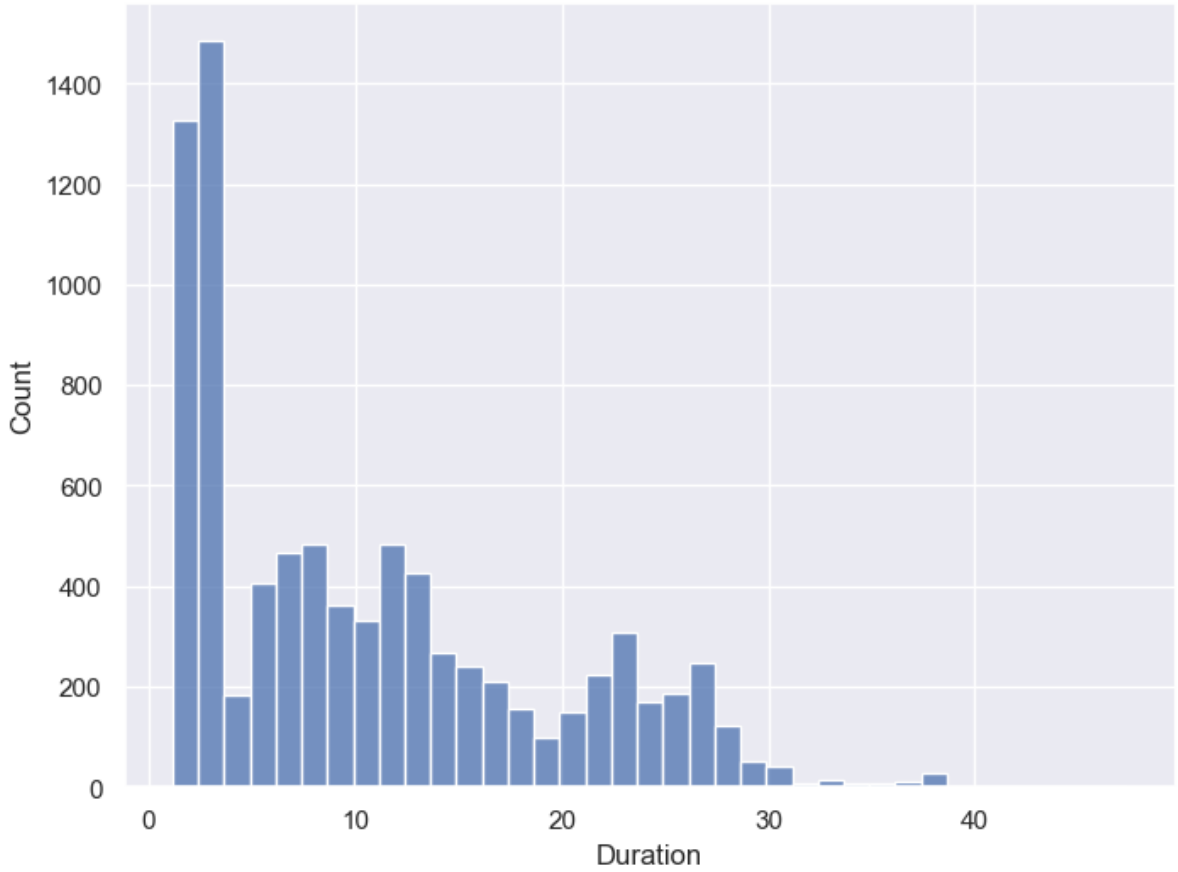
11

In [604…
```
planes["Duration"] = planes["Duration"].str.replace("h", ".")
planes["Duration"] = planes["Duration"].str.replace("m", "")
planes["Duration"] = planes["Duration"].str.replace(" ", "")
planes["Duration"] = planes["Duration"].astype(float)
```

In [605…
```
print(planes["Duration"].head())
```

```
0    19.00
1     5.25
2     4.45
3     2.25
4    15.30
Name: Duration, dtype: float64
```

In [606…
```
sns.histplot(data=planes, x = "Duration")
plt.show()
```



12

In [607…
```
planes["airline_median_duration"] = planes.groupby("Airline")["Duration"].tr
print(planes[["Airline", "airline_median_duration"]].value_counts())
```

```
Airline            airline_median_duration
Jet Airways        13.20                      3082
IndiGo             2.55                       1632
Air India          15.50                      1399
Multiple carriers  10.15                       959
SpiceJet           2.30                        653
Vistara            3.10                        376
Air Asia           2.50                        260
GoAir              2.55                        147
dtype: int64
```

In [608…
```
planes["airline_mean_duration"] = planes.groupby("Destination")["Price"].tra
print(planes[["Destination", "airline_mean_duration"]].value_counts())
```

```
Destination  airline_mean_duration
Cochin       10262.0                3631
Banglore     9345.0                 2291
Delhi        4823.0                  998
New Delhi    10948.0                 720
Hyderabad    3855.5                  562
Kolkata      3850.0                  306
dtype: int64
```

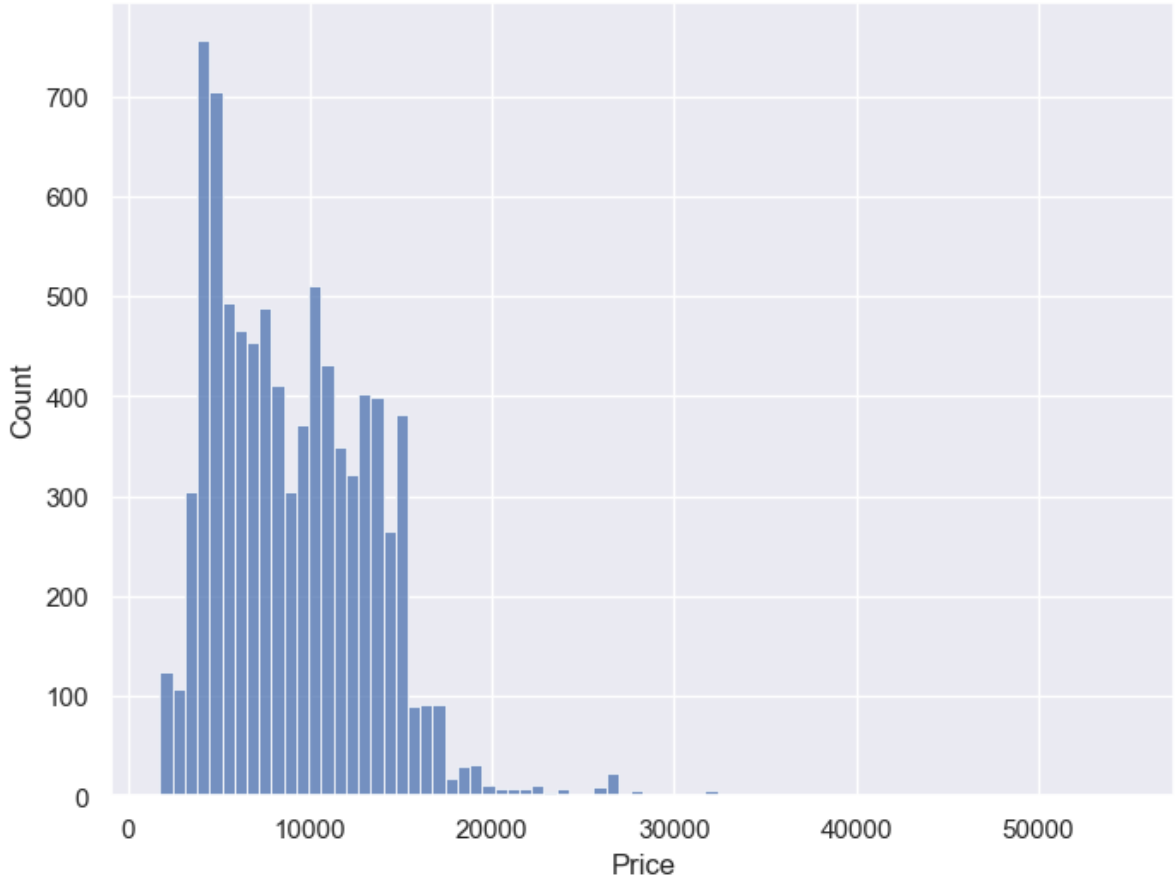
13

In [609…
```
import seaborn as sns


sns.histplot(data=planes, x="Price")

print(planes["Duration"].describe())
```

```
count    8508.000000
mean       10.566984
std         8.489069
min         1.150000
25%         2.500000
50%         8.400000
75%        15.400000
max        47.400000
Name: Duration, dtype: float64
```



14

In [610…
```
seventy_fifth = planes["Price"].quantile(0.75)
twenty_fifth = planes["Price"].quantile(0.25)
seventy_fifth
```

Out[610]:
```
12242.0
```

In [611…
```
twenty_fifth
```

Out[611]:
```
5228.0
```

In [612…
```
price_iqr = seventy_fifth - twenty_fifth
print(price_iqr)
```

```
7014.0
```

In [613…
```
upper = seventy_fifth + (1.5 * price_iqr)
lower = twenty_fifth - (1.5 * price_iqr)
print(upper, lower)
```

```
22763.0 -5293.0
```

In [615…
```
planes = planes[(planes["Price"] > lower) & (planes["Price"] < upper) ]
print(planes["Price"].describe())
```

```
count     8438.000000
mean      8877.466046
std       4001.838236
min       1759.000000
25%       5224.000000
50%       8372.000000
75%      12121.000000
max      22270.000000
Name: Price, dtype: float64
```

In [ ]: