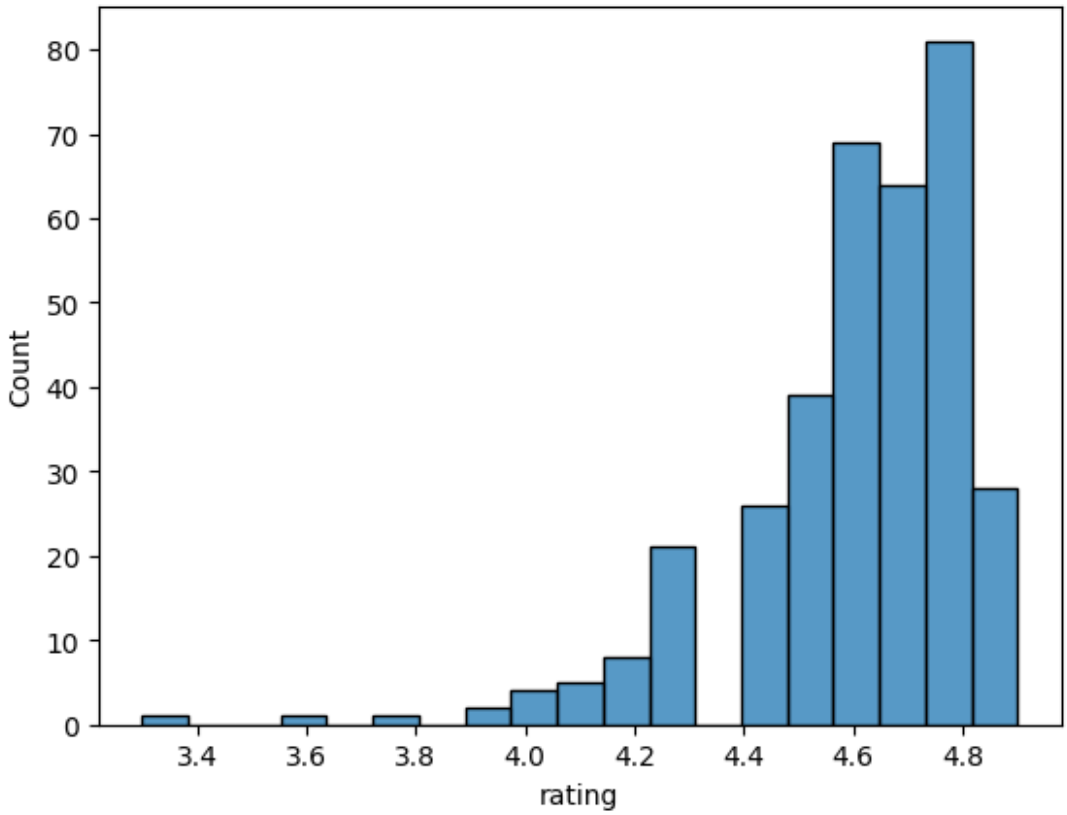
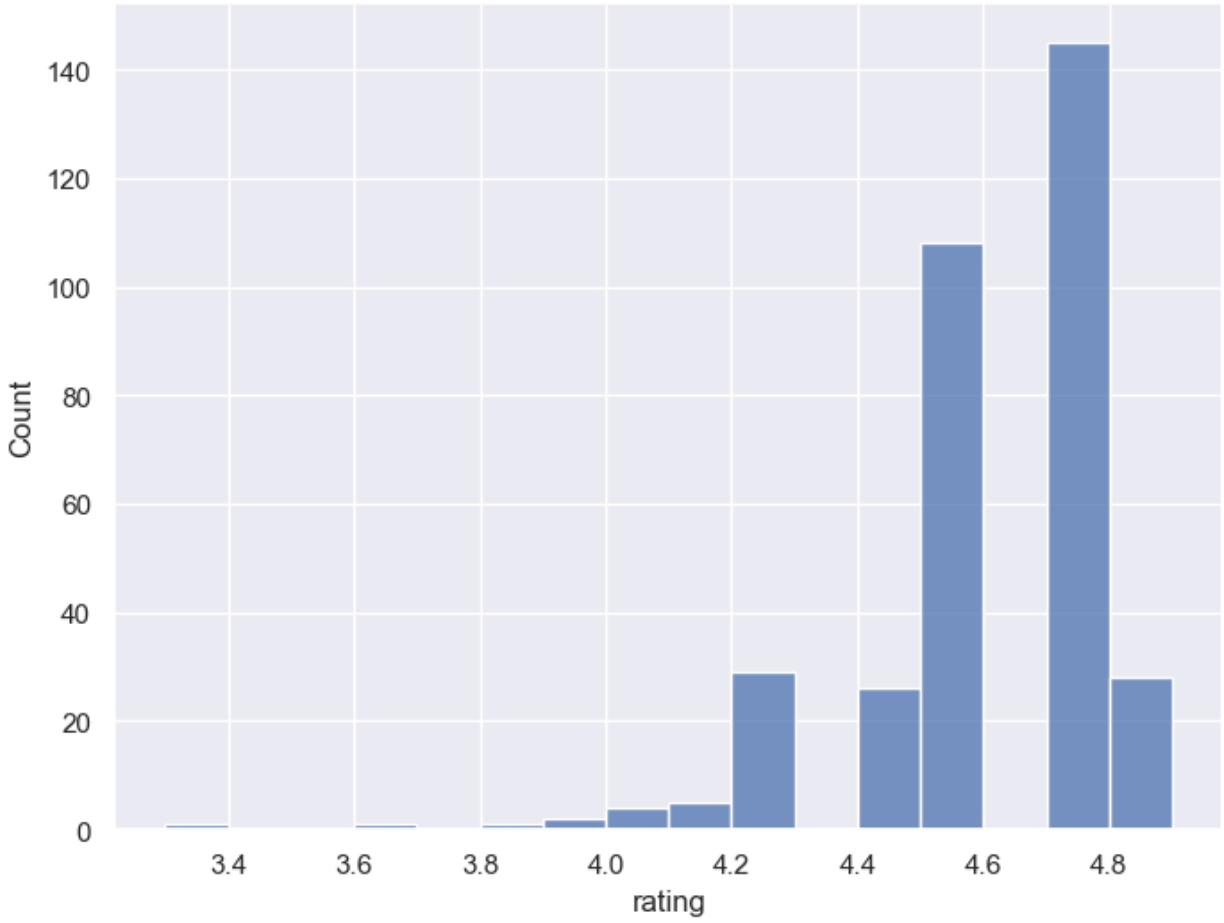
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
books = pd.read_csv('clean_books.csv')
sns.histplot(data=books,x='rating')
plt.show()
```




```
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                                11/22/63: 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
```

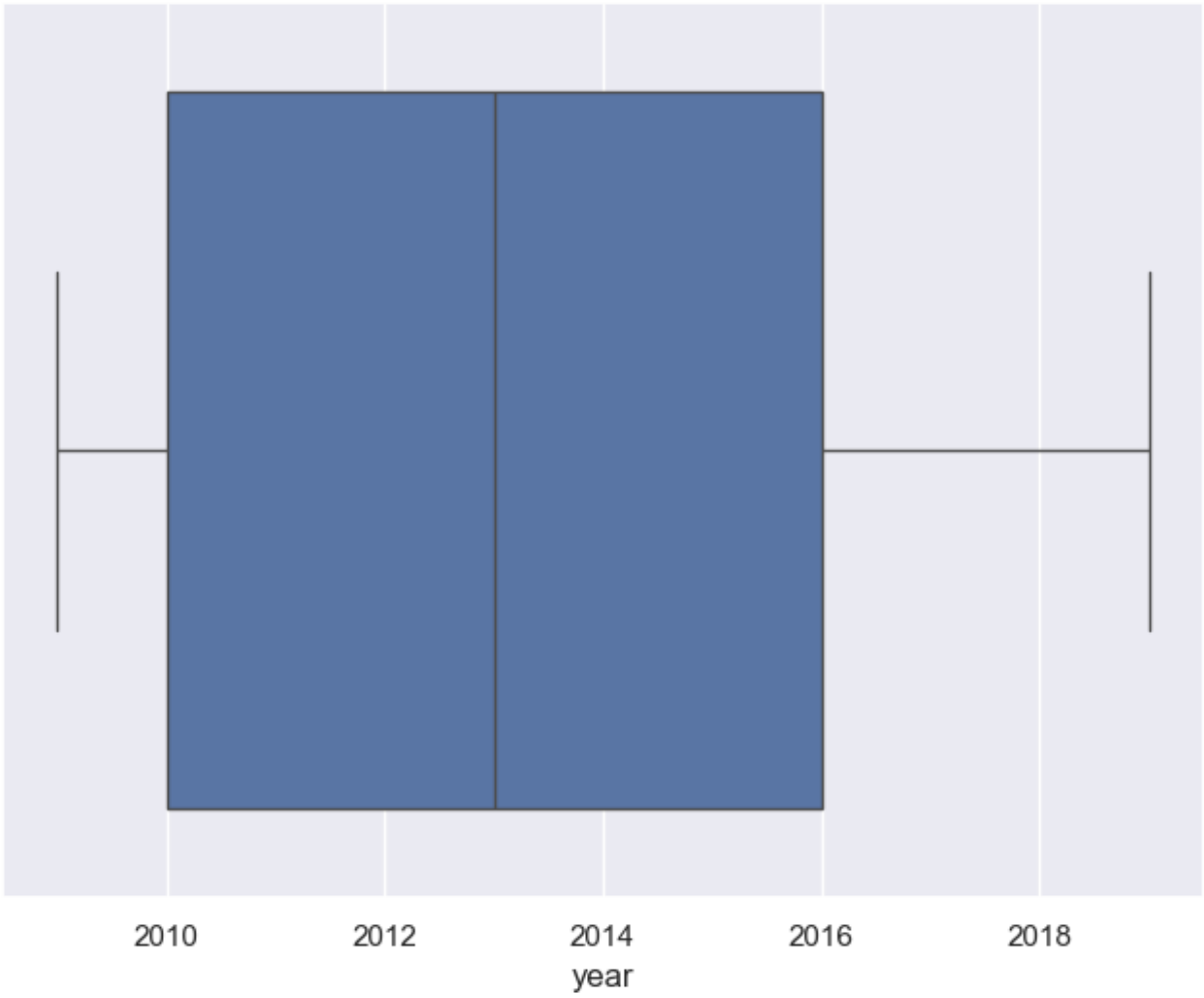
```
books["year"].min()
```

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

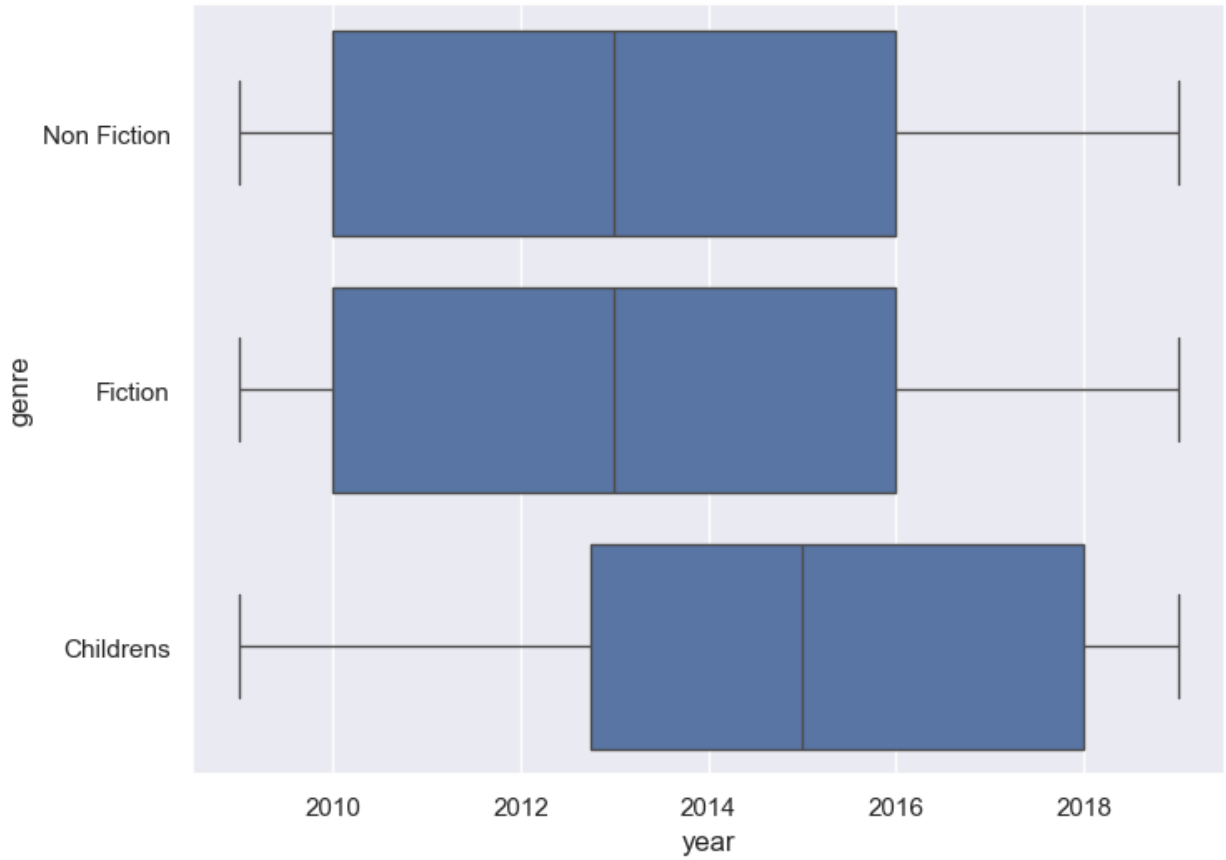
```
books["year"].max()
```

```
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()
```

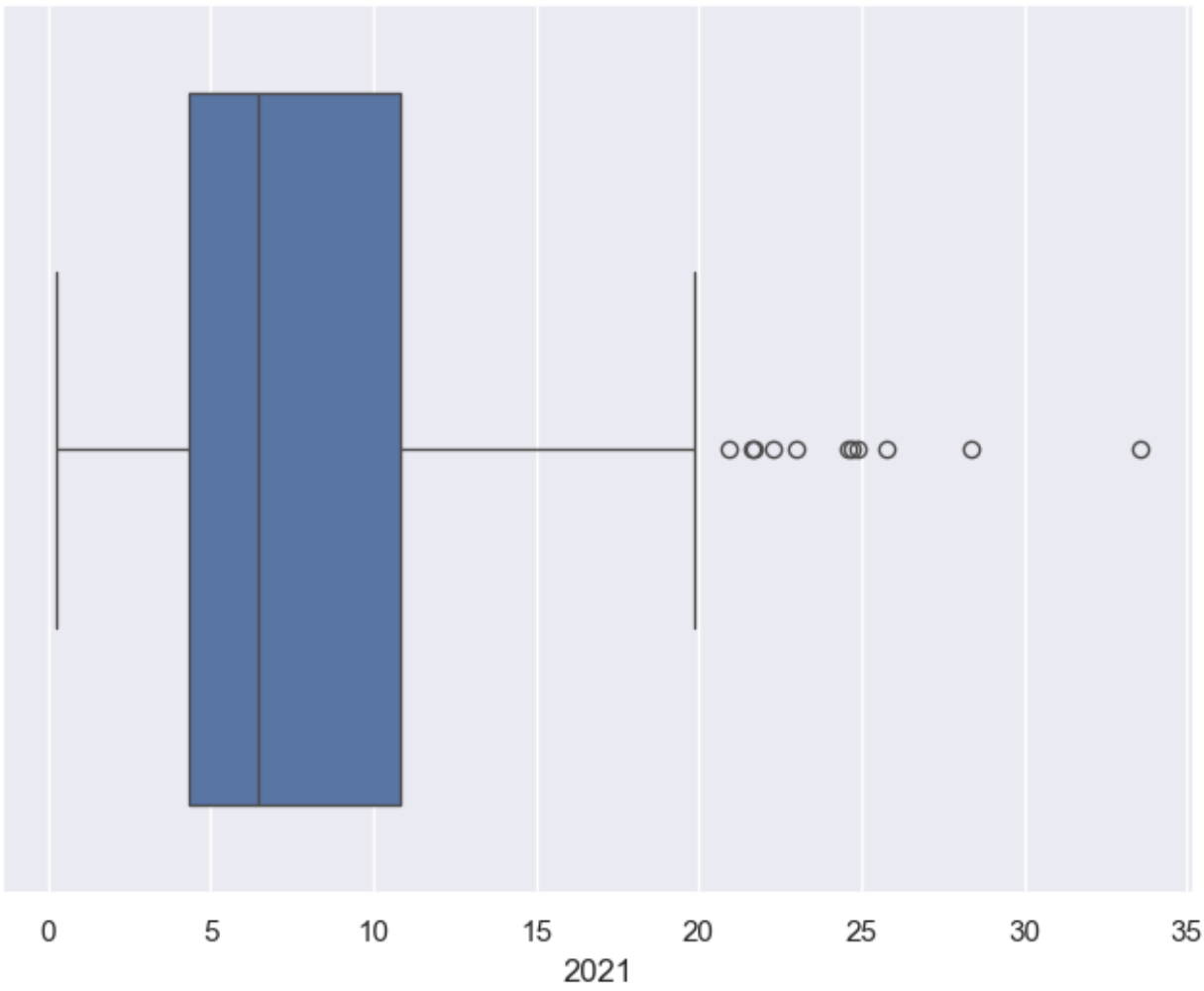
```
np.float64(33.56)
```

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

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

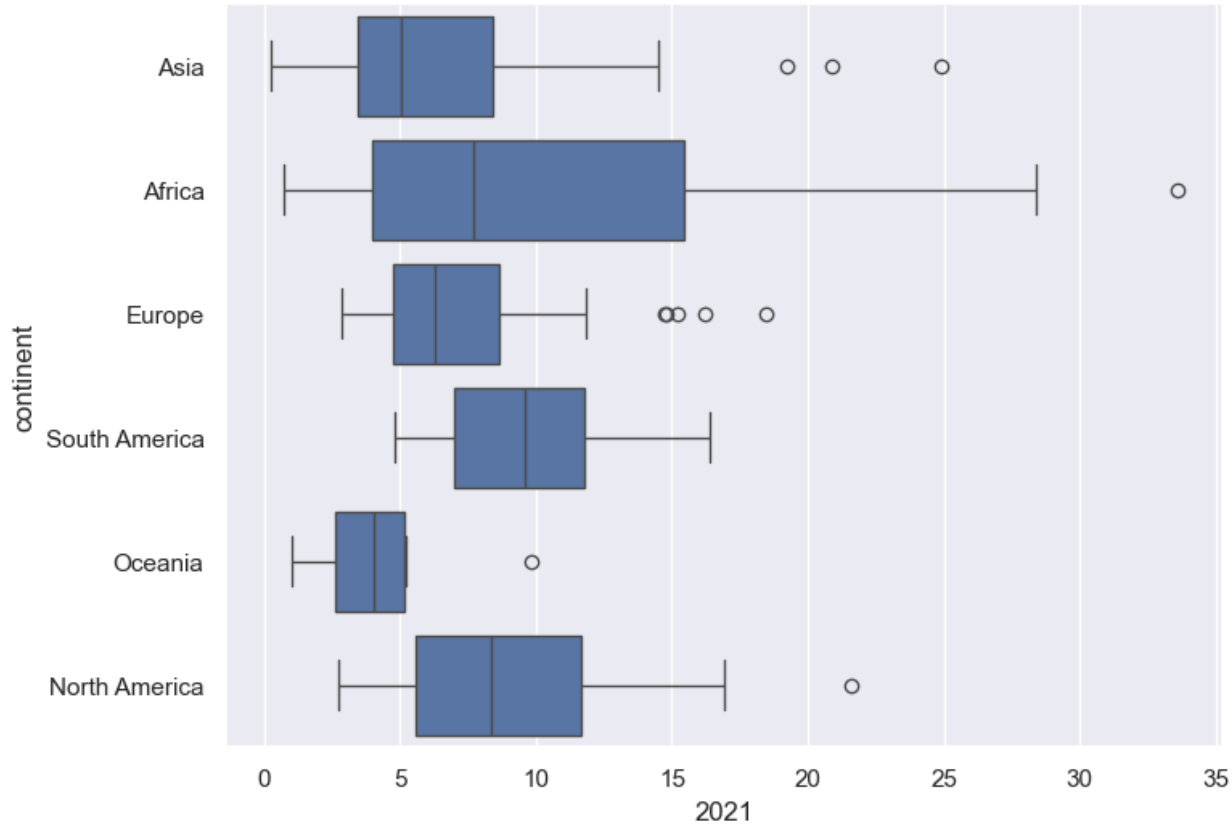
```
sns.boxplot(data=unemployment,x="2021")
```

```
<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
```

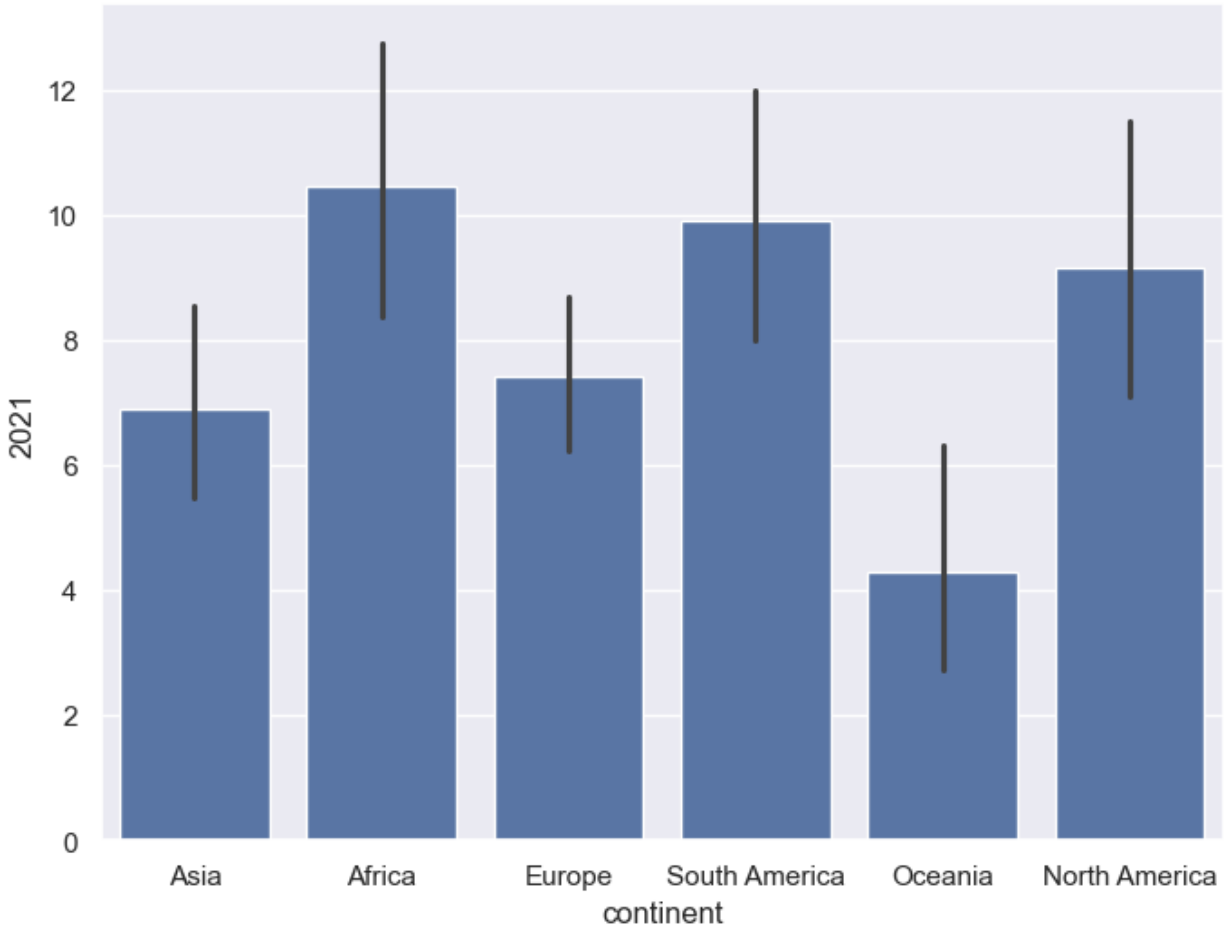
```
Europe         11.466667  6.969209  10.971282  6.759765  ...  8.359744

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_clean.csv")
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
```

```
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
```

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

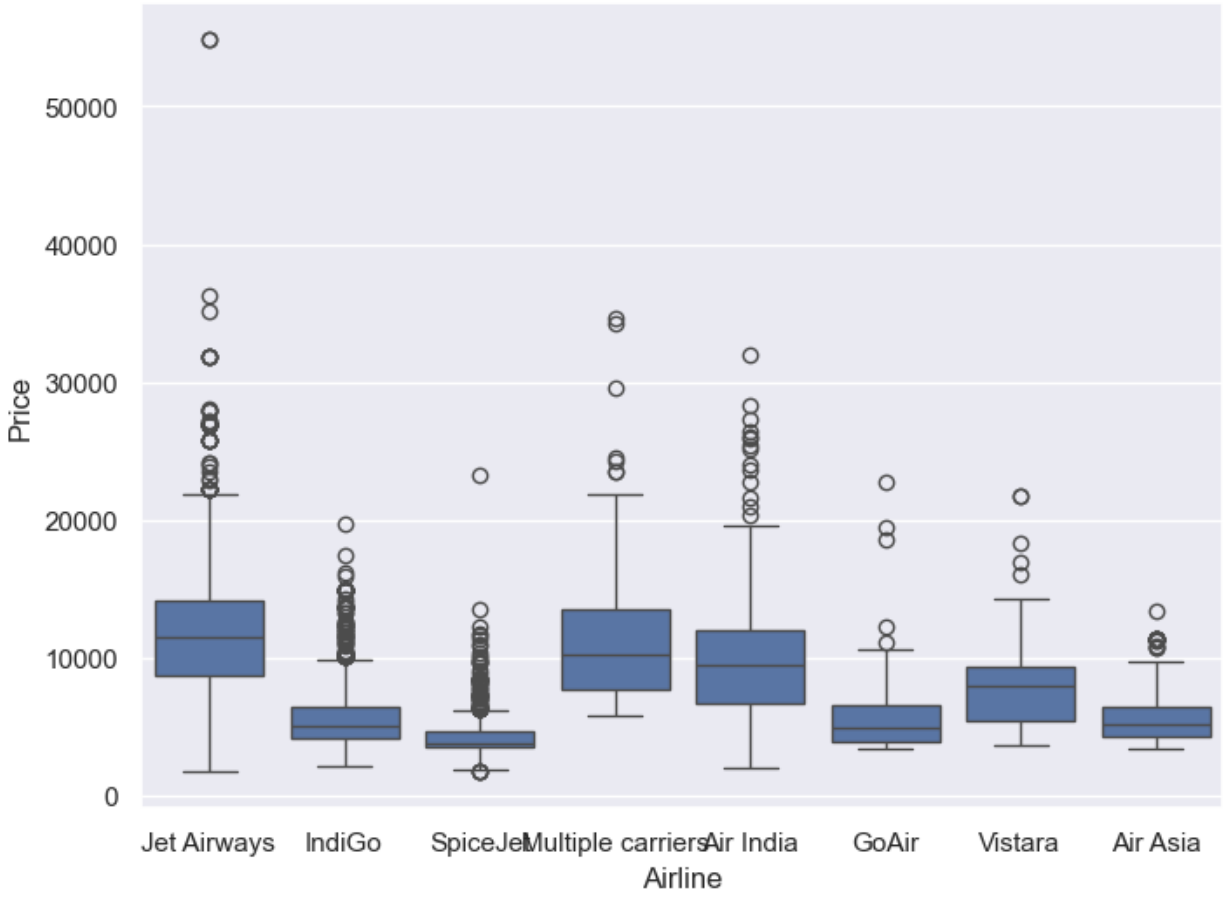
```
533.0
```

```
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
```

```
# 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())
```

```
                  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
```

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

```
Designation
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: count, dtype: int64
```

```
# 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())
```

```
Number of unique values in Airline column:  8
Number of unique values in Date_of_Journey column:  40
Number of unique values in Source column:  5
Number of unique values in Destination column:  6
Number of unique values in Route column:  122
Number of unique values in Dep_Time column:  218
Number of unique values in Arrival_Time column:  1220
Number of unique values in Duration column:  362
Number of unique values in Total_Stops column:  5
```

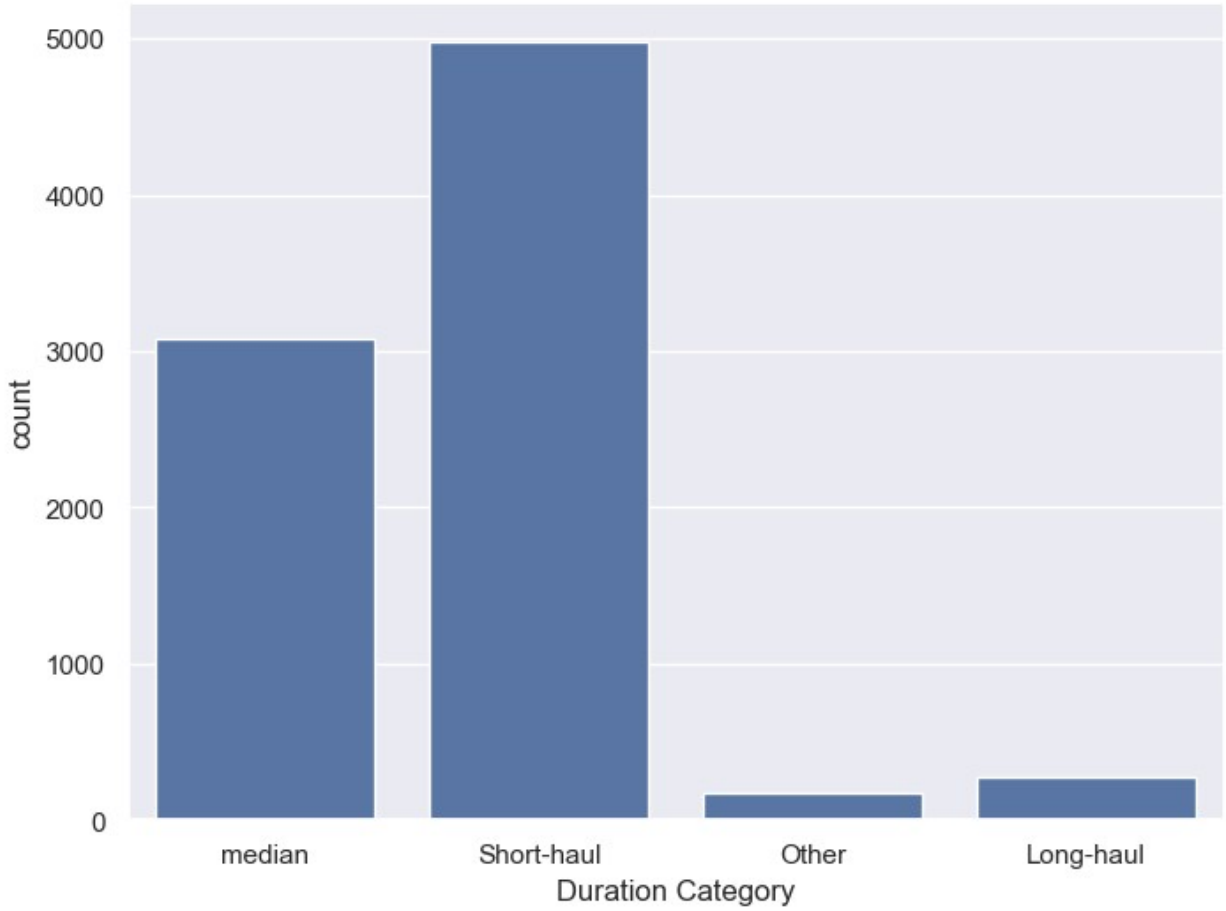
```
planes['Duration'].head()
```

```
0        19h
1     5h 25m
2     4h 45m
3     2h 25m
4    15h 30m
Name: Duration, dtype: object
```

```
flight_category = ["Short-haul","median","Long-haul"]

short_flight = "0h|1h|2h|3h|4h"
median_flight = "5h|6h|7h|8h|9h"
long_flight = "10h|11h|12h|13h|14h|15h|16h"

conditions = [
    (planes['Duration'].str.contains("0h|1h|2h|3h|4h")),
    (planes['Duration'].str.contains("5h|6h|7h|8h|9h")),
    (planes['Duration'].str.contains("10h|11h|12h|13h|14h|15h|16h"))

]

planes["Duration Category"] =
np.select(conditions,flight_category,default="Other")

print(planes[['Duration','Duration Category']].head(5))
sns.countplot(data=planes,x='Duration Category')
plt.show()
```

```
  Duration Duration Category
0      19h            median
1   5h 25m            median
2   4h 45m        Short-haul
```

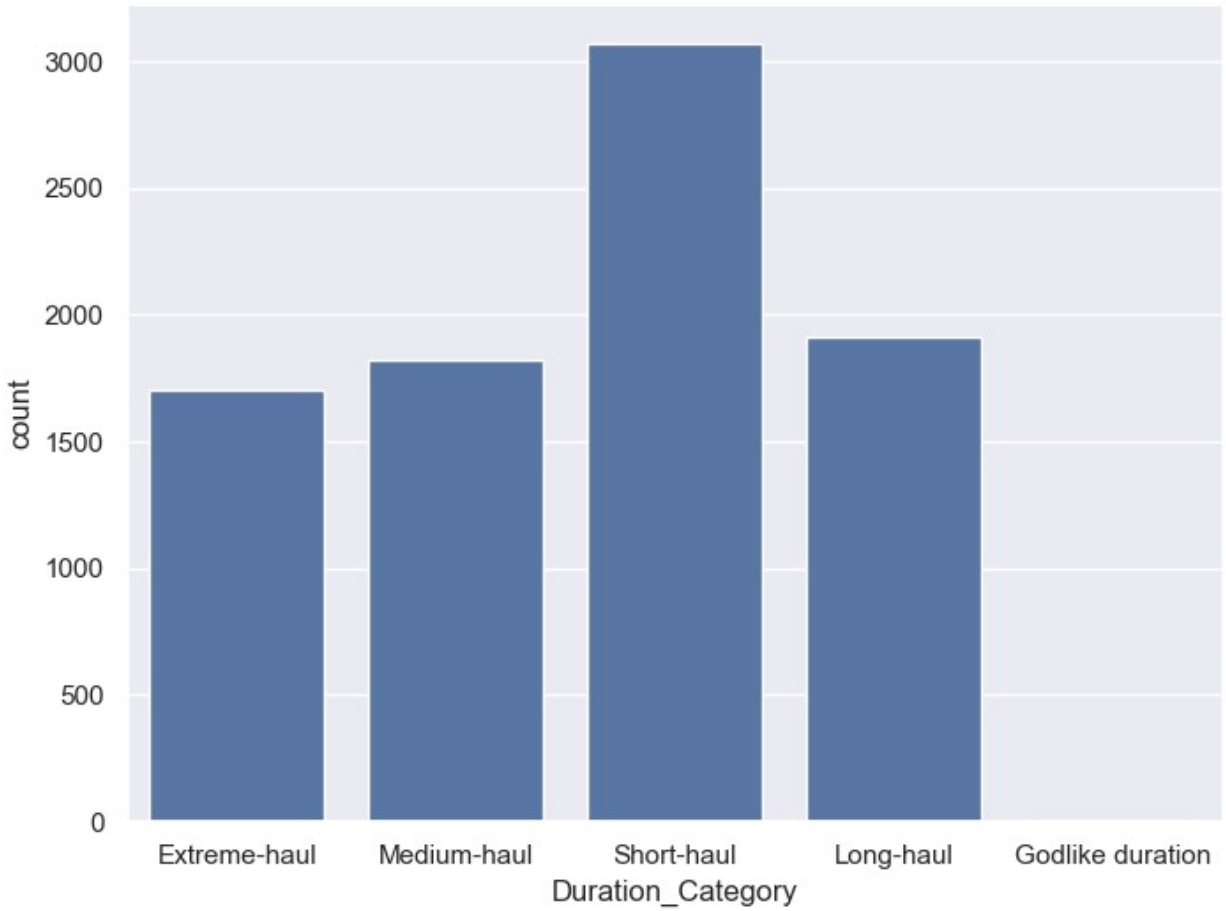
```
3   2h 25m        Short-haul
4  15h 30m            median
```




```
flight_categories = ['Extreme-haul',"Long-haul", "Medium-haul",
"Short-haul"]
# Create conditions for values in flight_categories to be created
conditions = [
    (planes["Duration"].str.contains("17h|18h|19h|20h|21h|22h|23h|24h|
25h|26h|27h|28h")),
    (planes["Duration"].str.contains("10h|11h|12h|13h|14h|15h|16h")),
    (planes["Duration"].str.contains("5h|6h|7h|8h|9h")),
    (planes["Duration"].str.contains('0h|1h|2h|3h|4h'))
]

# Apply the conditions list to the flight_categories
planes["Duration_Category"] = np.select(conditions,
                                        flight_categories,
default="Godlike duration")

# Plot the counts of each category
print(planes[['Duration','Duration_Category']].head(30))
```

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

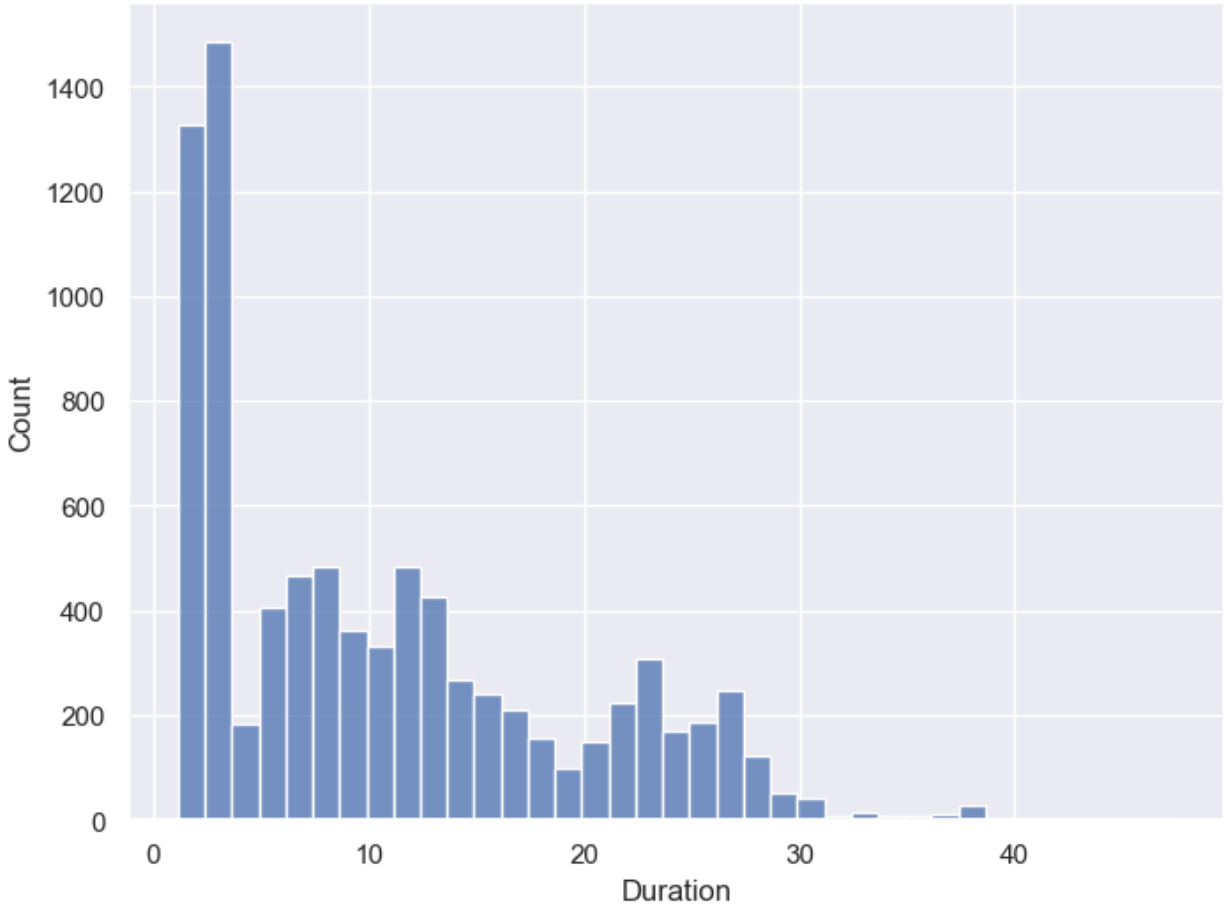
	Duration	Duration_Category
0	19h	Extreme-haul
1	5h 25m	Medium-haul
2	4h 45m	Short-haul
3	2h 25m	Short-haul
4	15h 30m	Long-haul
5	21h 5m	Extreme-haul
6	25h 30m	Extreme-haul
7	7h 50m	Medium-haul
8	13h 15m	Long-haul
9	2h 35m	Short-haul
10	2h 15m	Short-haul
11	12h 10m	Long-haul
12	2h 35m	Short-haul
13	26h 35m	Extreme-haul
14	4h 30m	Short-haul
15	22h 35m	Extreme-haul
16	23h	Extreme-haul
17	20h 35m	Extreme-haul
18	5h 10m	Medium-haul
19	15h 20m	Long-haul
20	2h 50m	Short-haul
21	2h 55m	Short-haul
22	13h 20m	Long-haul
23	15h 10m	Long-haul
24	5h 45m	Medium-haul
25	5h 55m	Medium-haul
26	2h 50m	Short-haul
27	2h 15m	Short-haul
28	2h 15m	Short-haul
29	13h 25m	Long-haul




```
planes['Duration'].head()
```

```
0        19h
1     5h 25m
2     4h 45m
3     2h 25m
4    15h 30m
Name: Duration, dtype: object
```

```
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)

sns.histplot(data=planes,x='Duration')
plt.show()
```




```
airline_median_duration = planes.groupby("Airline")
["Duration"].median()
planes["std_dev"] = planes.groupby("Airline")
["Duration"].transform(lambda x: x.std())
print(planes[["Airline","Duration"]].value_counts())
```

```
Airline      Duration
IndiGo       2.50        193
             2.35        141
Jet Airways  3.00        140
IndiGo       1.30        110
Jet Airways  1.30        105
                        ...
Air India    36.25         1
             11.25         1
             39.50         1
Jet Airways  9.25          1
GoAir        6.50          1
Name: count, Length: 743, dtype: int64
```

```
price_destination_mean = planes.groupby("Destination")["Price"].mean()
planes["std_dev"] = planes.groupby("Destination")
```

```
["Price"].transform(lambda x: x.std())
print(planes[["Destination","Price"]].value_counts())
```

```
Destination  Price
Cochin       10262.0    192
Banglore     10844.0    160
             4804.0     119
Delhi        7229.0     117
             4823.0      97
                        ...
Cochin       14667.0      1
             6197.0       1
             14762.0      1
             14817.0      1
             23533.0      1
Name: count, Length: 1723, dtype: int64
```

```
divorce = pd.read_csv("divorce.csv", parse_dates=["marriage_date"])
divorce.dtypes
```

```
divorce_date                 object
dob_man                      object
education_man                object
income_man                  float64
dob_woman                    object
education_woman              object
income_woman                float64
marriage_date        datetime64[ns]
marriage_duration           float64
num_kids                    float64
dtype: object
```

```
divorce["marriage_date"] = pd.to_datetime(divorce[["month", "day",
"year"]])
divorce.head(2)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call
last)
Cell In[117], line 1
----> 1 divorce["marriage_date"] = pd.to_datetime(divorce[["month",
"day", "year"]])
      2 divorce.head(2)

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/frame.py:4108, in
DataFrame.__getitem__(self, key)
   4106     if is_iterator(key):
```

```
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")
[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/indexes/base.py:6200, in
Index._get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer =
self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/
site-packages/pandas/core/indexes/base.py:6249, in
Index._raise_if_missing(self, key, indexer, axis_name)
   6247 if nmissing:
   6248     if nmissing == len(indexer):
-> 6249         raise KeyError(f"None of [{key}] are in the
[{axis_name}]")
   6251     not_found = list(ensure_index(key)[missing_mask.nonzero()
[0]].unique())
   6252     raise KeyError(f"{not_found} not in index")

KeyError: "None of [Index(['month', 'day', 'year'], dtype='object')]
are in the [columns]"
```

```
divorce["marriage_month"] = divorce["marriage_date"].dt.month
divorce.head()
```

```
  divorce_date     dob_man education_man  income_man   dob_woman  \
0   2006-09-06  1975-12-18     Secondary      2000.0  1983-08-01
1   2008-01-02  1976-11-17  Professional      6000.0  1977-03-13
2   2011-01-02  1969-04-06   Preparatory      5000.0  1970-02-16
3   2011-01-02  1979-11-13     Secondary     12000.0  1981-05-13
4   2011-01-02  1982-09-20  Professional      6000.0  1988-01-30

  education_woman  income_woman marriage_date  marriage_duration
num_kids  \
0       Secondary        1800.0    2000-06-26                5.0
1.0
1    Professional        6000.0    2001-09-02                7.0
NaN
2    Professional        5000.0    2000-02-02                2.0
```

```
2.0
3       Secondary        12000.0    2006-05-13                 2.0
NaN
4    Professional        10000.0    2007-08-06                 3.0
NaN

   marriage_year  marriage_month
0           2000               6
1           2001               9
2           2000               2
3           2006               5
4           2007               8
```

```
sns.lineplot(data=divorce, x = "marriage_month", y =
"marriage_duration")
plt.show()
```


```
divorce = pd.read_csv("divorce.csv", parse_dates=["divorce_date",
"dob_man", "dob_woman"])
divorce.dtypes
```

```
divorce_date         datetime64[ns]
dob_man              datetime64[ns]
education_man                object
income_man                  float64
dob_woman            datetime64[ns]
education_woman              object
income_woman                float64
marriage_date                object
marriage_duration           float64
num_kids                    float64
dtype: object
```

```
divorce["marriage_date"] = pd.to_datetime(divorce["marriage_date"])
divorce.dtypes
```

```
divorce_date         datetime64[ns]
dob_man              datetime64[ns]
education_man                object
income_man                  float64
dob_woman            datetime64[ns]
education_woman              object
income_woman                float64
marriage_date        datetime64[ns]
marriage_duration           float64
num_kids                    float64
dtype: object
```

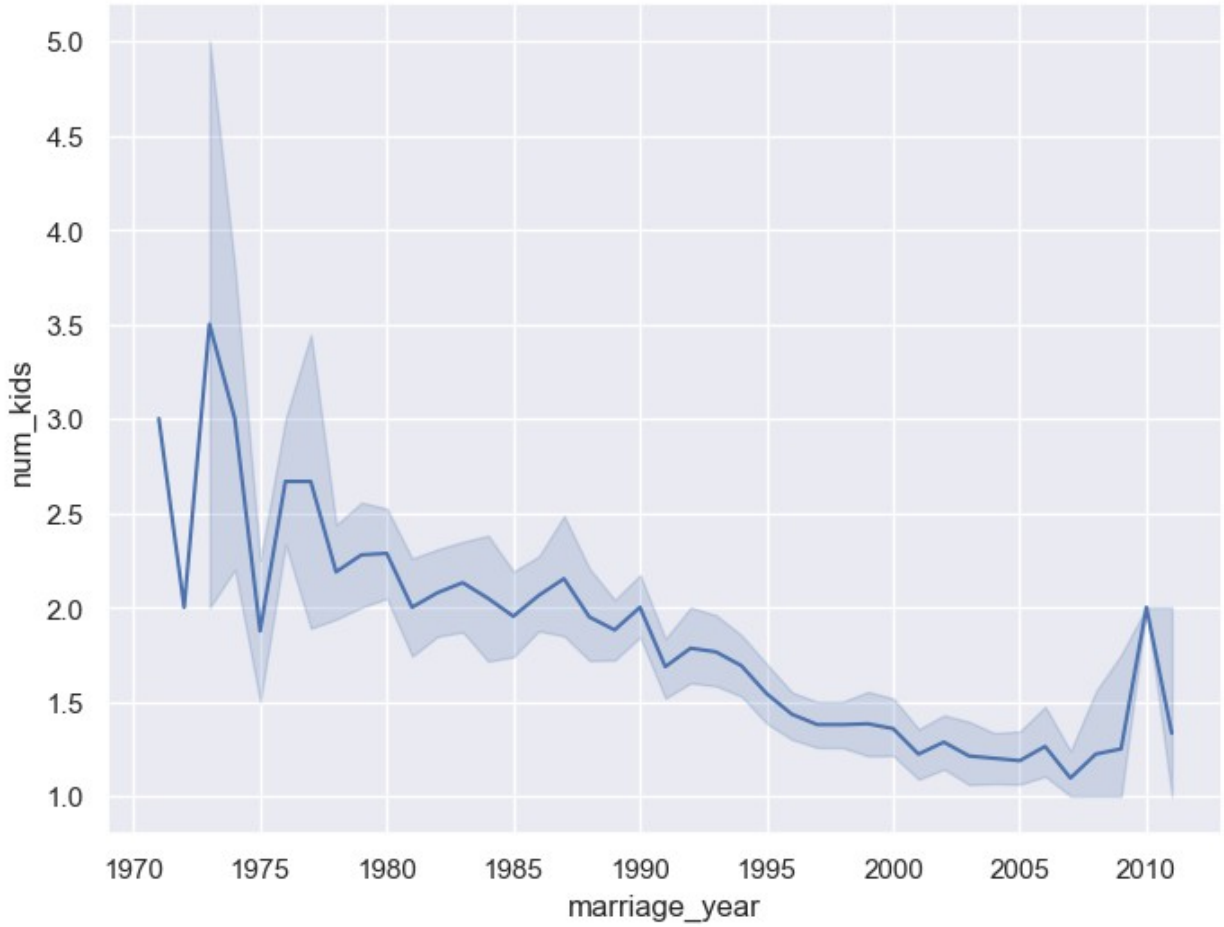
```
divorce["marriage_year"] = divorce["marriage_date"].dt.year
divorce.head()
```

```
  divorce_date    dob_man education_man  income_man  dob_woman  \
0   2006-09-06 1975-12-18     Secondary      2000.0 1983-08-01
1   2008-01-02 1976-11-17  Professional      6000.0 1977-03-13
2   2011-01-02 1969-04-06   Preparatory      5000.0 1970-02-16
3   2011-01-02 1979-11-13     Secondary     12000.0 1981-05-13
4   2011-01-02 1982-09-20  Professional      6000.0 1988-01-30

  education_woman  income_woman marriage_date  marriage_duration
num_kids  \
0       Secondary        1800.0    2000-06-26                5.0
1.0
1    Professional        6000.0    2001-09-02                7.0
NaN
2    Professional        5000.0    2000-02-02                2.0
2.0
3       Secondary       12000.0    2006-05-13                2.0
NaN
4    Professional       10000.0    2007-08-06                3.0
NaN
```

```
   marriage_year
0           2000
1           2001
2           2000
3           2006
4           2007
```

```
sns.lineplot(data=divorce, x = "marriage_year", y = "num_kids")
```

```
<Axes: xlabel='marriage_year', ylabel='num_kids'>
```



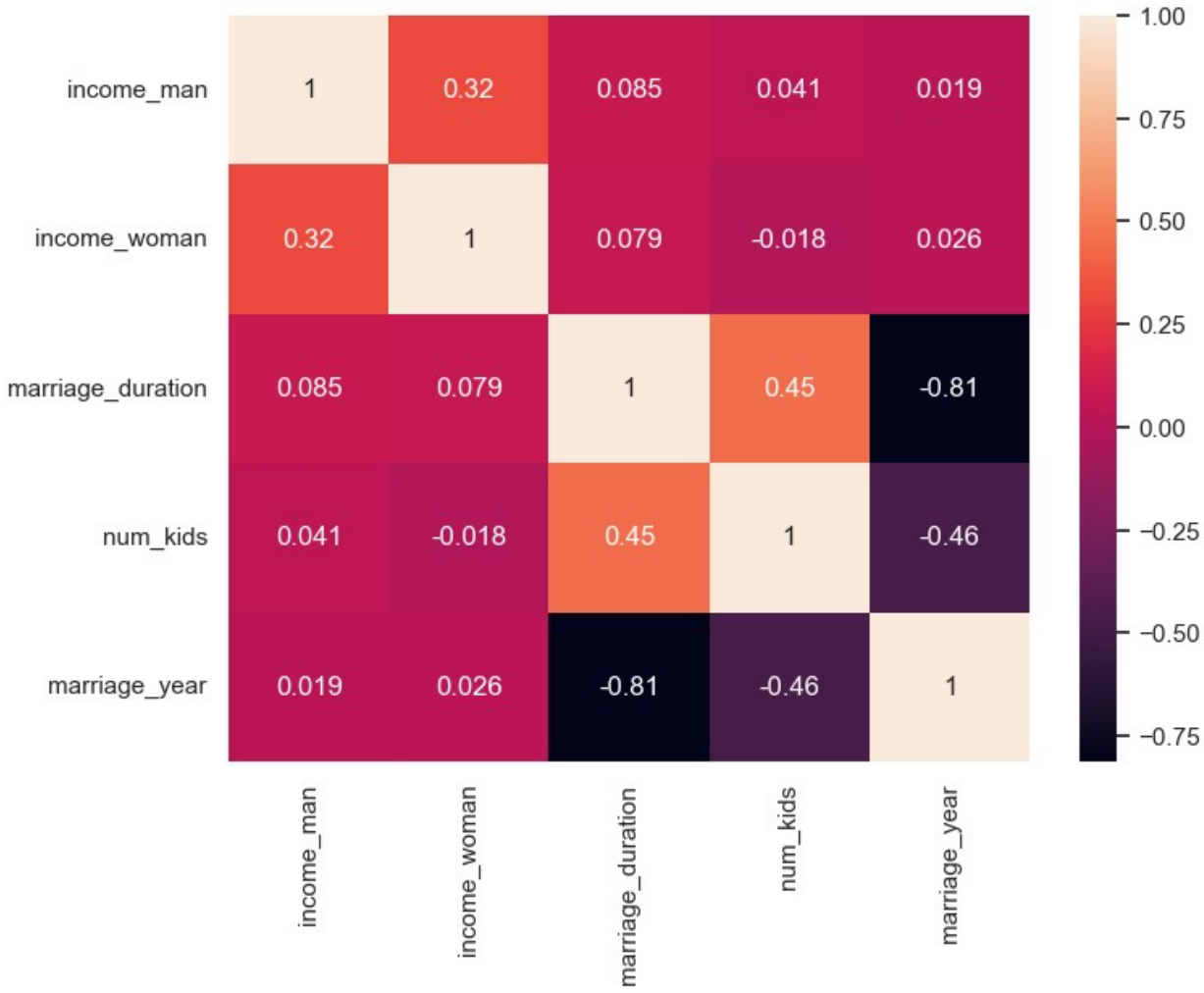

```
divorce.corr(numeric_only=True)
```

```
                  income_man  income_woman  marriage_duration
num_kids  \
income_man          1.000000      0.318047           0.085321
0.040848
income_woman        0.318047      1.000000           0.078677 -
0.018015
marriage_duration   0.085321      0.078677           1.000000
0.447358
```

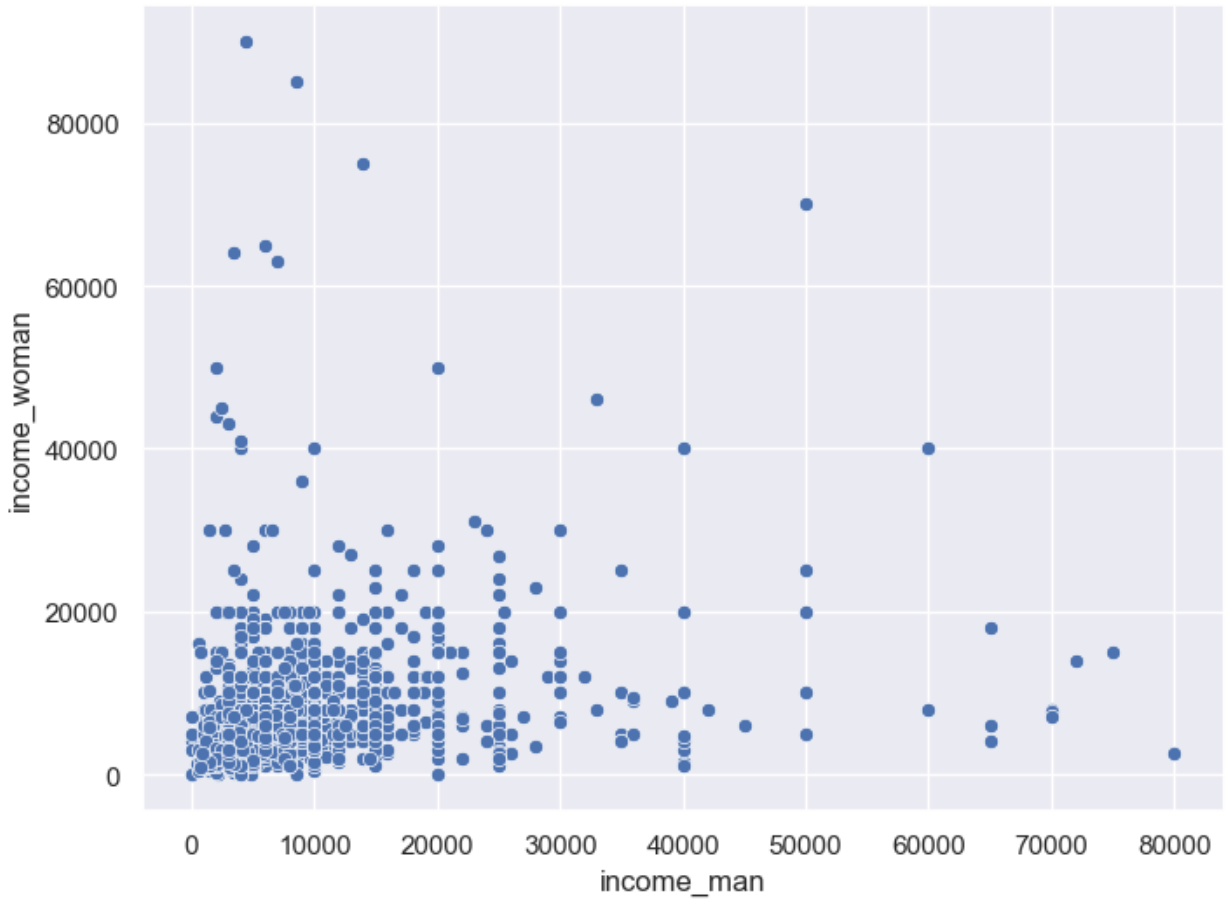
```
num_kids              0.040848     -0.018015            0.447358
1.000000
marriage_year         0.019170      0.026433           -0.812469 -
0.461495
```

```
                    marriage_year
income_man               0.019170
income_woman             0.026433
marriage_duration       -0.812469
num_kids                -0.461495
marriage_year            1.000000
```

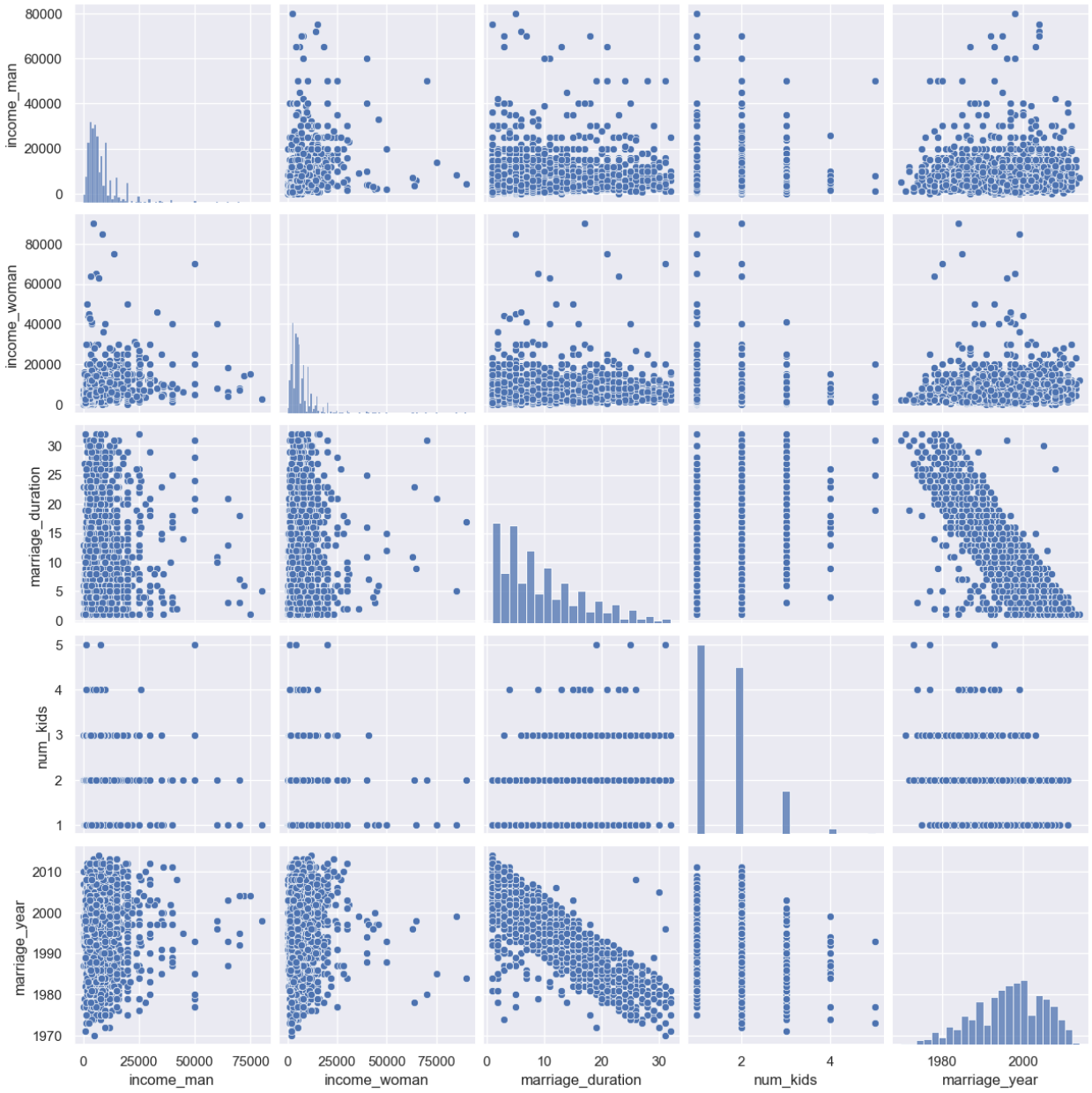
```
sns.heatmap(divorce.corr(numeric_only=True), annot=True)
plt.show()
```



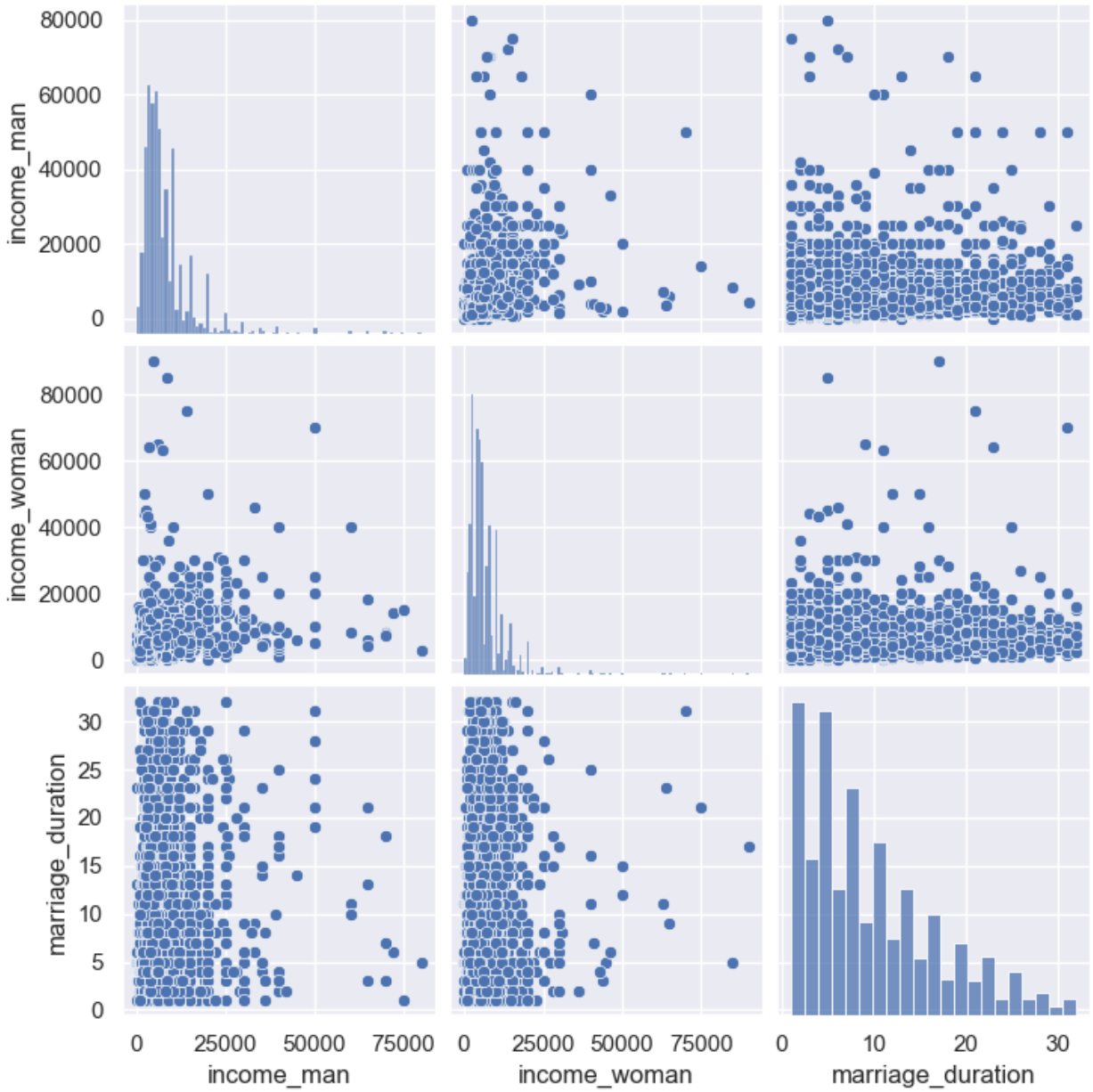

```
sns.scatterplot(data=divorce, x = "income_man", y = "income_woman")
plt.show()
```



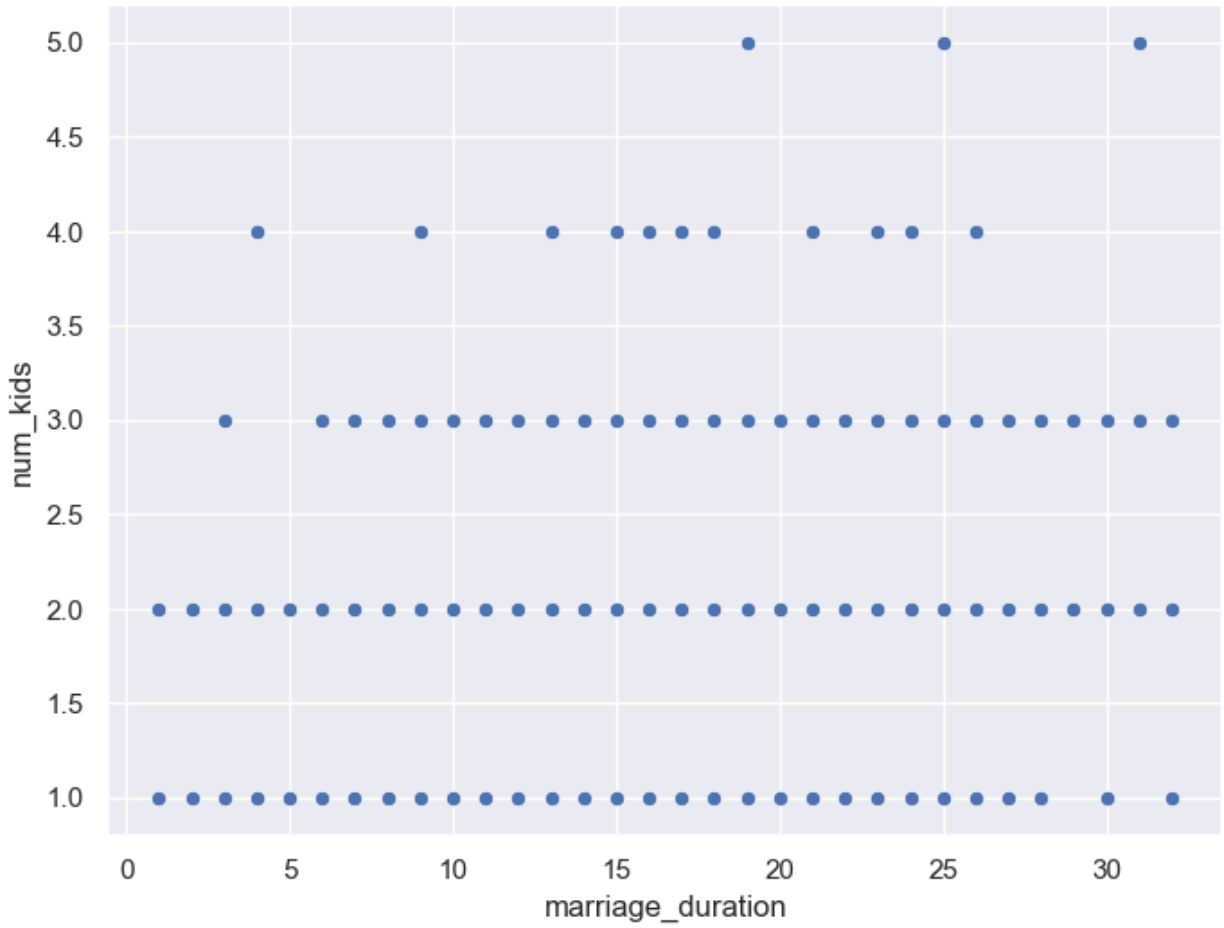

```
sns.pairplot(data = divorce)
plt.show()
```



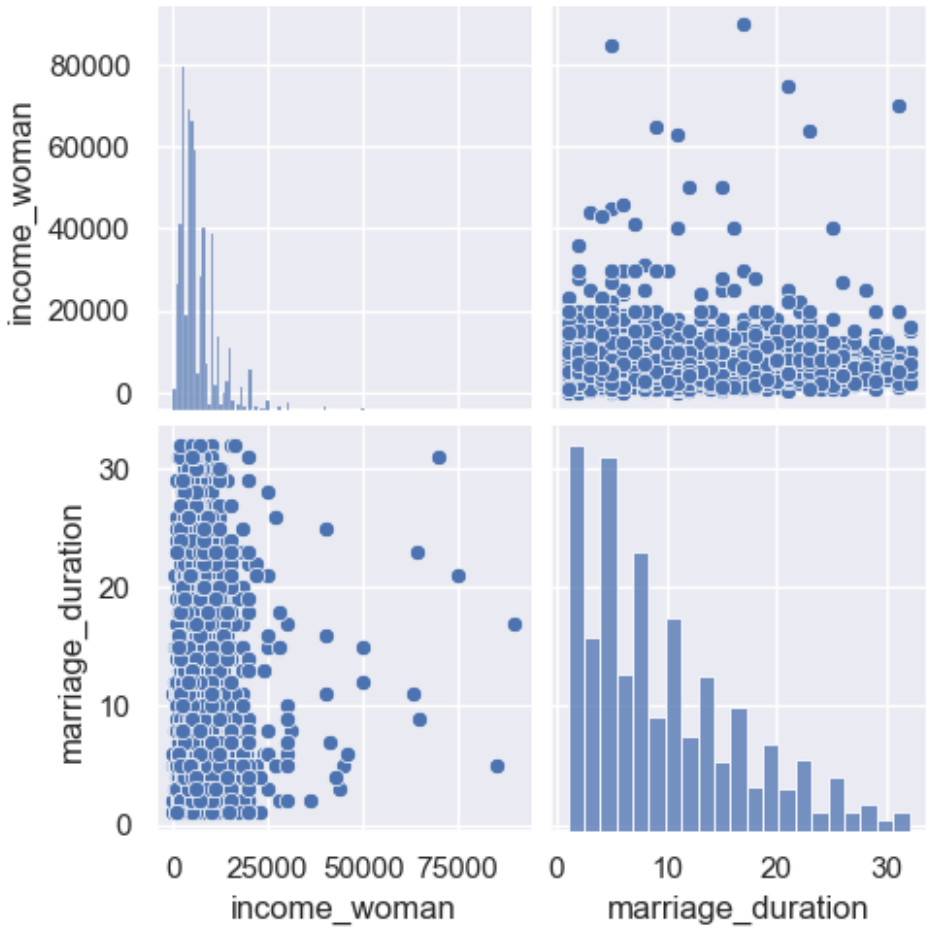

```
sns.pairplot(data=divorce, vars = ["income_man","income_woman",
"marriage_duration"])
plt.show()
```




```
sns.scatterplot(data=divorce,x="marriage_duration",y="num_kids")
plt.show()
```



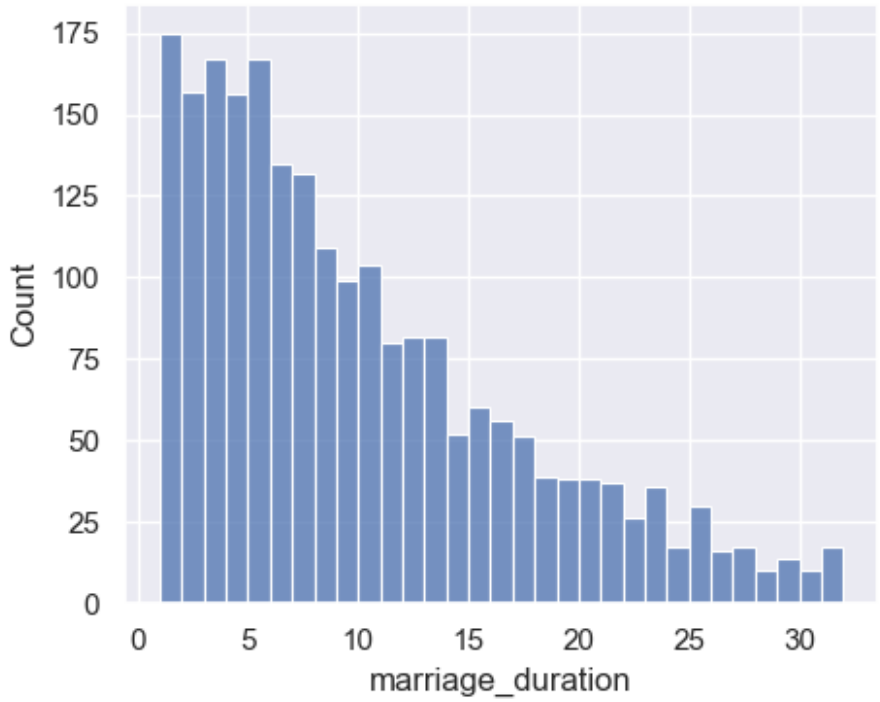

```
sns.pairplot(data=divorce,vars=["income_woman", "marriage_duration"])
plt.show()
```



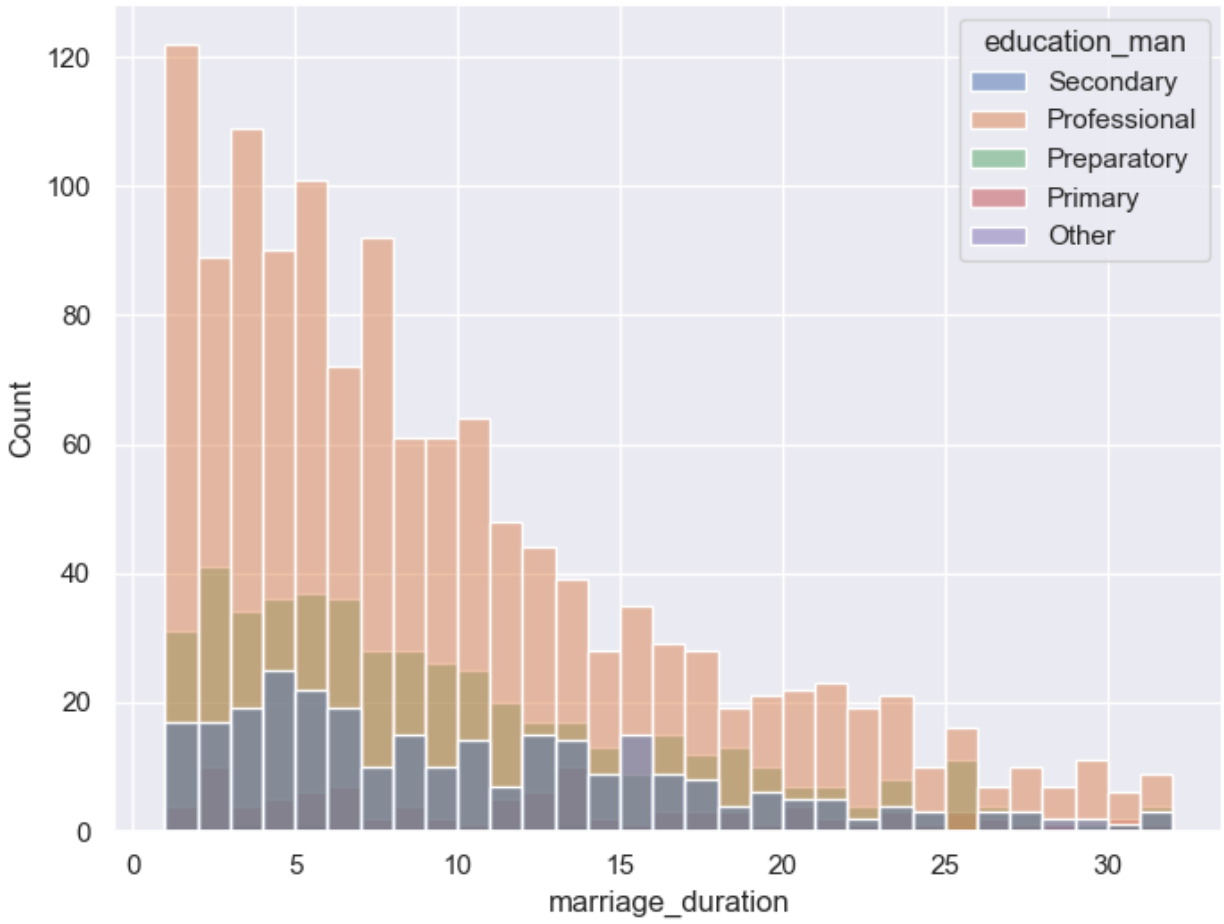

```
divorce["education_man"].value_counts()
```

```
education_man
Professional    1313
Preparatory      501
Secondary        288
Primary          100
Other              3
Name: count, dtype: int64
```

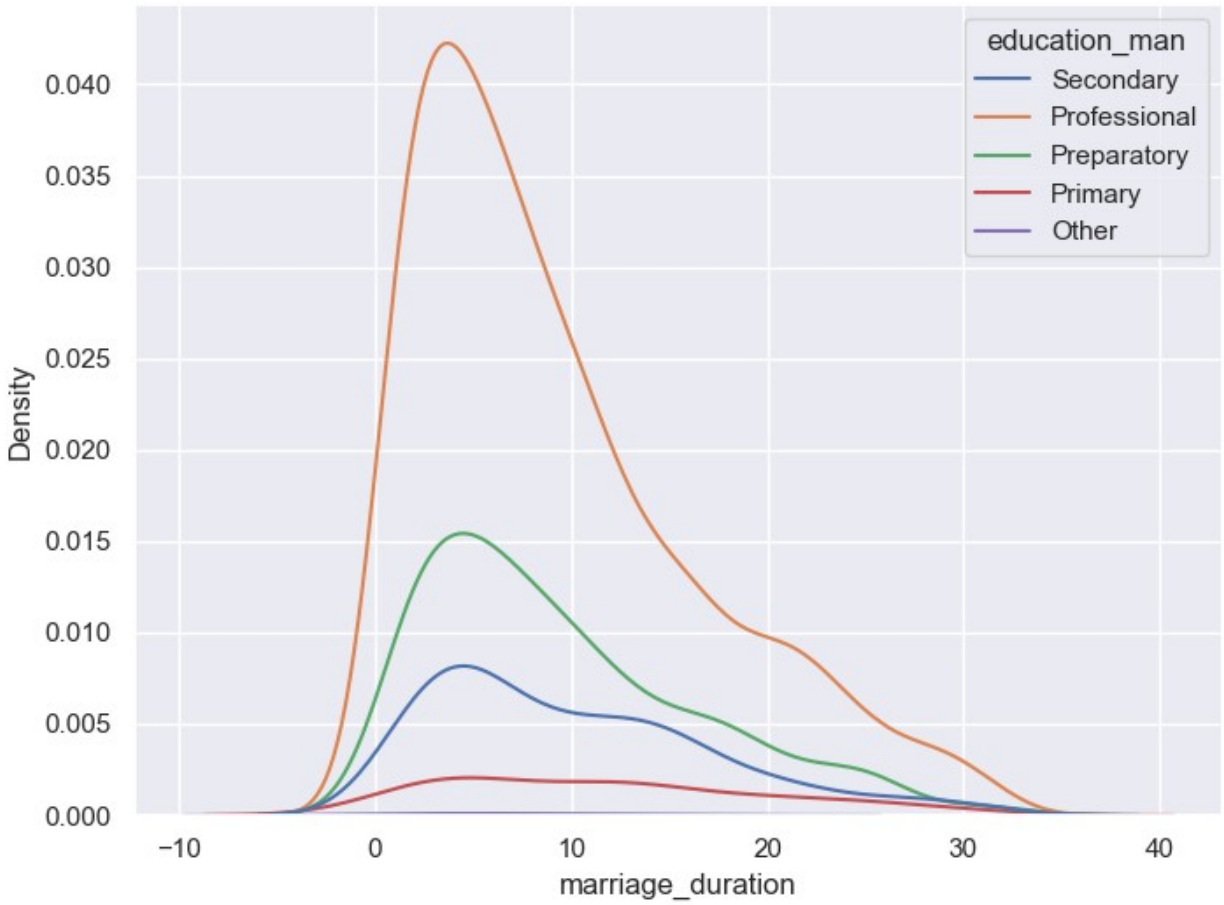
```
plt.figure(figsize=(5,4))
sns.histplot(data=divorce, x="marriage_duration",binwidth=1)
plt.show()
```



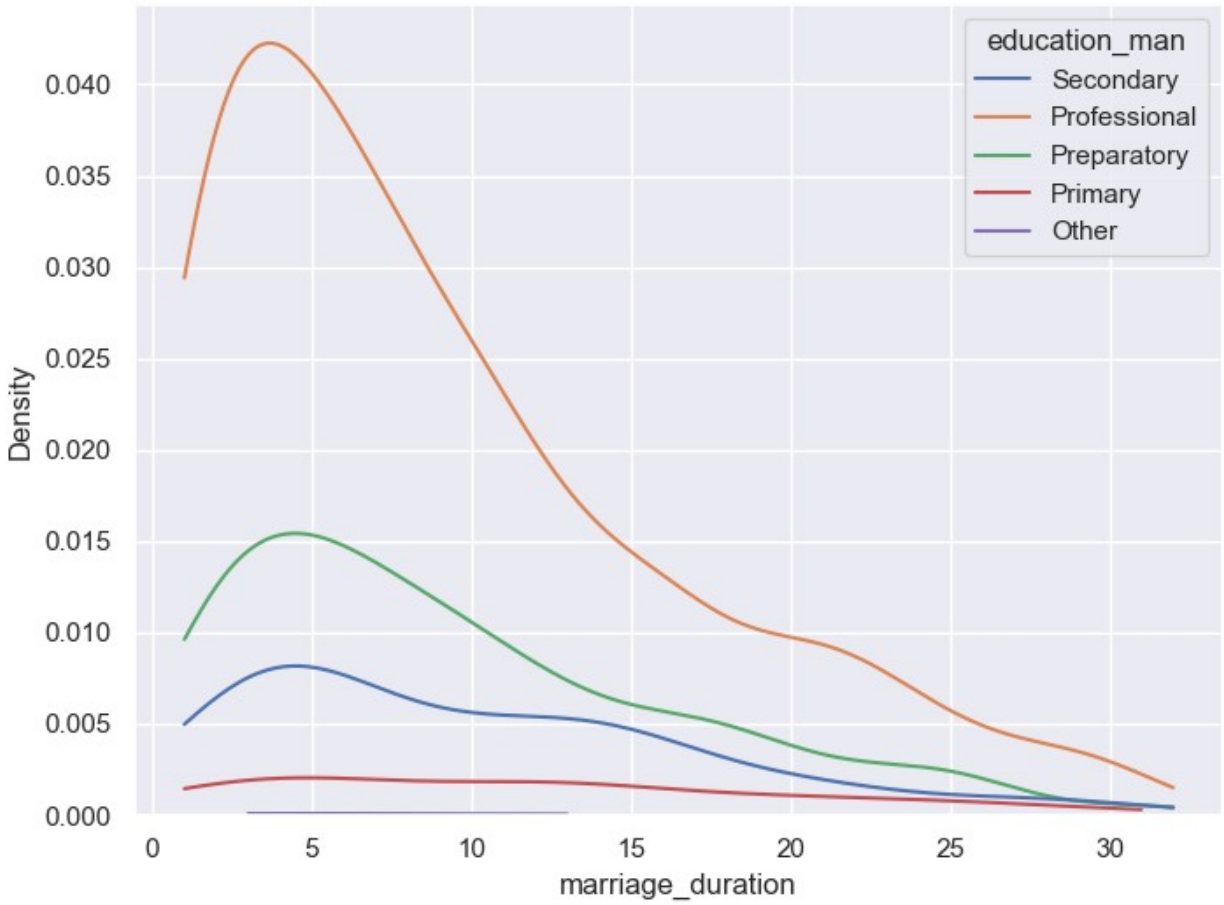

```
sns.histplot(data = divorce, x ="marriage_duration", hue =
"education_man", binwidth=1)
plt.show()
```



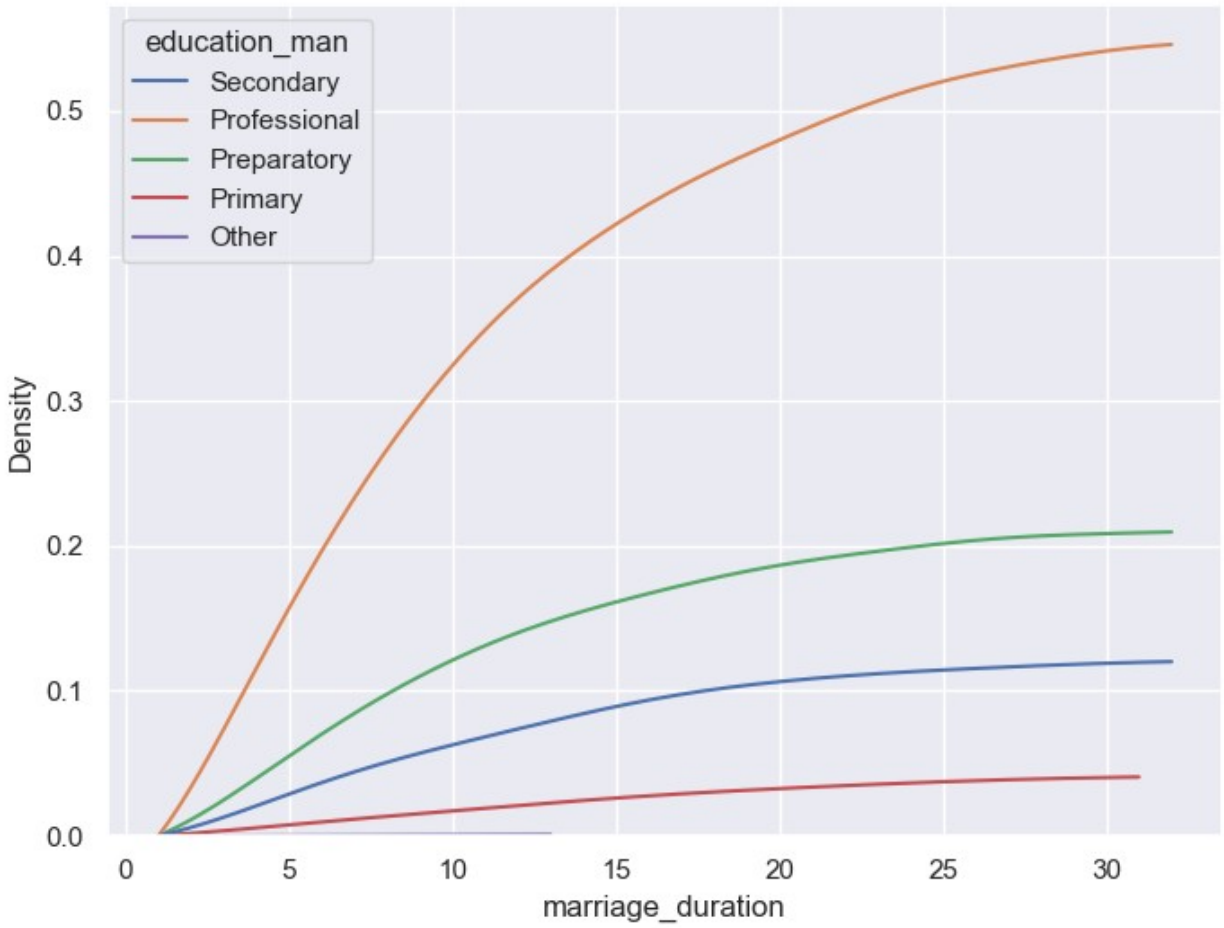

```
sns.kdeplot(data=divorce,x="marriage_duration",hue="education_man")
plt.show()
```



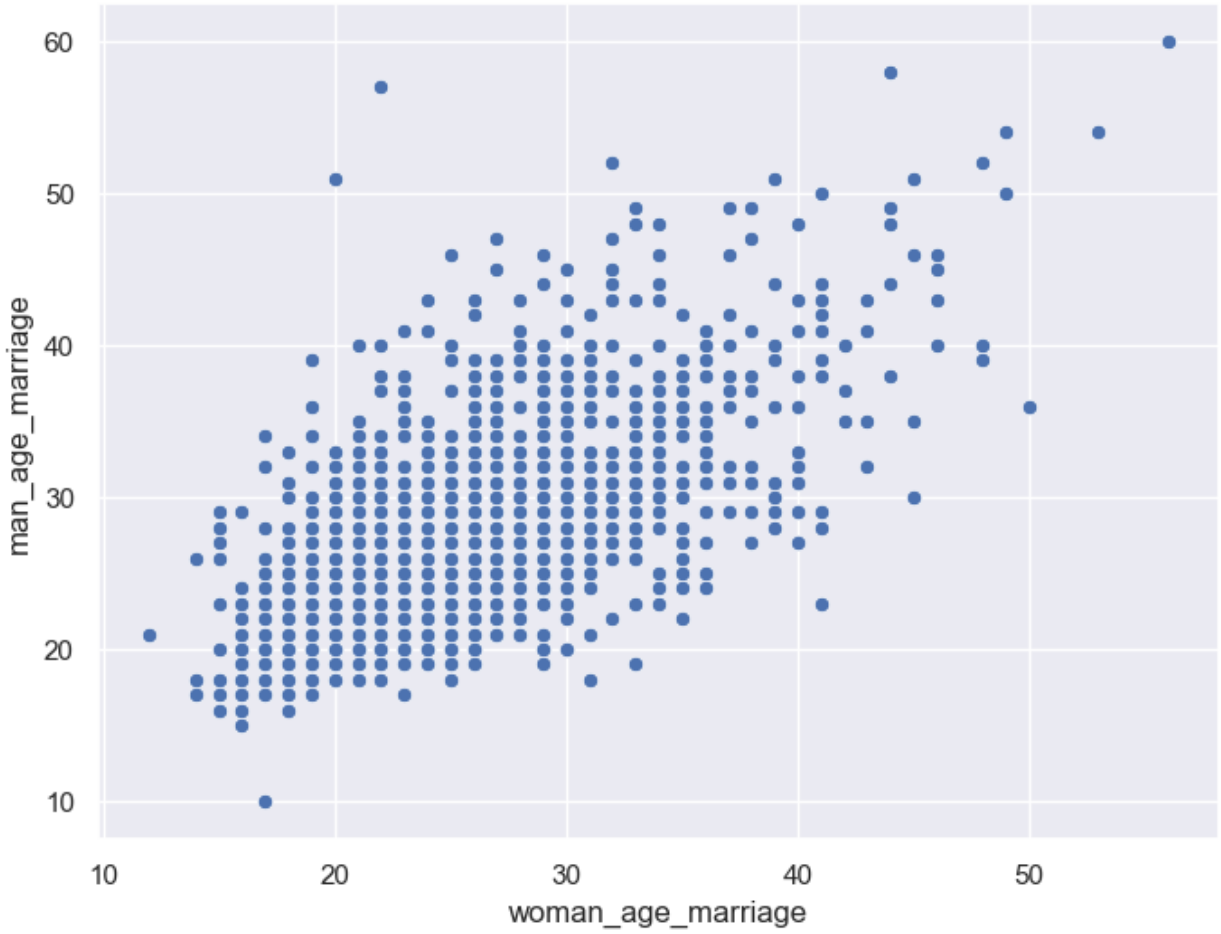

```
sns.kdeplot(data=divorce,x="marriage_duration",hue="education_man",cut
=0)
plt.show()
```



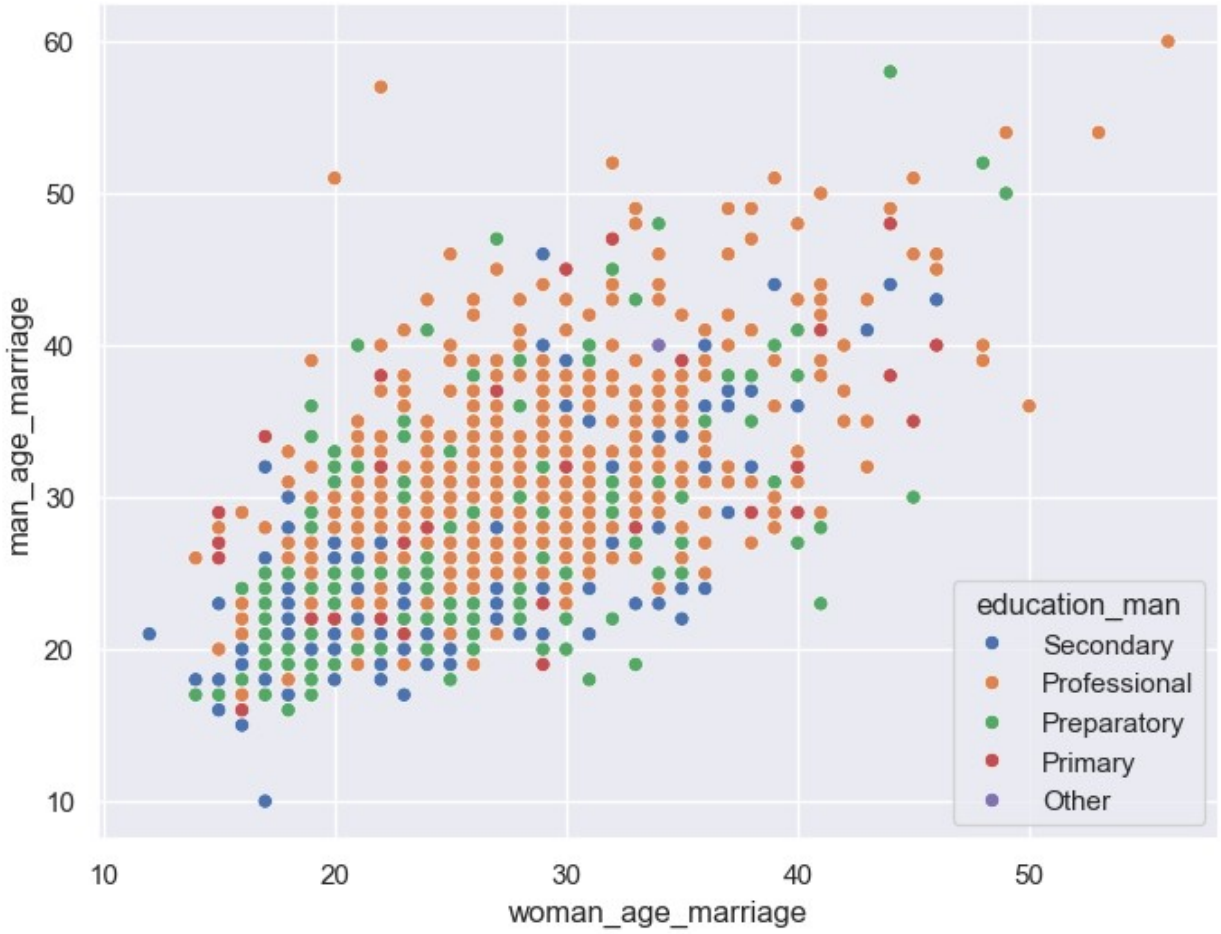

```
sns.kdeplot(data=divorce,x="marriage_duration",hue="education_man",cut
=0,cumulative=True)
plt.show()
```



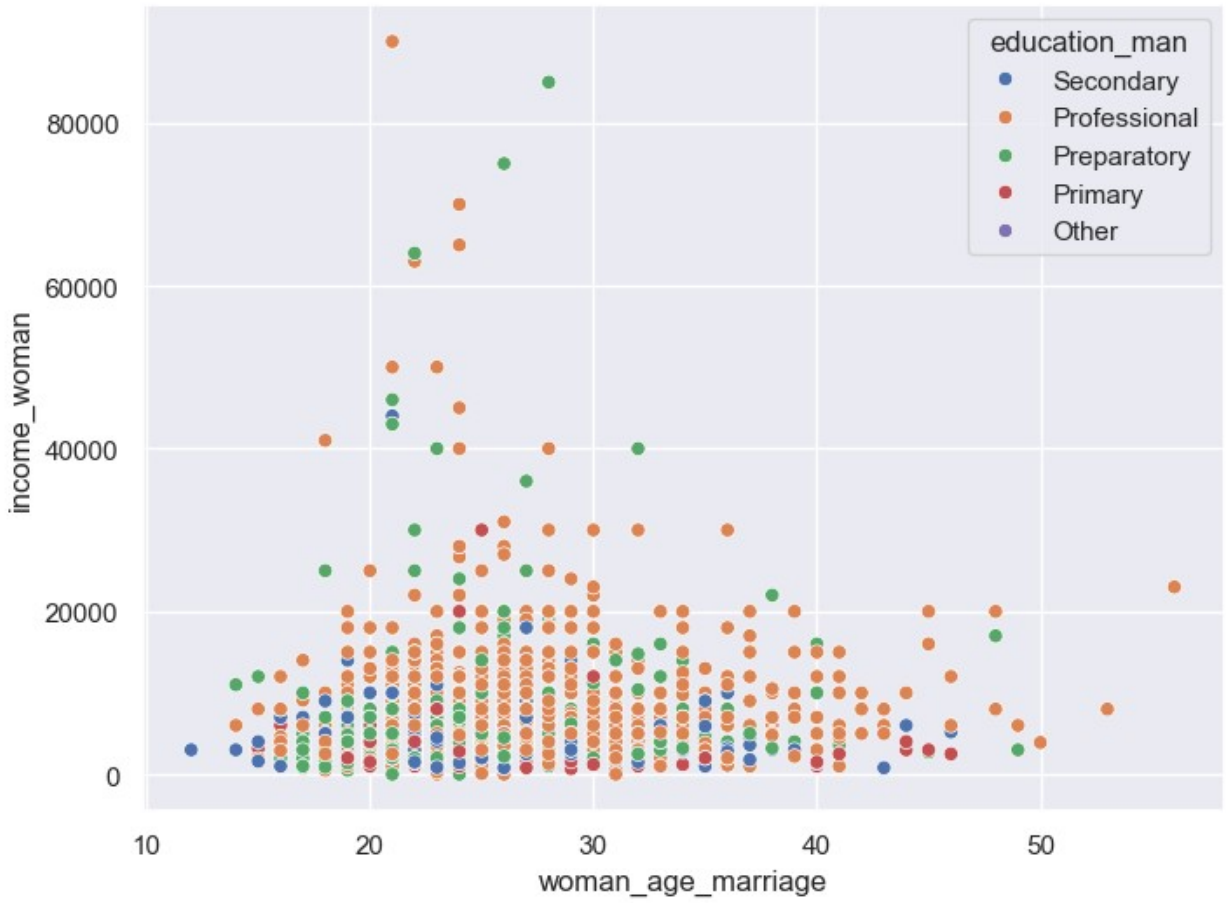

```
divorce["man_age_marriage"]= divorce["marriage_year"] -
divorce["dob_man"].dt.year
divorce["woman_age_marriage"]= divorce["marriage_year"] -
divorce["dob_woman"].dt.year
sns.scatterplot(data=divorce,x="woman_age_marriage",y="man_age_marriag
e")
plt.show()
```



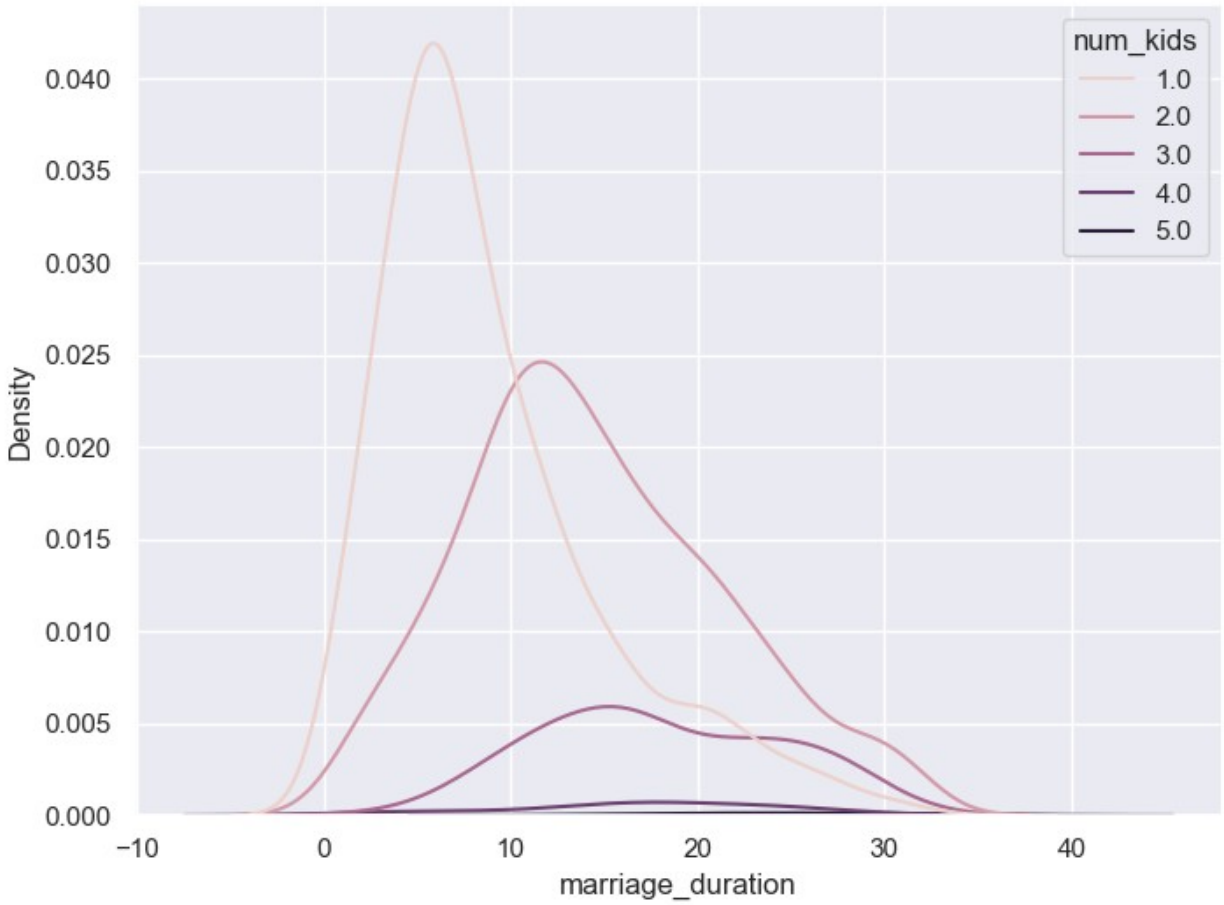

```
sns.scatterplot(data=divorce,x="woman_age_marriage",y="man_age_marriag
e",hue="education_man")
plt.show()
```



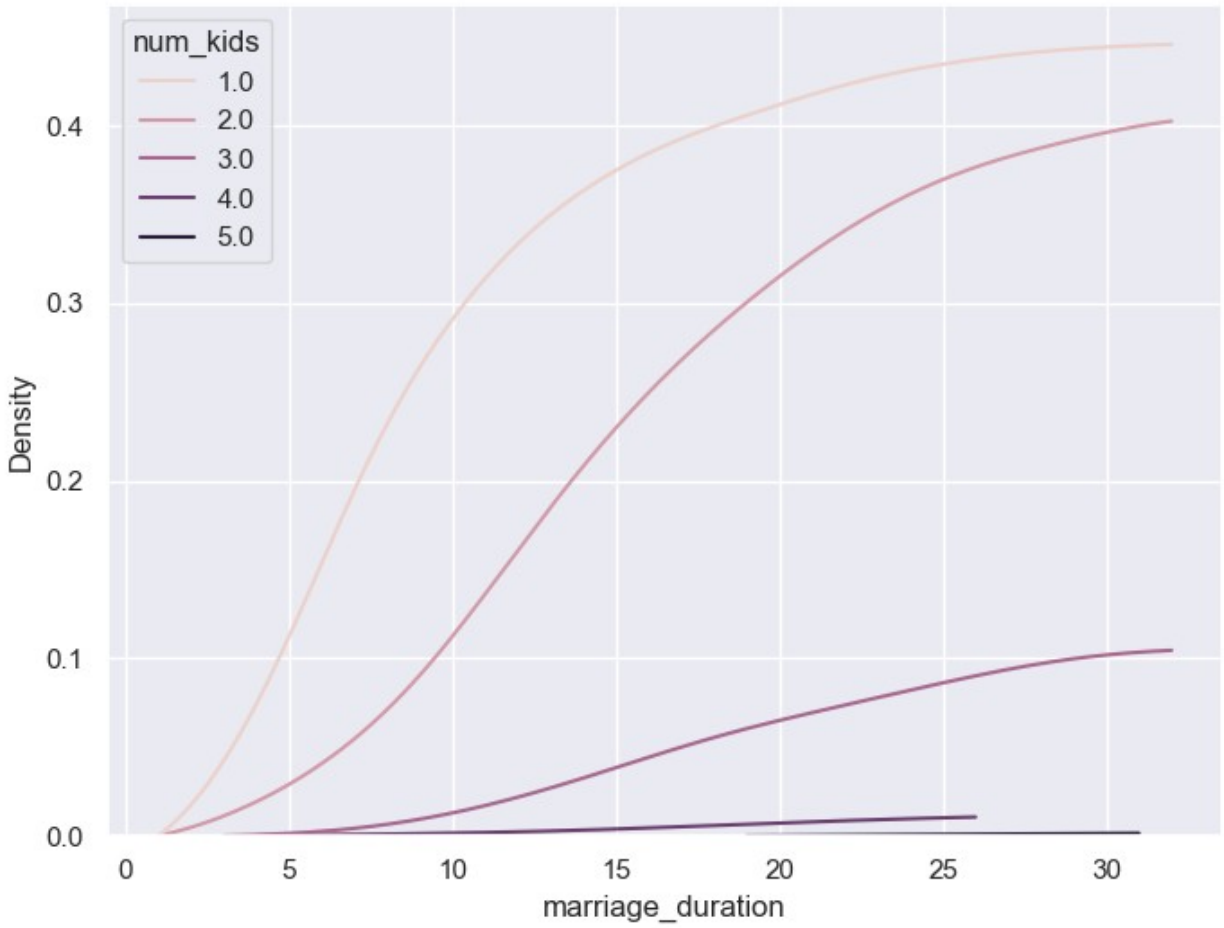

```
sns.scatterplot(data=divorce,x="woman_age_marriage",y="income_woman",h
ue="education_man")
plt.show()
```




```
sns.kdeplot(data=divorce,x="marriage_duration",hue="num_kids")
plt.show()
```




```
sns.kdeplot(data=divorce,x="marriage_duration",hue="num_kids",cut=0,cu
mulative=True)
plt.show()
```




```
planes = pd.read_csv("Airlines_unclean.csv")
print(planes["Destination"].value_counts())
```

```
Destination
Cochin       4391
Banglore     2773
Delhi        1219
New Delhi     888
Hyderabad     673
Kolkata       369
Name: count, dtype: int64
```

```
pd.crosstab(planes["Source"], planes["Destination"])
```

Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Source						
Banglore	0	0	1199	0	0	868
Chennai	0	0	0	0	364	0
Delhi	0	4318	0	0	0	0
Kolkata	2720	0	0	0	0	0
Mumbai	0	0	0	662	0	0

```
planes.groupby(["Source","Destination"])
["Price"].median().reset_index()
```

	Source	Destination	Price
0	Banglore	Delhi	4823.0
1	Banglore	New Delhi	10976.5
2	Chennai	Kolkata	3850.0
3	Delhi	Cochin	10262.0
4	Kolkata	Banglore	9345.0
5	Mumbai	Hyderabad	3342.0

```
pd.crosstab(planes["Source"],planes["Destination"],values=planes["Pric
e"],aggfunc="median")
```

Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Source						
Banglore	NaN	NaN	4823.0	NaN	NaN	10976.5
Chennai	NaN	NaN	NaN	NaN	3850.0	NaN
Delhi	NaN	10262.0	NaN	NaN	NaN	NaN
Kolkata	9345.0	NaN	NaN	NaN	NaN	NaN
Mumbai	NaN	NaN	NaN	3342.0	NaN	NaN

```
salaries = pd.read_csv("Salary_Rupee_USD.csv",index_col=0)
salaries["Job_Category"].value_counts(normalize=True)
```

```
Job_Category
Data Science         0.277641
Data Engineering     0.272727
Data Analytics       0.226044
Machine Learning     0.120393
Other                0.068796
Managerial           0.034398
Name: proportion, dtype: float64
```

```
pd.crosstab(salaries["Company_Size"],salaries["Experience"])
```

Experience	EN	EX	MI	SE
Company_Size				
L	24	7	49	44
M	25	9	58	136
S	18	1	21	15

```
pd.crosstab(salaries["Job_Category"],salaries["Company_Size"])
```

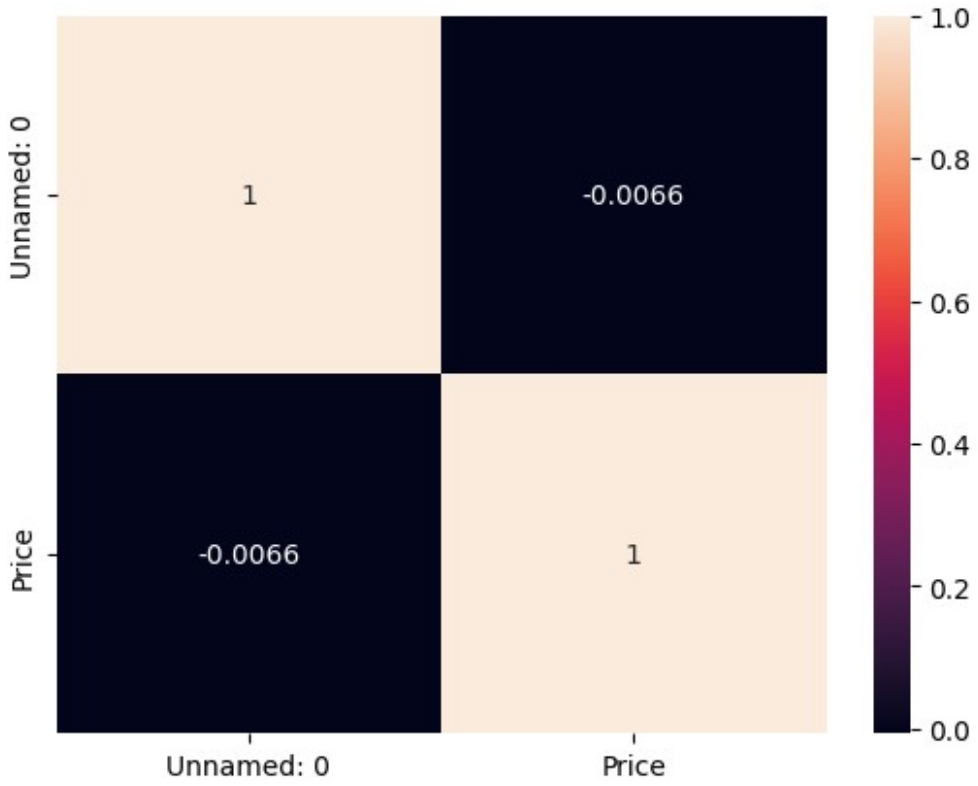
Company_Size	L	M	S
Job_Category			
Data Analytics	23	61	8
Data Engineering	28	72	11
Data Science	38	59	16
Machine Learning	17	19	13

```
Managerial          5   8   1
Other              13   9   6
```

```
pd.crosstab(salaries["Job_Category"],salaries["Company_Size"],values=s
alaries["Salary_USD"],aggfunc="mean")
```

```
Company_Size                    L              M             S
Job_Category
Data Analytics     112851.749217   95912.685246  53741.877000
Data Engineering   118939.035000  121287.060500  86927.136000
Data Science        96489.520105  116044.455864  62241.749250
Machine Learning   140779.491529  100794.236842  78812.586462
Managerial         190551.448800  150713.628000  31484.700000
Other               92873.911385   89750.578667  69871.248000
```

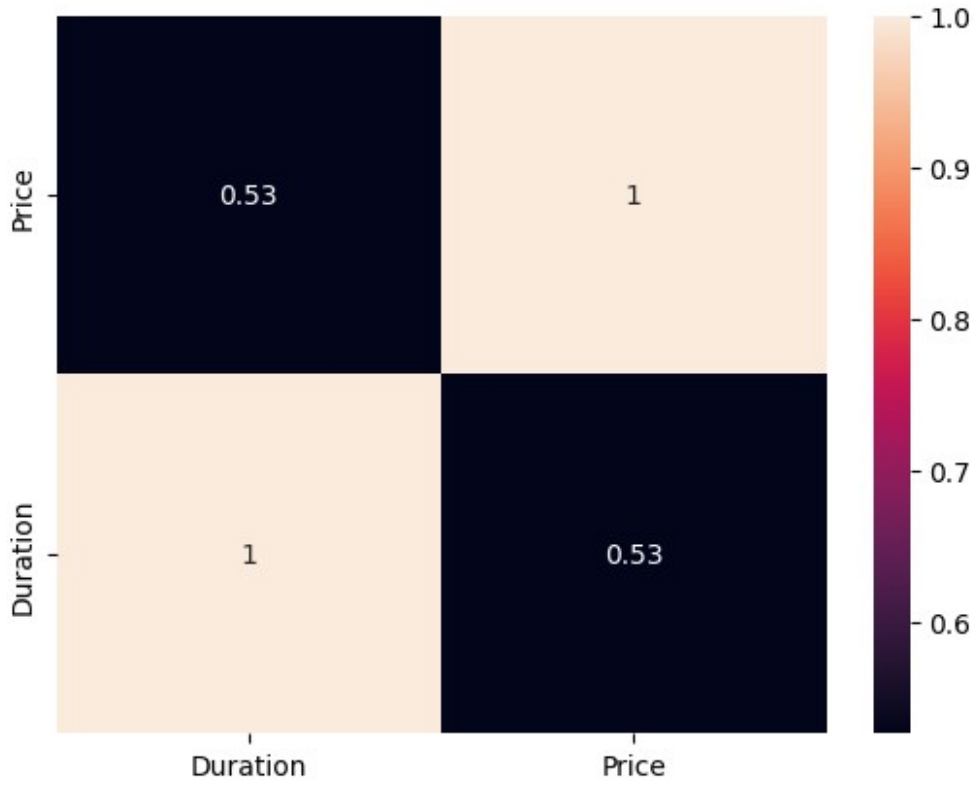
```
sns.heatmap(planes.corr(numeric_only=True),annot=True)
plt.show()
```




```
planes = pd.read_csv('Airlines_unclean.csv', index_col = 0,
parse_dates=['Date_of_Journey','Dep_Time','Arrival_Time'], date_format
= "%d/%m/%Y" )
# Remove the string character
planes["Duration"] = planes["Duration"].str.replace("h", ".")
planes["Duration"] = planes["Duration"].str.replace("m", "")
planes["Duration"] = planes["Duration"].str.replace(" ", "")
```

```
# Convert to float data type
planes["Duration"] = planes["Duration"].astype(float)
print(planes.info())
ax = sns.heatmap(planes.corr(numeric_only=True), annot=True)
ax.set_ylim([0,2])
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10660 entries, 0 to 10659
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10233 non-null  object
 1   Date_of_Journey  10338 non-null  datetime64[ns]
 2   Source           10473 non-null  object
 3   Destination      10313 non-null  object
 4   Route            10404 non-null  object
 5   Dep_Time         10400 non-null  object
 6   Arrival_Time     10466 non-null  object
 7   Duration         10446 non-null  float64
 8   Total_Stops      10448 non-null  object
 9   Additional_Info  10071 non-null  object
 10  Price            10044 non-null  float64
dtypes: datetime64[ns](1), float64(2), object(8)
memory usage: 999.4+ KB
None
```




```
planes.head()
```

```
       Airline Date_of_Journey    Source Destination
Route  \
0  Jet Airways      2019-06-09     Delhi      Cochin  DEL → LKO → BOM
→ COK
1       IndiGo      2019-05-12   Kolkata    Banglore        CCU → NAG
→ BLR
2       IndiGo      2019-03-01  Banglore   New Delhi        BLR → NAG
→ DEL
3     SpiceJet      2019-06-24   Kolkata    Banglore              CCU
→ BLR
4  Jet Airways      2019-03-12  Banglore   New Delhi        BLR → BOM
→ DEL

  Dep_Time     Arrival_Time  Duration Total_Stops  \
0     9:25   10/6/2023 4:25     19.00     2 stops
1    18:05            23:30      5.25      1 stop
2    16:50            21:35      4.45      1 stop
3     9:00            11:25      2.25    non-stop
4    18:55  13/3/2023 10:25     15.30      1 stop

               Additional_Info    Price
0                      No info  13882.0
1                      No info   6218.0
```

```
2                      No info  13302.0
3                      No info   3873.0
4  In-flight meal not included  11087.0
```

```
#remove Nan values
threshold = len(planes) * 0.05
print(threshold)
# Count the number of missing values in each column
print(planes.isna().sum())

# Find the five percent threshold
threshold = len(planes) * 0.05

# Create a filter
cols_to_drop = planes.columns[planes.isna().sum() <= threshold]

# Drop missing values for columns below the threshold
planes.dropna(subset=cols_to_drop, inplace=True)

print(planes.isna().sum())

#planes = planes.drop(columns = ['Additional_Info'])

# Calculate median plane ticket prices by Airline
airline_prices = planes.groupby("Airline")["Price"].median()
print(airline_prices)
print('==========================================')

# Convert to a dictionary
prices_dict = airline_prices.to_dict()
print(prices_dict)
print('==========================================')

# Map the dictionary to missing values of Price by Airline
planes["Price"] =
planes["Price"].fillna(planes["Airline"].map(prices_dict))

# Check for missing values
print(planes.isna().sum())
```

```
533.0
Airline            427
Date_of_Journey    322
Source             187
Destination        347
Route              256
Dep_Time           260
Arrival_Time       194
Duration           214
Total_Stops        212
```

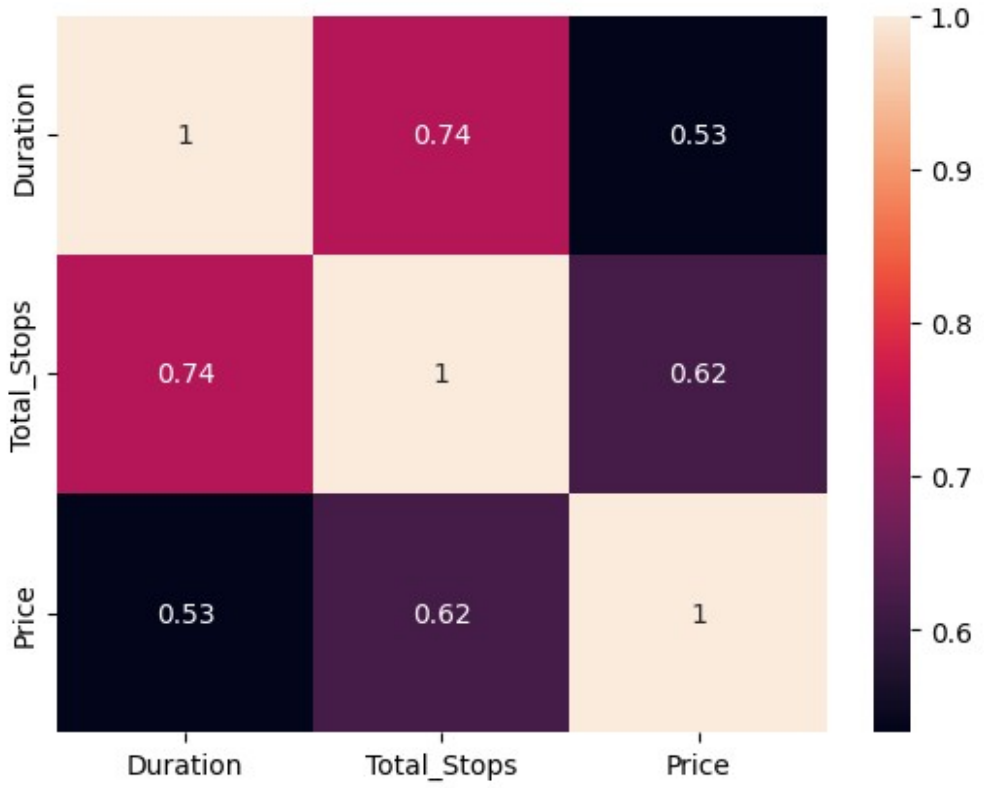
```
Additional_Info    589
Price              616
dtype: int64
Airline              0
Date_of_Journey      0
Source               0
Destination          0
Route                0
Dep_Time             0
Arrival_Time         0
Duration             0
Total_Stops          0
Additional_Info    300
Price              368
dtype: int64
Airline
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
Name: Price, dtype: float64
==========================================
{'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}
==========================================
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                0
dtype: int64
```

```
print(planes["Total_Stops"].value_counts())
```

```
Total_Stops
1 stop      4467
non-stop    2786
2 stops     1219
3 stops       35
```

```
4 stops        1
Name: count, dtype: int64
```

```
planes["Total_Stops"] = planes["Total_Stops"].str.replace(" Stops",
"")
planes["Total_Stops"] = planes["Total_Stops"].str.replace(" Stop", "")
planes["Total_Stops"] = planes["Total_Stops"].str.replace("\\D",
"",regex=True)
planes["Total_Stops"] = planes["Total_Stops"].astype(int)

sns.heatmap(planes.corr(numeric_only=True),annot=True)
plt.show()
```

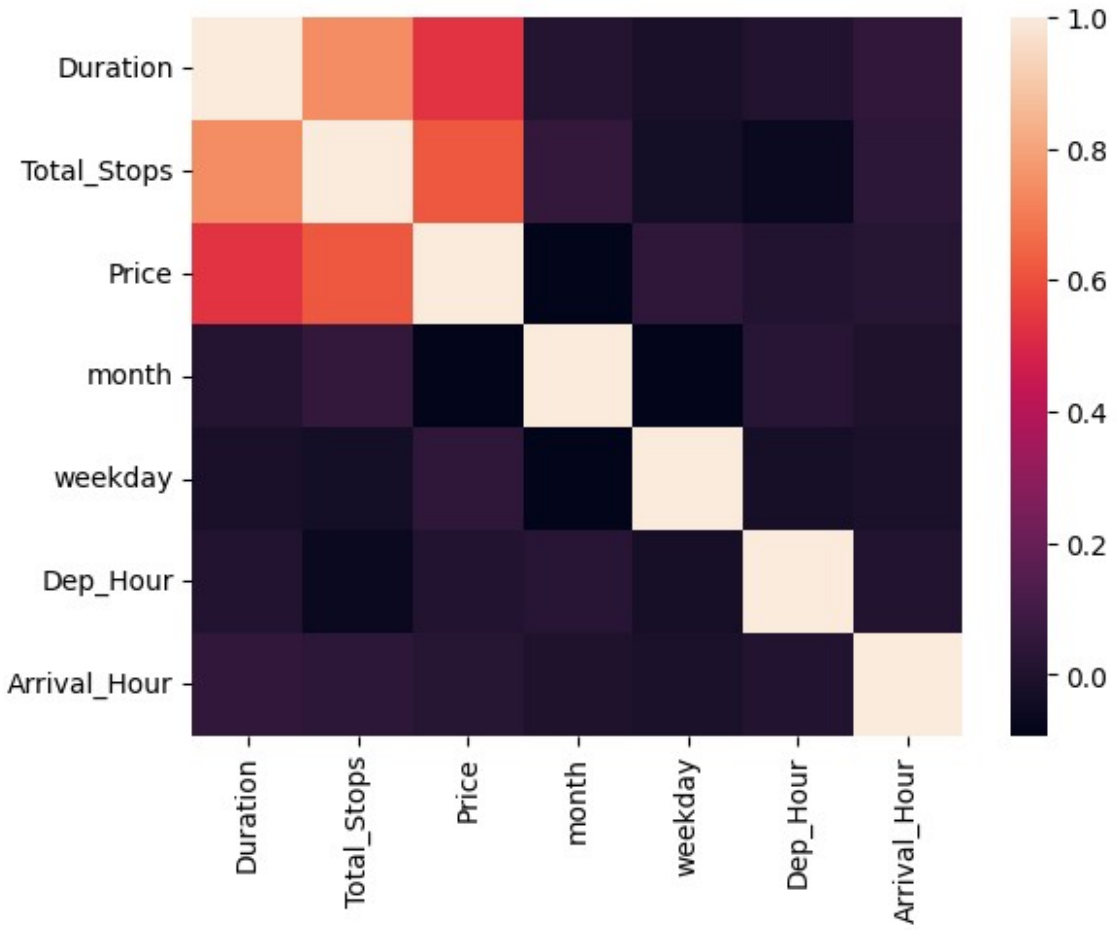



```
planes["month"] = planes["Date_of_Journey"].dt.month
planes["weekday"] = planes["Date_of_Journey"].dt.weekday
planes[["month","weekday","Date_of_Journey"]].head(10)
```

	month	weekday	Date_of_Journey
0	6	6	2019-06-09
1	5	6	2019-05-12
2	3	4	2019-03-01
3	6	0	2019-06-24
4	3	1	2019-03-12
5	3	4	2019-03-01
6	3	1	2019-03-12

```
7       5         0       2019-05-27
8       6         5       2019-06-01
9       4         3       2019-04-18
```

```
planes['Dep_Time'] = pd.to_datetime(planes['Dep_Time'],
format='mixed')
planes['Arrival_Time'] = pd.to_datetime(planes['Arrival_Time'],
format='mixed')

planes["Dep_Hour"] = planes["Dep_Time"].dt.hour
planes["Arrival_Hour"] = planes["Arrival_Time"].dt.hour

sns.heatmap(planes.corr(numeric_only=True))
plt.show()
```




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

```
count     8508.000000
mean      9033.468441
std       4366.382574
min       1759.000000
```

```
25%       5228.000000
50%       8452.000000
75%      12242.000000
max      54826.000000
Name: Price, dtype: float64
```

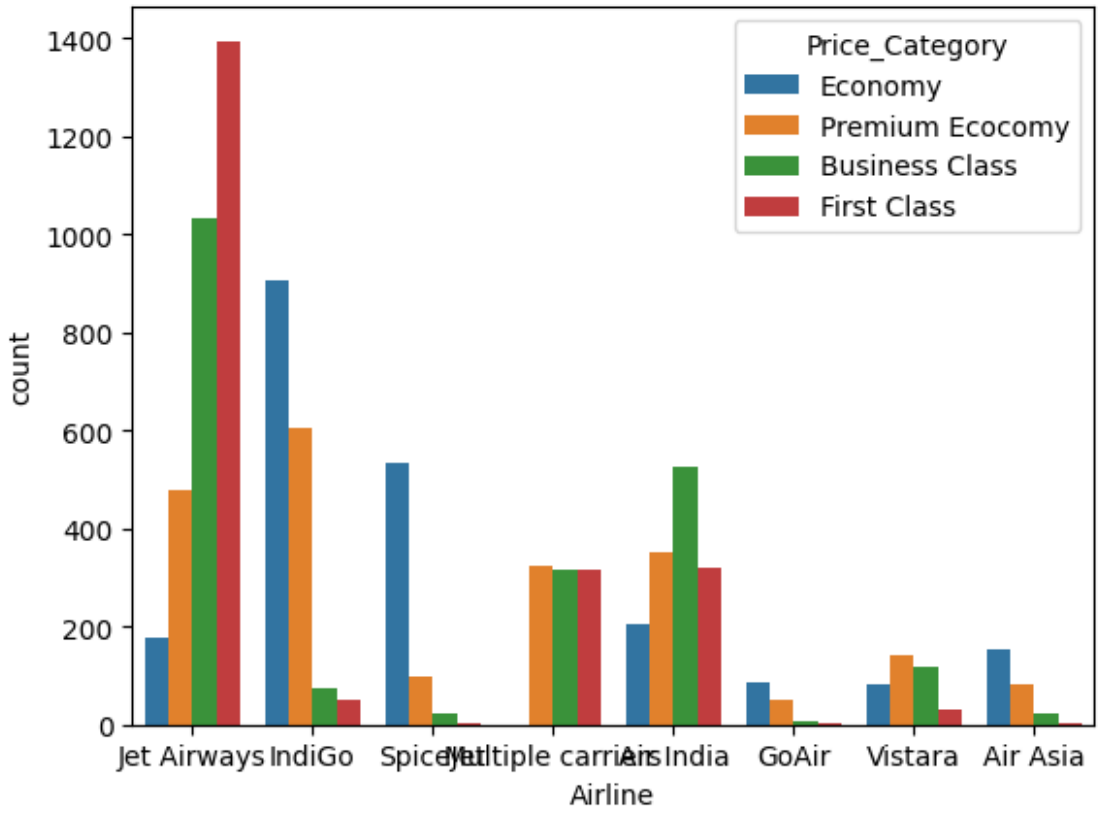
```
twenty_fifth = planes["Price"].quantile(0.25)
median = planes["Price"].median()
seventy_fifth = planes["Price"].quantile(0.75)
maximum = planes["Price"].max()
```

```
labels = ["Economy","Premium Ecocomy", "Business Class", "First
Class"]
bins = [0, twenty_fifth, median, seventy_fifth, maximum]
```

```
planes["Price_Category"] = pd.cut(planes["Price"],labels=labels,bins =
bins)
planes[["Price","Price_Category"]].head()
```

```
     Price   Price_Category
0  13882.0      First Class
1   6218.0  Premium Ecocomy
2  13302.0      First Class
3   3873.0          Economy
4  11087.0   Business Class
```

```
sns.countplot(data=planes,x="Airline",hue = "Price_Category")
plt.show()
```




```
salaries =
pd.read_csv("Salaries_with_date_of_response.csv",index_col=0,parse_dat
es=['date_of_response'])
print(salaries.dtypes)
salaries.head()
```

```
Designation                object
date_of_response           object
Experience                 object
Employment_Status          object
Salary_In_Rupees          float64
Employee_Location          object
Company_Location           object
Company_Size               object
Remote_Working_Ratio        int64
Salary_USD                float64
Job_Category               object
dtype: object
```

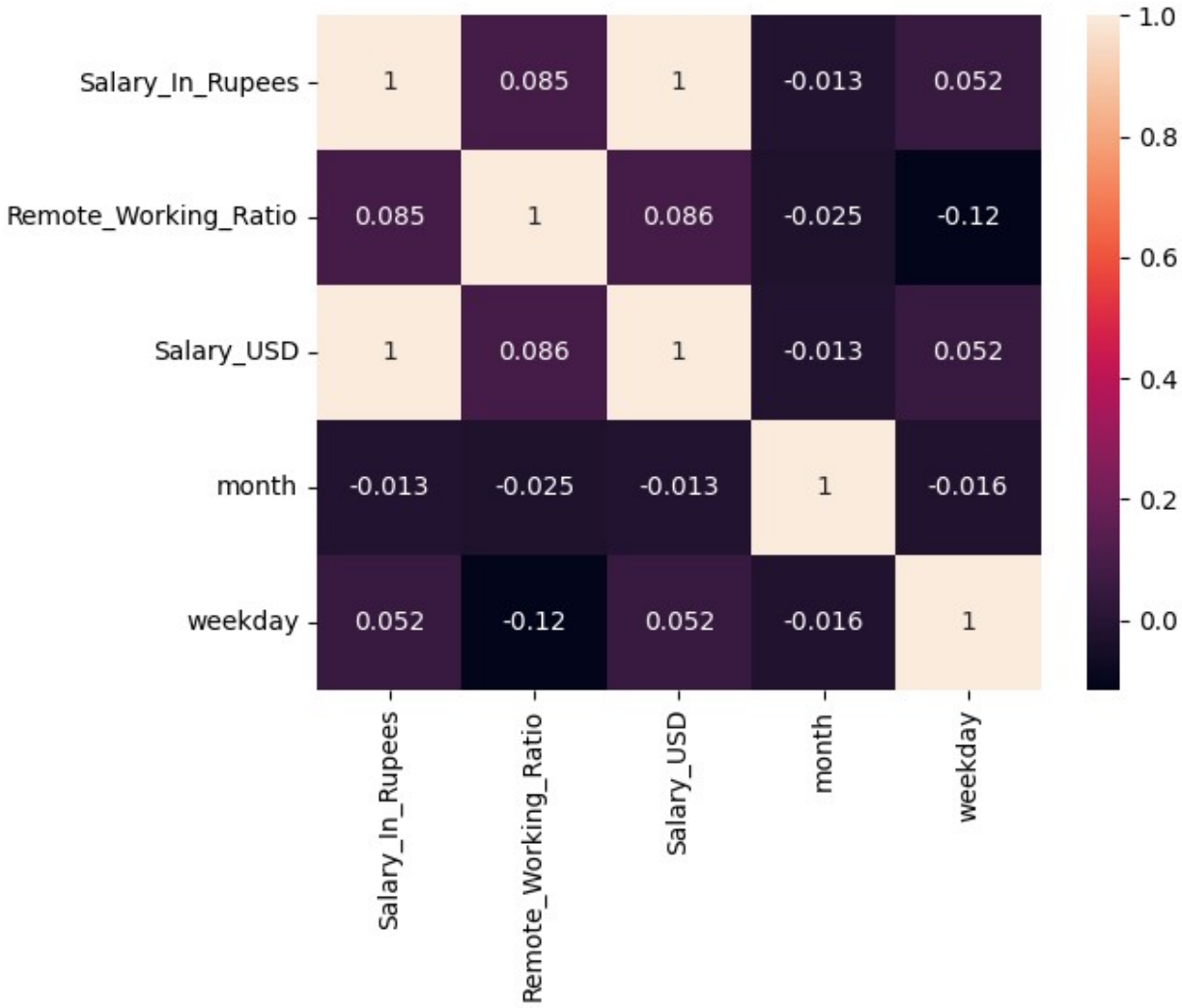
```
                  Designation date_of_response Experience
Employment_Status  \
0  Machine Learning Scientist         7/1/2020         SE
FT
1           Big Data Engineer        19/9/2020         SE
```

```
FT
2        Product Data Analyst       21/11/2020         MI
FT
3   Machine Learning Engineer       29/11/2020         SE
FT
4                Data Analyst         7/9/2020         EN
FT

   Salary_In_Rupees Employee_Location Company_Location Company_Size  \
0        20700000.0                JP                JP            S
1         8680000.0                GB                GB            M
2         1590000.0                HN                HN            S
3        11900000.0                US                US            L
4         5730000.0                US                US            L

   Remote_Working_Ratio  Salary_USD      Job_Category
0                     0  248256.840  Machine Learning
1                    50  104099.820  Data Engineering
2                     0   19096.680    Data Analytics
3                    50  143225.100  Machine Learning
4                   100   68748.048    Data Analytics
```

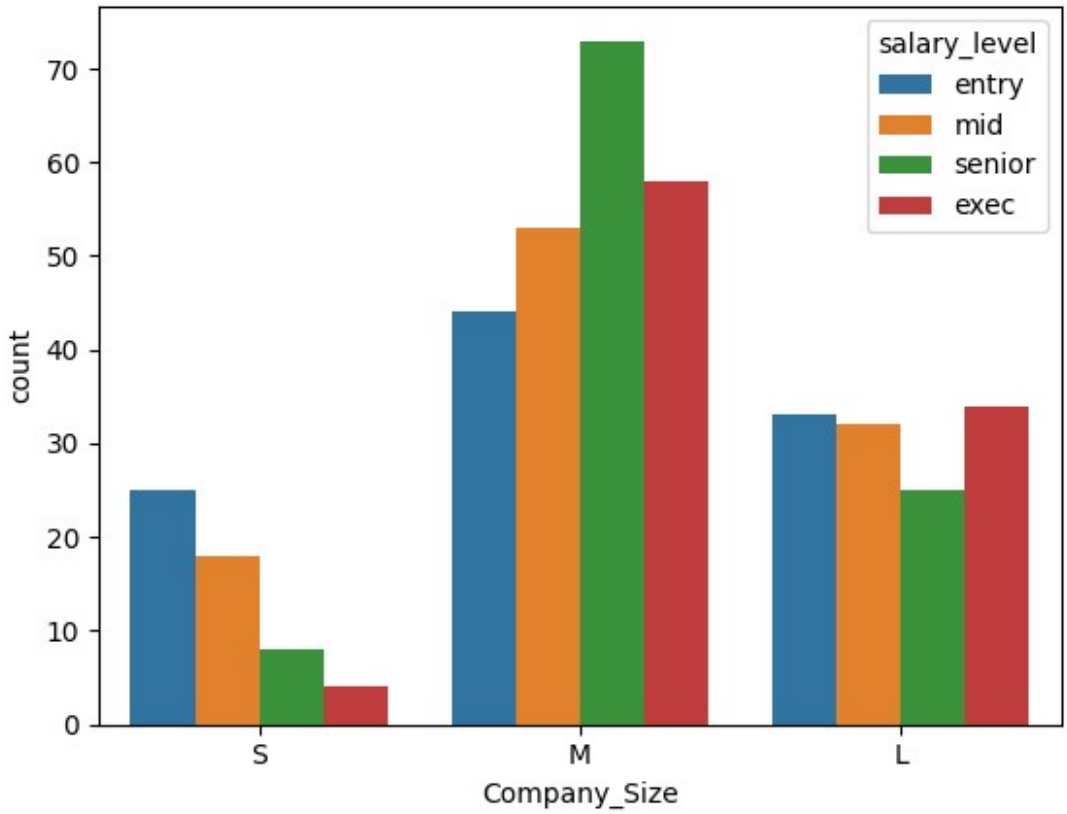
```
salaries["date_of_response"] =
pd.to_datetime(salaries["date_of_response"],format="mixed")
salaries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 407 entries, 0 to 406
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Designation           407 non-null    object
 1   date_of_response      407 non-null    datetime64[ns]
 2   Experience            407 non-null    object
 3   Employment_Status     407 non-null    object
 4   Salary_In_Rupees      407 non-null    float64
 5   Employee_Location     407 non-null    object
 6   Company_Location      407 non-null    object
 7   Company_Size          407 non-null    object
 8   Remote_Working_Ratio  407 non-null    int64
 9   Salary_USD            407 non-null    float64
 10  Job_Category          407 non-null    object
dtypes: datetime64[ns](1), float64(2), int64(1), object(7)
memory usage: 38.2+ KB
```

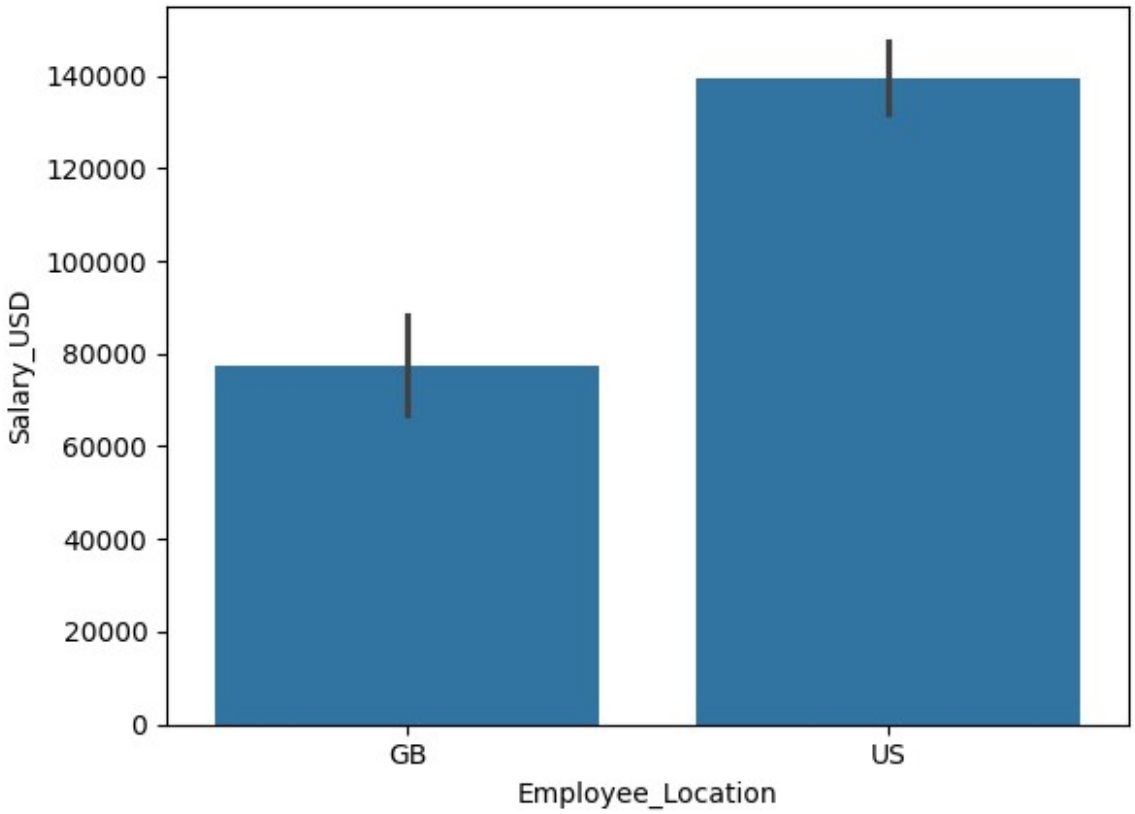
```
salaries['month'] = salaries['date_of_response'].dt.month
salaries['weekday'] = salaries['date_of_response'].dt.weekday
sns.heatmap(salaries.corr(numeric_only=True),annot=True)
plt.show()
```



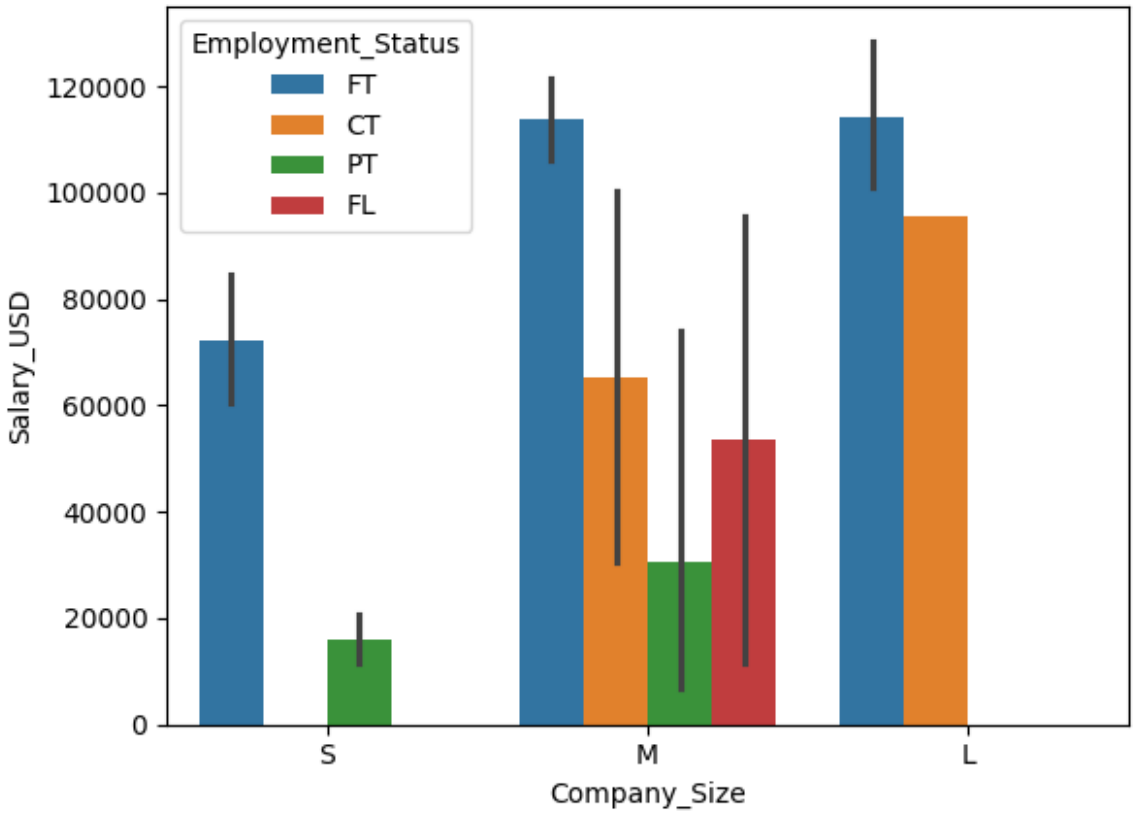

```
twenty_fifth = salaries["Salary_USD"].quantile(0.25)
salaries_median = salaries["Salary_USD"].median()
seventy_fifth = salaries["Salary_USD"].quantile(0.75)
maximum = salaries["Salary_USD"].max()

salary_labels = ['entry','mid','senior','exec']
salary_ranges = [0,twenty_fifth,salaries_median,seventy_fifth,maximum]

salaries["salary_level"] =
pd.cut(salaries["Salary_USD"],labels=salary_labels,bins =
salary_ranges)
sns.countplot(data=salaries,x="Company_Size",hue="salary_level")
plt.show()
```




```
usa_and_gb = salaries[(salaries["Employee_Location"] == "US") |
(salaries["Employee_Location"] == "GB")]
sns.barplot(data = usa_and_gb,x="Employee_Location",y="Salary_USD")
plt.show()
```




```
sns.barplot(data = salaries,
x="Company_Size",y="Salary_USD",hue="Employment_Status")
plt.show()
```


Employment_Status
FT
CT
PT
FL
120000
100000
80000
60000
40000
20000
0
Salary_USD
S
M
L
Company_Size