In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [2]:
```
divorce = pd.read_csv("divorce.csv")
divorce.dtypes
```

Out[2]:
```
divorce_date          object
dob_man               object
education_man         object
income_man           float64
dob_woman             object
education_woman       object
income_woman         float64
marriage_date         object
marriage_duration    float64
num_kids             float64
dtype: object
```

In [3]:
```
divorce = pd.read_csv("divorce.csv" , parse_dates=["marriage_date"])
divorce.dtypes
```

Out[3]:
```
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
```

In [4]:
```
divorce["marriage_date"] = pd.to_datetime(divorce["marriage_date"])
divorce.dtypes
```

Out[4]:
```
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
```

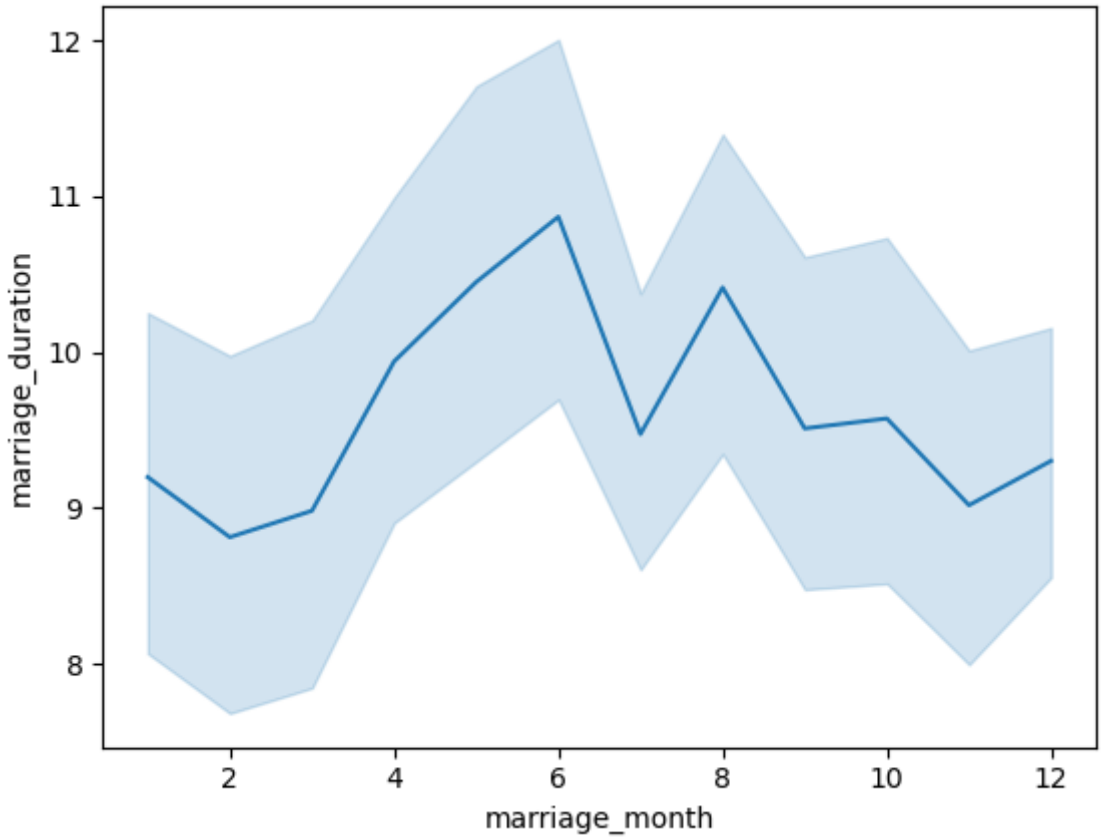
In [5]:
```
# divorce["marriage_date"] = pd.to_datetime(divorce[["month" , "day" , "year
# divorce.head(2)
```

In [6]:
```
divorce["marriage_month"] = divorce["marriage_date"].dt.month
divorce.head()
```

Out[6]:

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

In [7]:
```
sns.lineplot(data=divorce, x="marriage_month", y="marriage_duration")
plt.show()
```



In [8]:
```
DataFrame = pd.read_csv("divorce.csv")
DataFrame.dtypes
```

Out[8]:
```
divorce_date          object
dob_man               object
education_man         object
income_man           float64
dob_woman             object
education_woman       object
income_woman         float64
marriage_date         object
marriage_duration    float64
num_kids             float64
dtype: object
```

1

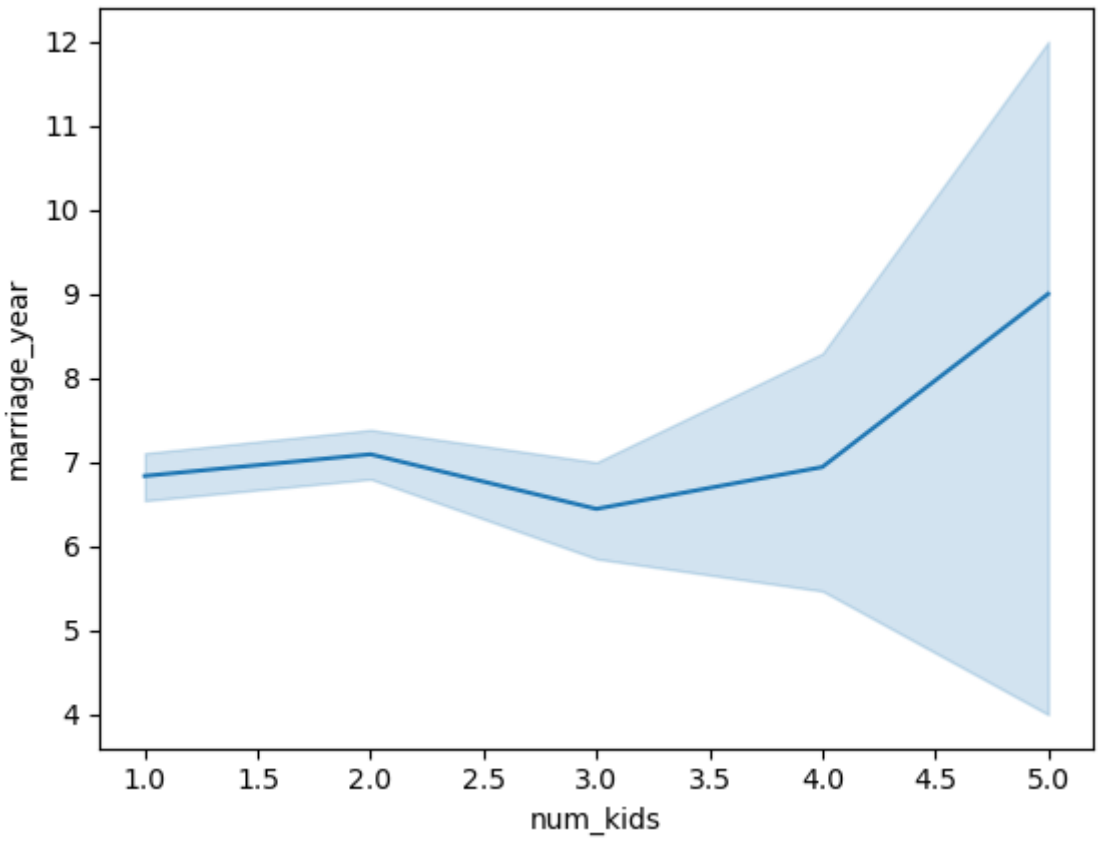
In [10]:
```
divorce = pd.read_csv("divorce.csv" , parse_dates=["divorce_date" , "dob_man
divorce.dtypes
```

Out[10]:
```
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
```

2

In [11]:
```
divorce["marriage_date"] = pd.to_datetime(divorce["marriage_date"])
divorce["marriage_year"] = divorce["marriage_date"].dt.month
sns.lineplot(data=divorce, x = "num_kids" , y = "marriage_year")
```

Out[11]:
```
<AxesSubplot:xlabel='num_kids', ylabel='marriage_year'>
```



CORRELATION

In [47]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import numpy as np
%matplotlib inline
```

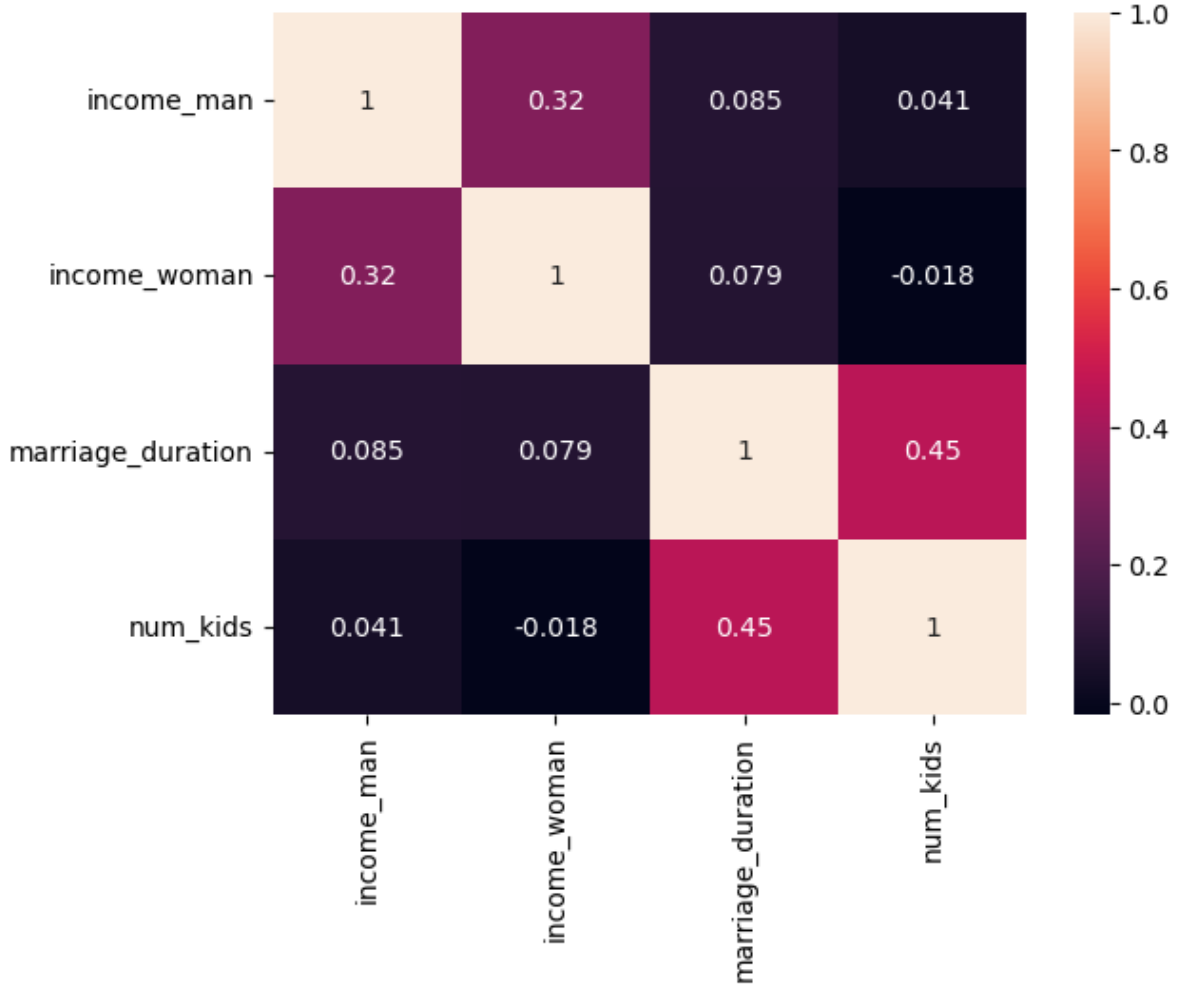
In [48]:
```
divorce = pd.read_csv("divorce.csv")
divorce.corr()
```

Out[48]:

	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

HEATMAP

In [49]:
```
sns.heatmap(divorce.corr(), annot = True)
plt.show()
```



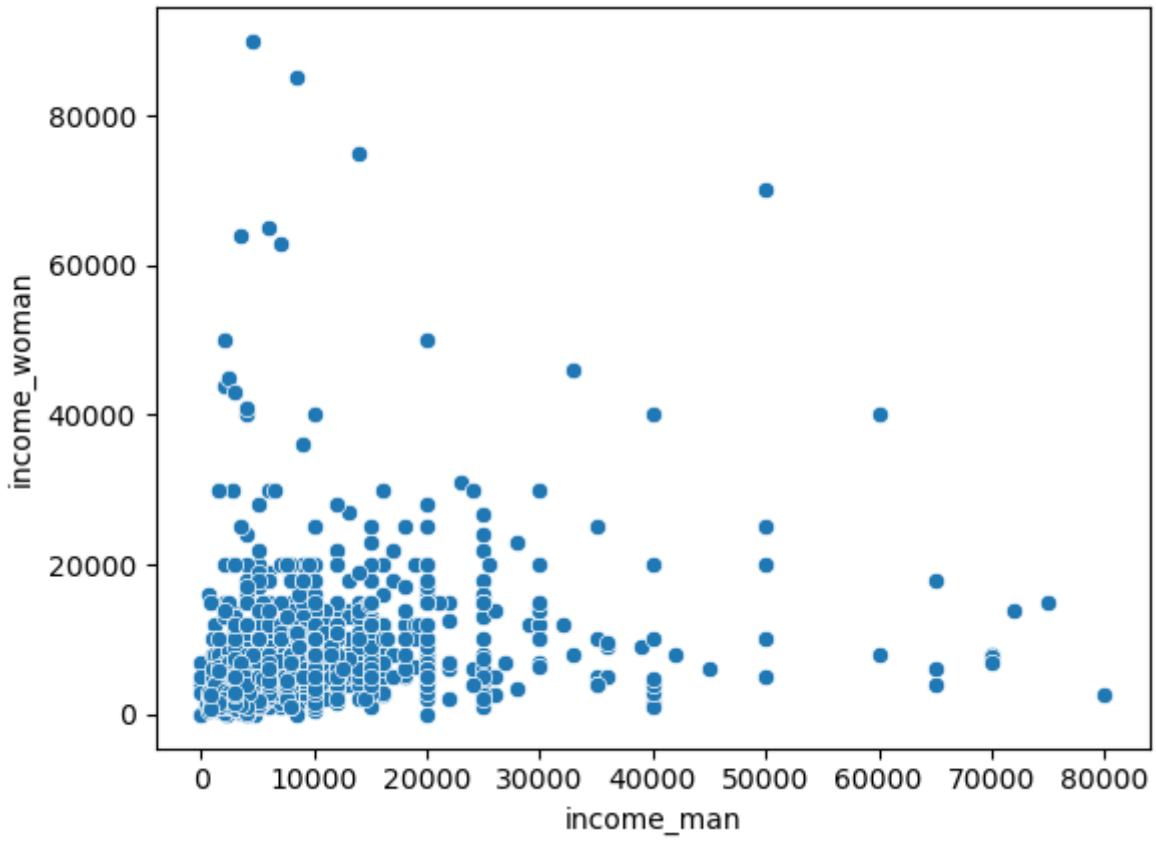
In [50]:
```
divorce["divorce_date"].min()
```

Out[50]:
```
'2000-01-08'
```

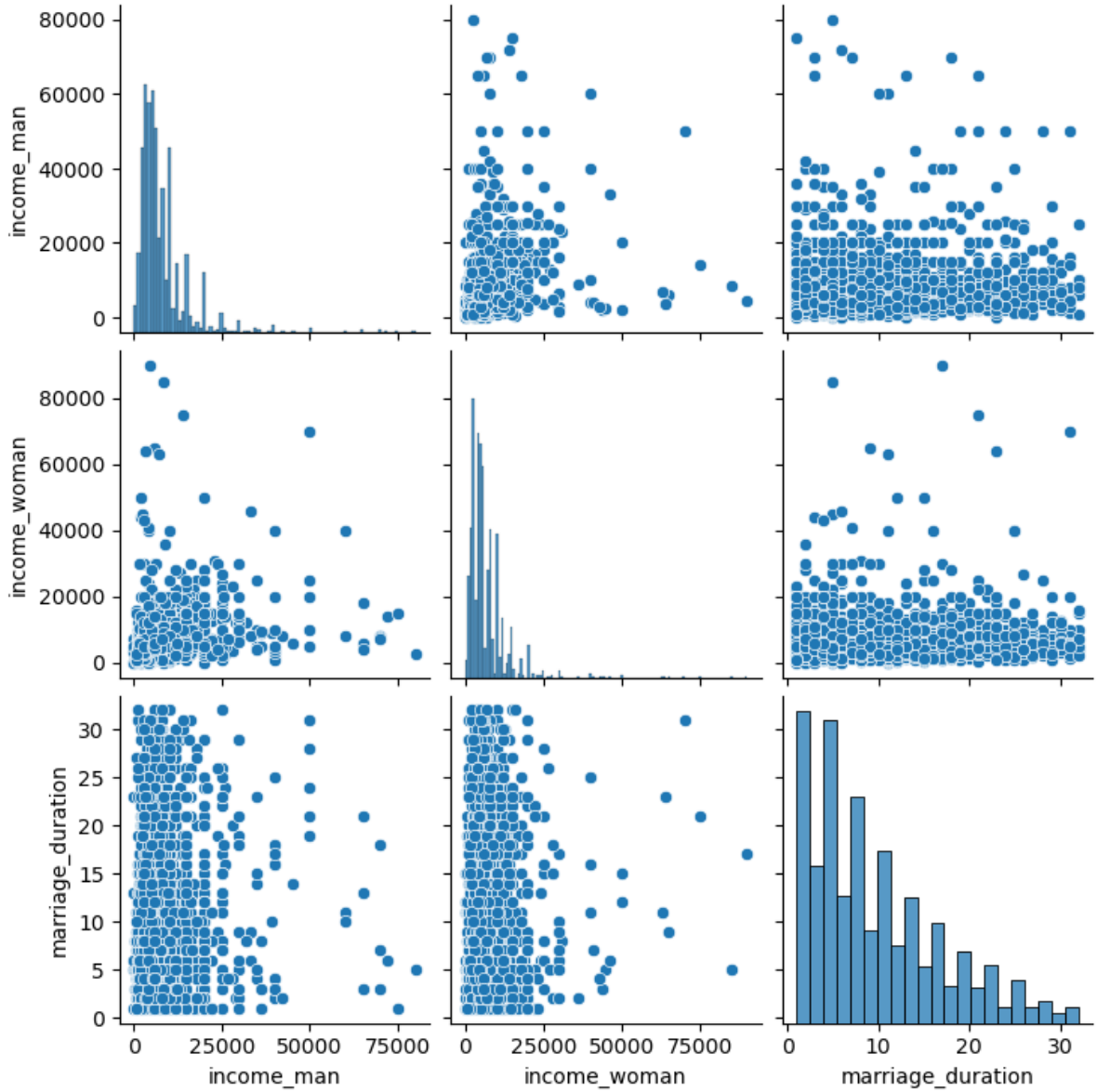
In [51]:
```
divorce["divorce_date"].max()
```

Out[51]:
```
'2015-11-03'
```

In [52]:
```
sns.scatterplot(data =divorce, x = "income_man" , y= "income_woman")
plt.show()
```

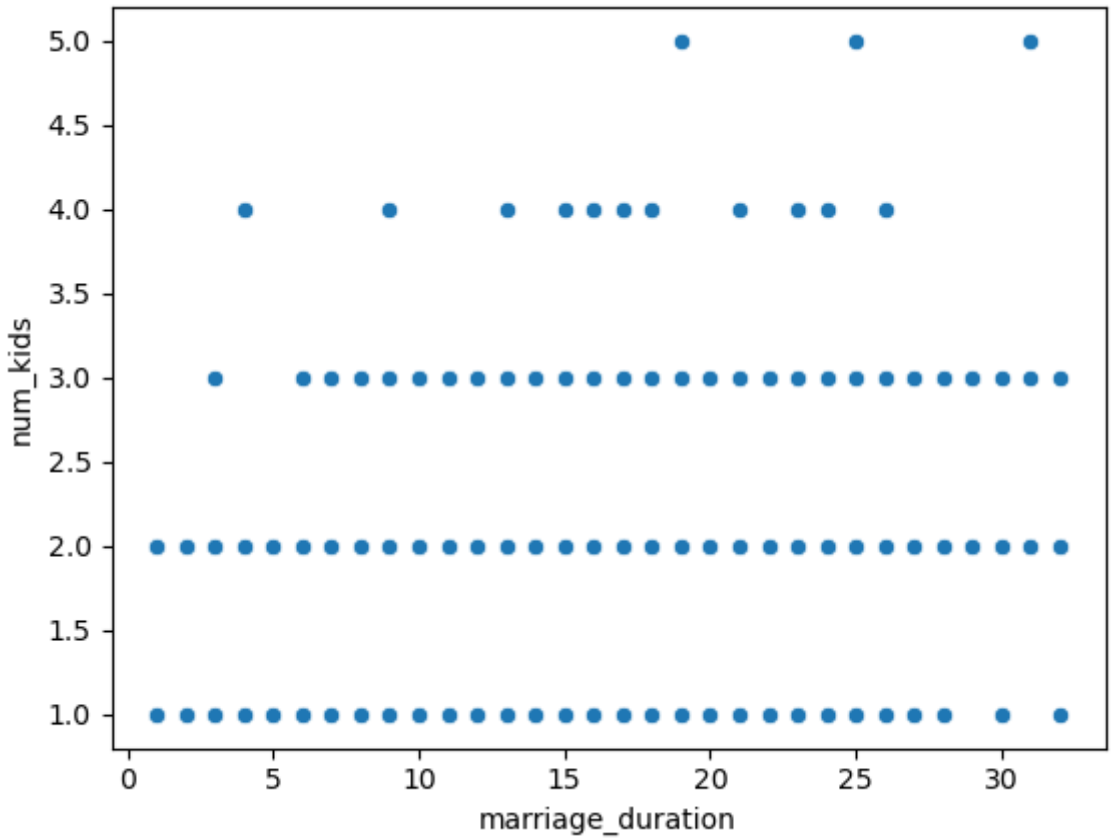


In [53]:
```
sns.pairplot(data=divorce , vars=["income_man", "income_woman" , "marriage_d
plt.show()
```



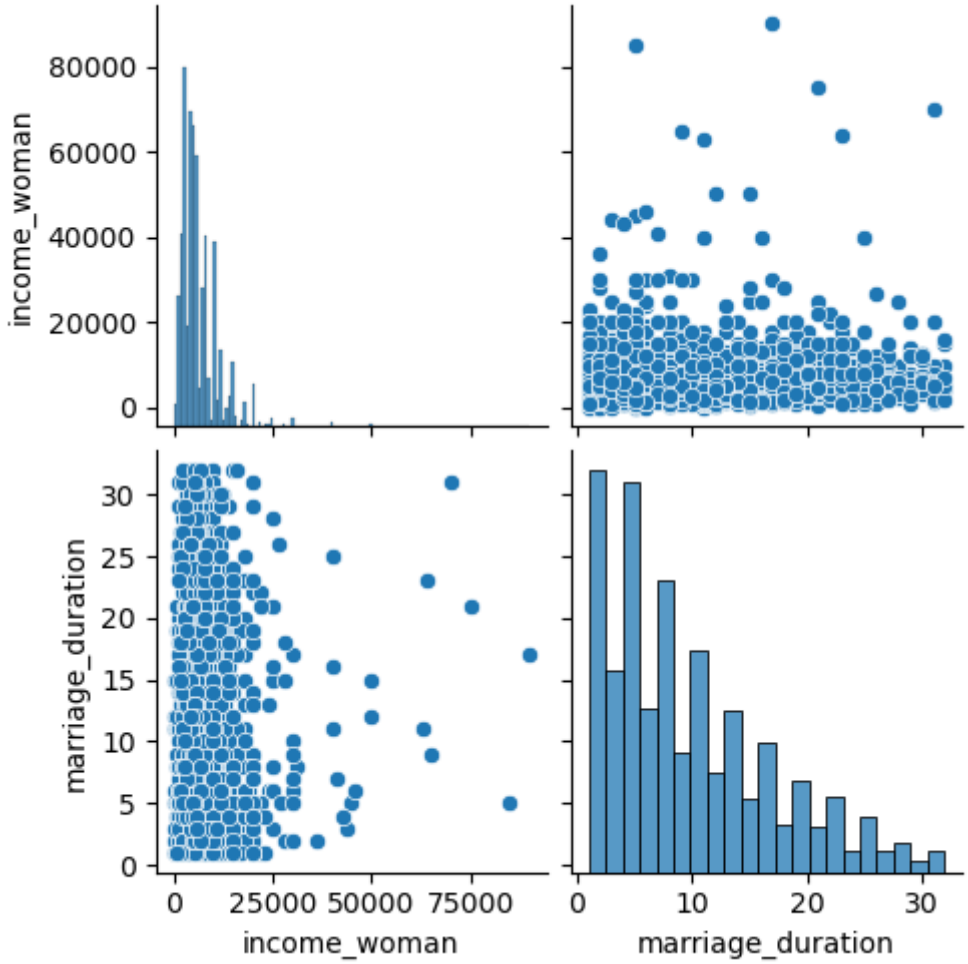
3

In [54]:
```
sns.scatterplot(data =divorce, x = "marriage_duration" , y= "num_kids")
plt.show()
```



4

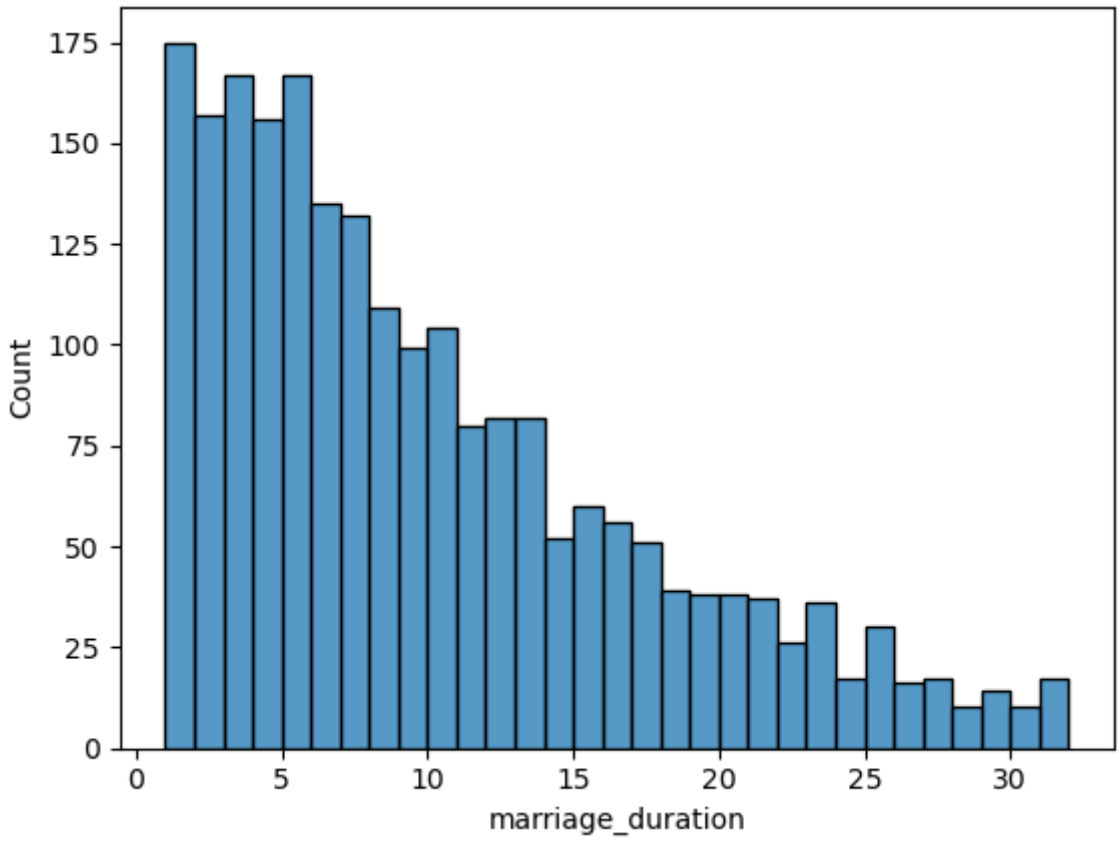
In [55]:
```
sns.pairplot(data=divorce , vars=["income_woman" , "marriage_duration"])
plt.show()
```



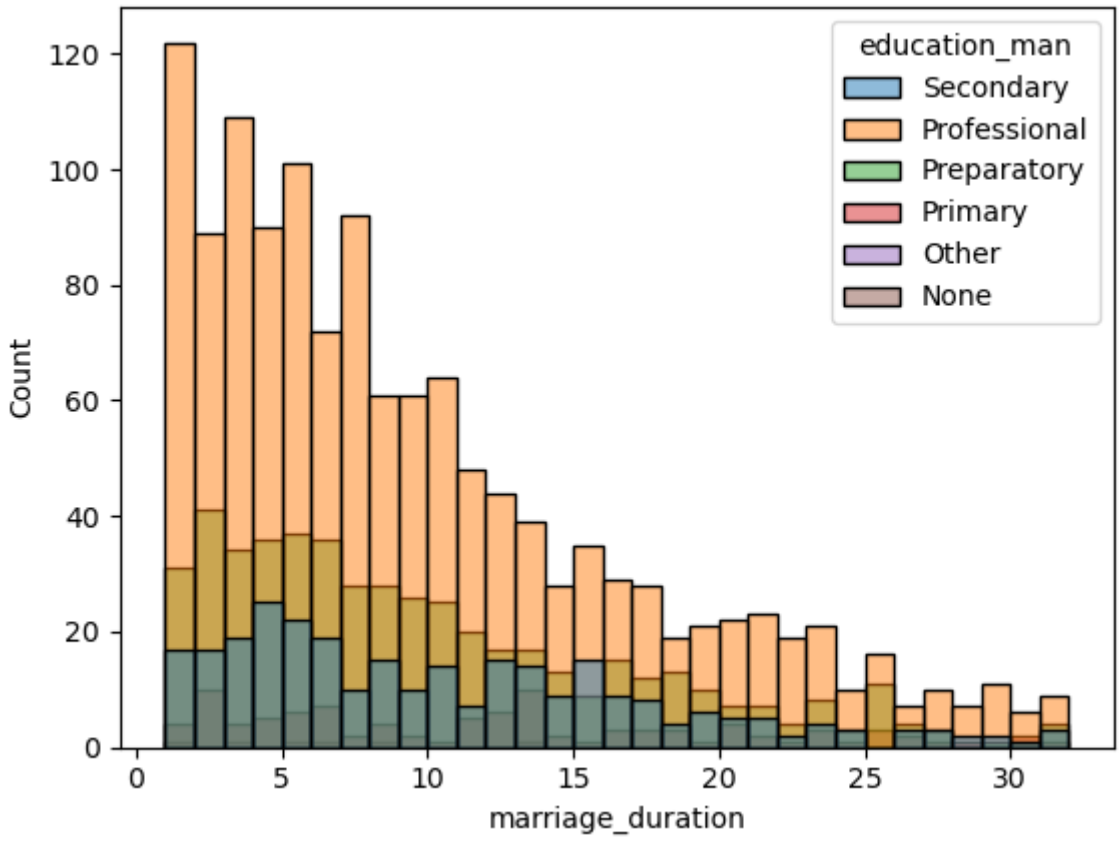
In [56]:
```
divorce["education_man"].value_counts()
```

Out[56]:
```
Professional    1313
Preparatory      501
Secondary        288
Primary          100
None               4
Other              3
Name: education_man, dtype: int64
```

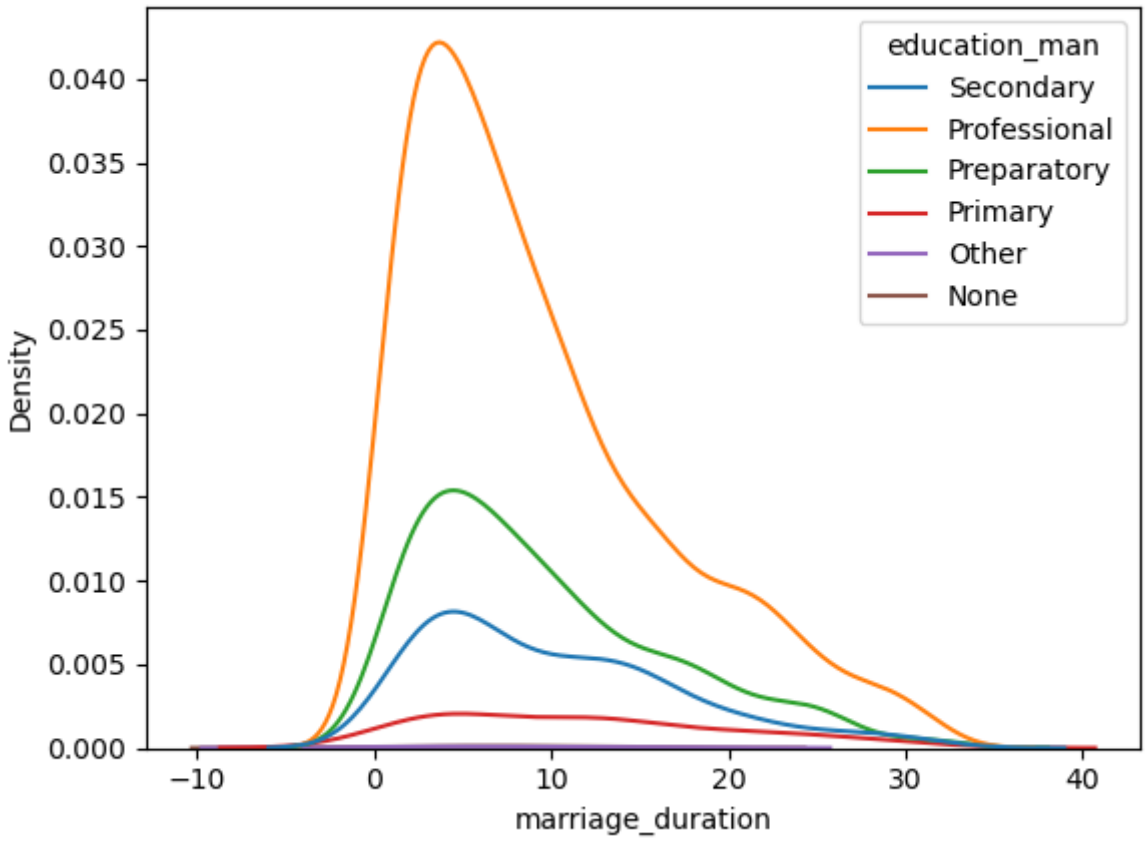
In [57]:
```
sns.histplot(data=divorce, x = "marriage_duration" , binwidth= 1)
plt.show()
```



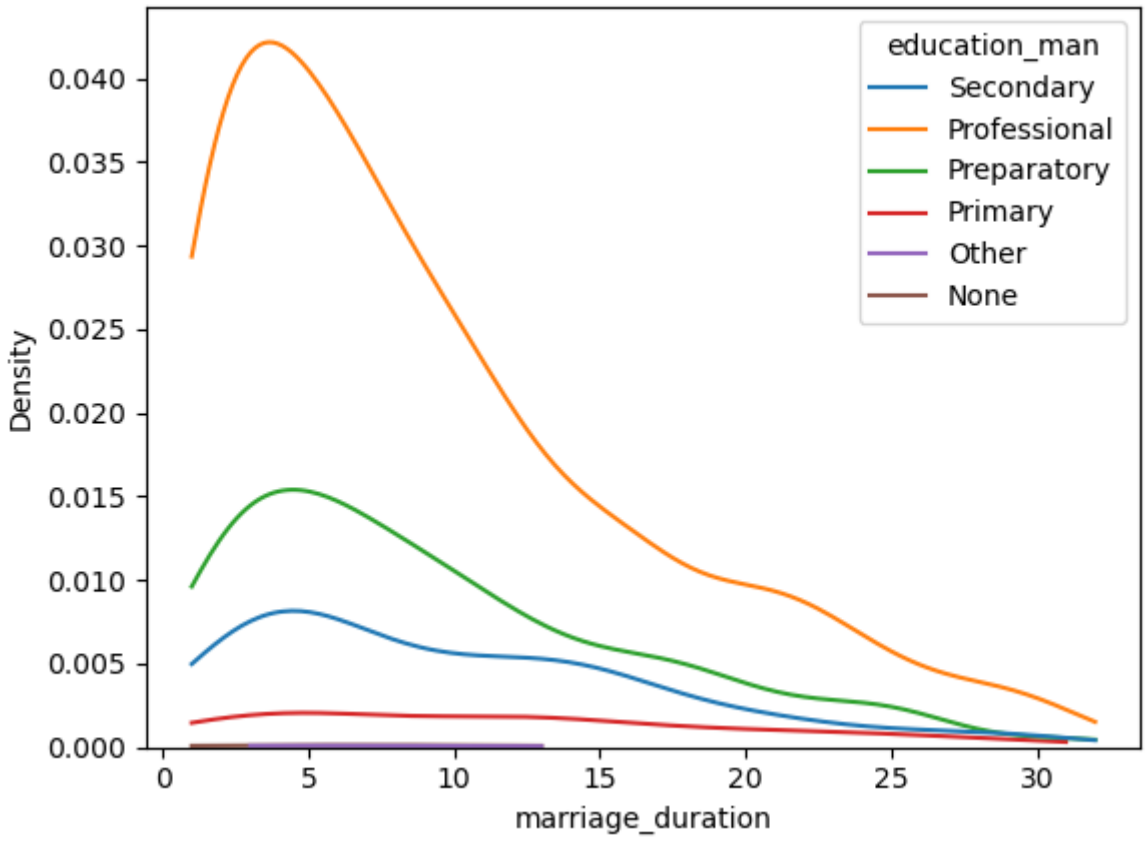
In [58]:
```
sns.histplot(data = divorce , x = "marriage_duration" , hue= "education_man"
plt.show()
```



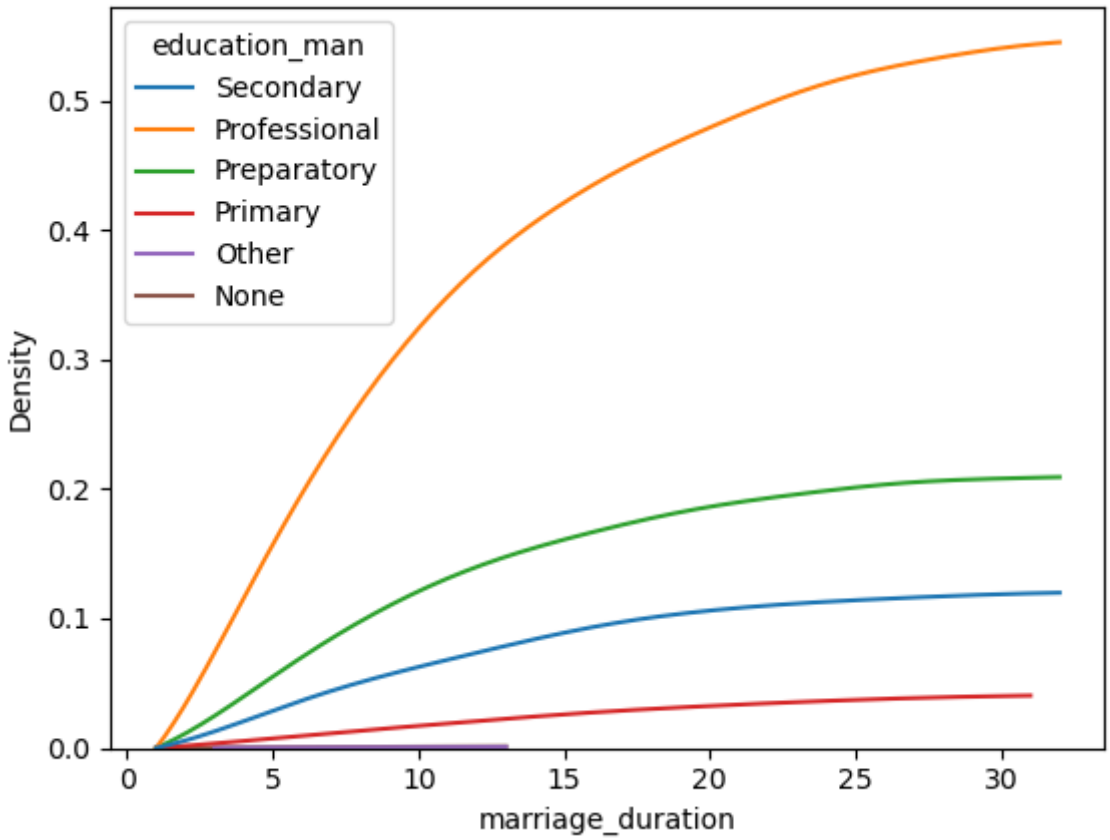
In [59]:
```
sns.kdeplot(data= divorce , x = "marriage_duration" , hue = "education_man")
plt.show()
```



In [60]:
```
sns.kdeplot(data=divorce , x = "marriage_duration" , hue="education_man" ,cu
plt.show()
```



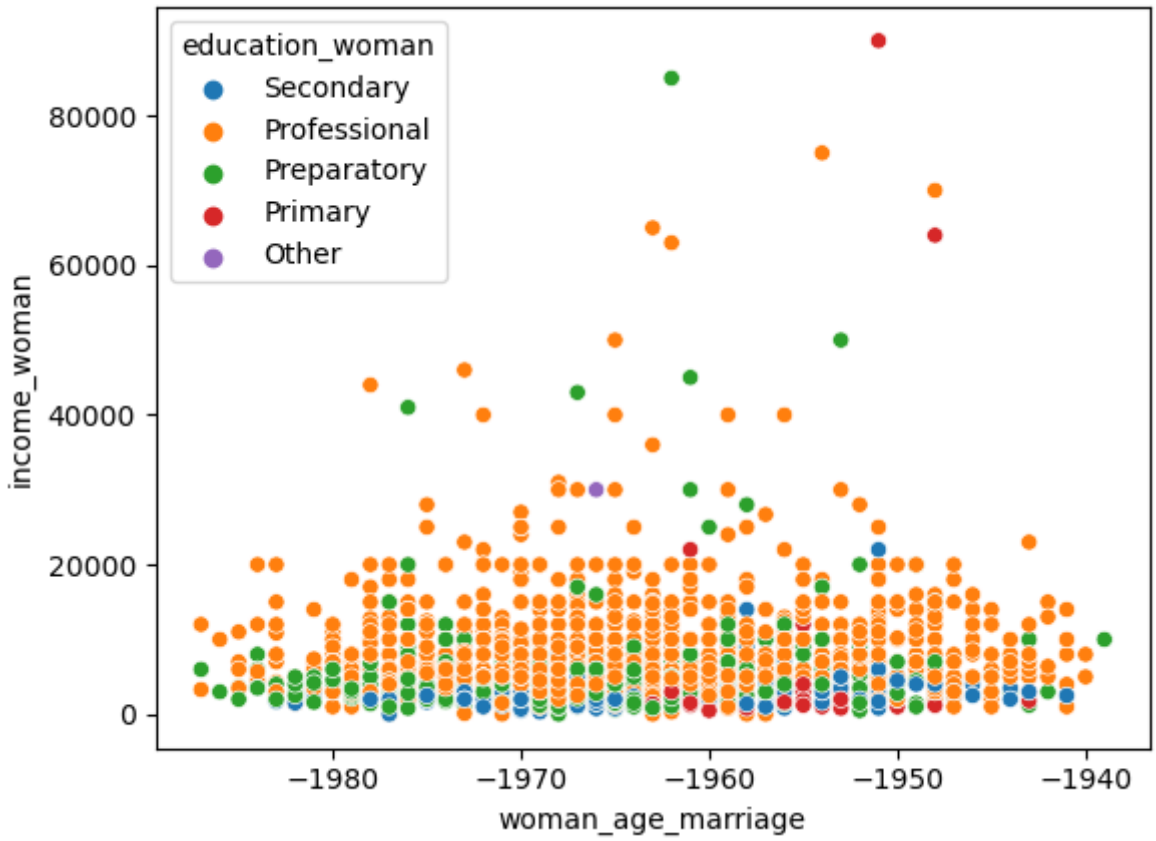
In [61]:
```
sns.kdeplot(data = divorce , x = "marriage_duration" , hue="education_man" ,
plt.show()
```



In [12]:
```
divorce["man_age_marriage"] = divorce["marriage_year"] - divorce["dob_man"].
divorce["woman_age_marriage"] = divorce["marriage_year"] - divorce["dob_woma
```

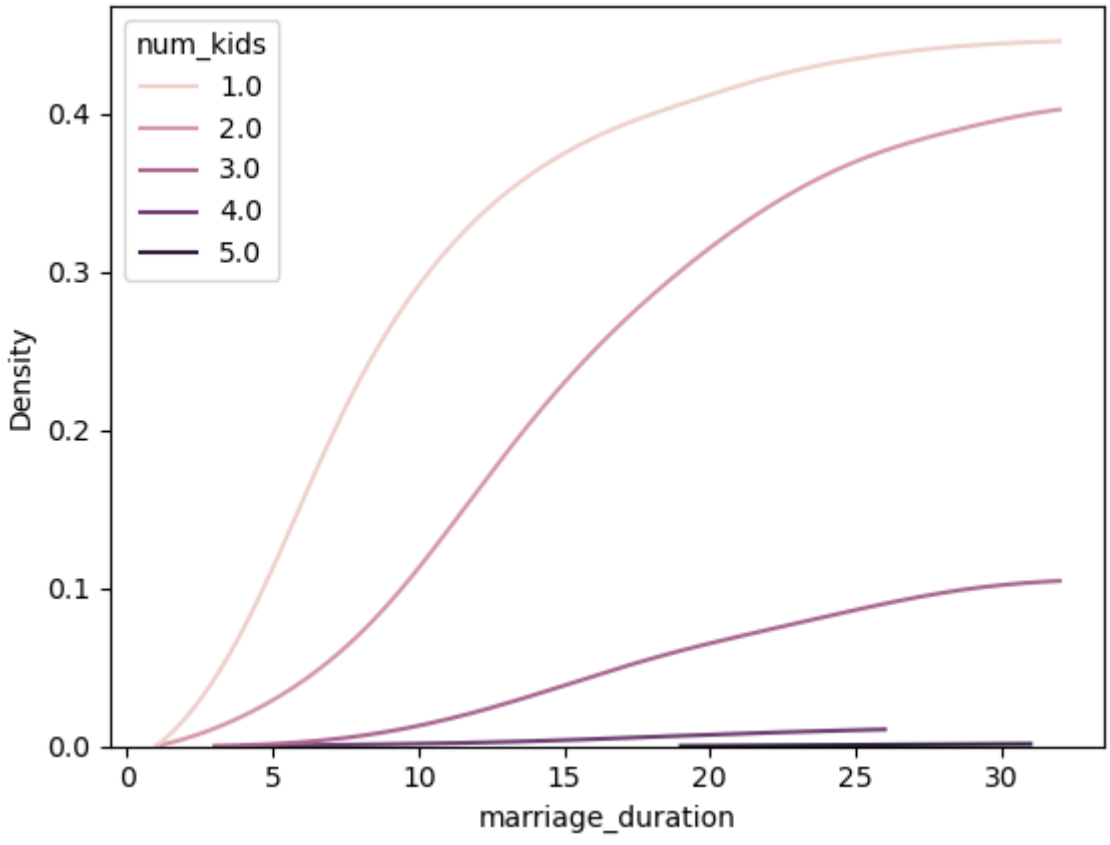
5

In [13]:
```
sns.scatterplot(data = divorce , x = "woman_age_marriage" , y = "income_woma
plt.show()
```



6

In [64]:
```
sns.kdeplot(data = divorce , x =  "marriage_duration" , hue = "num_kids", cu
plt.show()
```



CLASS FREQUEnCY

In [65]:
```
planes = pd.read_csv("Airlines_unclean.csv")
print(planes["Destination"].value_counts())
```

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

In [66]:
```
planes["Destination"].value_counts(normalize = True)
```

Out[66]:
```
Cochin       0.425773
Banglore     0.268884
Delhi        0.118200
New Delhi    0.086105
Hyderabad    0.065257
Kolkata      0.035780
Name: Destination, dtype: float64
```

In [67]:
```
pd.crosstab(planes["Source"] , planes["Destination"])
```

Out[67]:

Source \ Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
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

Aggregated values with pd.crosstab()

In [68]:
```
pd.crosstab(planes["Source"] , planes["Destination"] , values = planes["Pric
```

Out[68]:

Source \ Destination	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
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

7

In [69]:
```
salaries = pd.read_csv("Salary_Rupee_USD.csv", index_col = 0)
salaries["Job_Category"].value_counts(normalize = True)
```

Out[69]:
```
Data Science          0.277641
Data Engineering      0.272727
Data Analytics        0.226044
Machine Learning      0.120393
Other                 0.068796
Managerial            0.034398
Name: Job_Category, dtype: float64
```

8

In [70]:
```
pd.crosstab(salaries["Company_Size"] , salaries["Experience"])
```

Out[70]:

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

In [71]:
```
pd.crosstab(salaries["Job_Category"] , salaries["Company_Size"])
```

Out[71]:

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

In [72]:
```
pd.crosstab(salaries["Job_Category"] , salaries["Company_Size"] , values = s
```

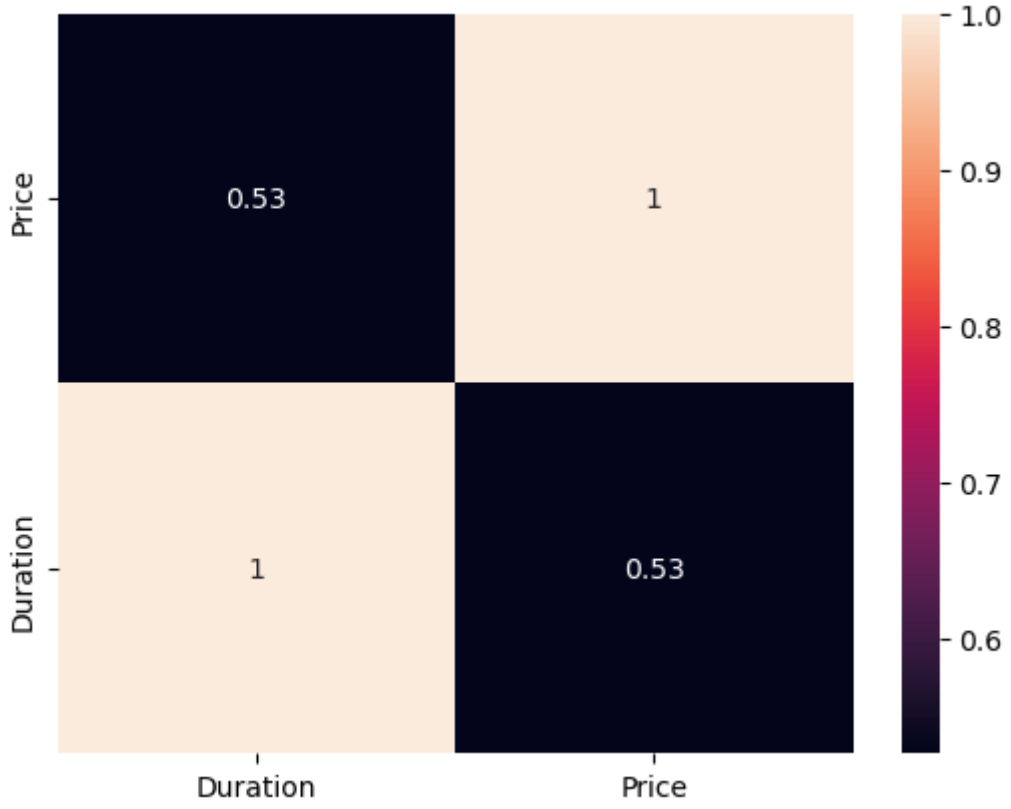
Out[72]:

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

CORRELATION

In [22]:
```
planes = pd.read_csv('Airlines_unclean.csv', index_col = 0,
parse_dates=['Date_of_Journey','Dep_Time','Arrival_Time'])
# 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(), annot=True)
ax.set_ylim([0,2])
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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  datetime64[ns]
 6   Arrival_Time     10466 non-null  datetime64[ns]
 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](3), float64(2), object(6)
memory usage: 999.4+ KB
None
```



TOTAL STOPS

In [23]:
```
print(planes["Total_Stops"].value_counts())
```

```
1 stop      5503
non-stop    3411
2 stops     1488
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

In [25]:
```
# 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("non-stop" , "0"
# planes["Total_Stops"] = planes["Total_Stops"].astype(int)
```

In [27]:
```
# sns.heatmap(planes.corr() ,annot = True)
# plt.show()
```

In [29]:
```
# print(planes.dtypes)
```

9

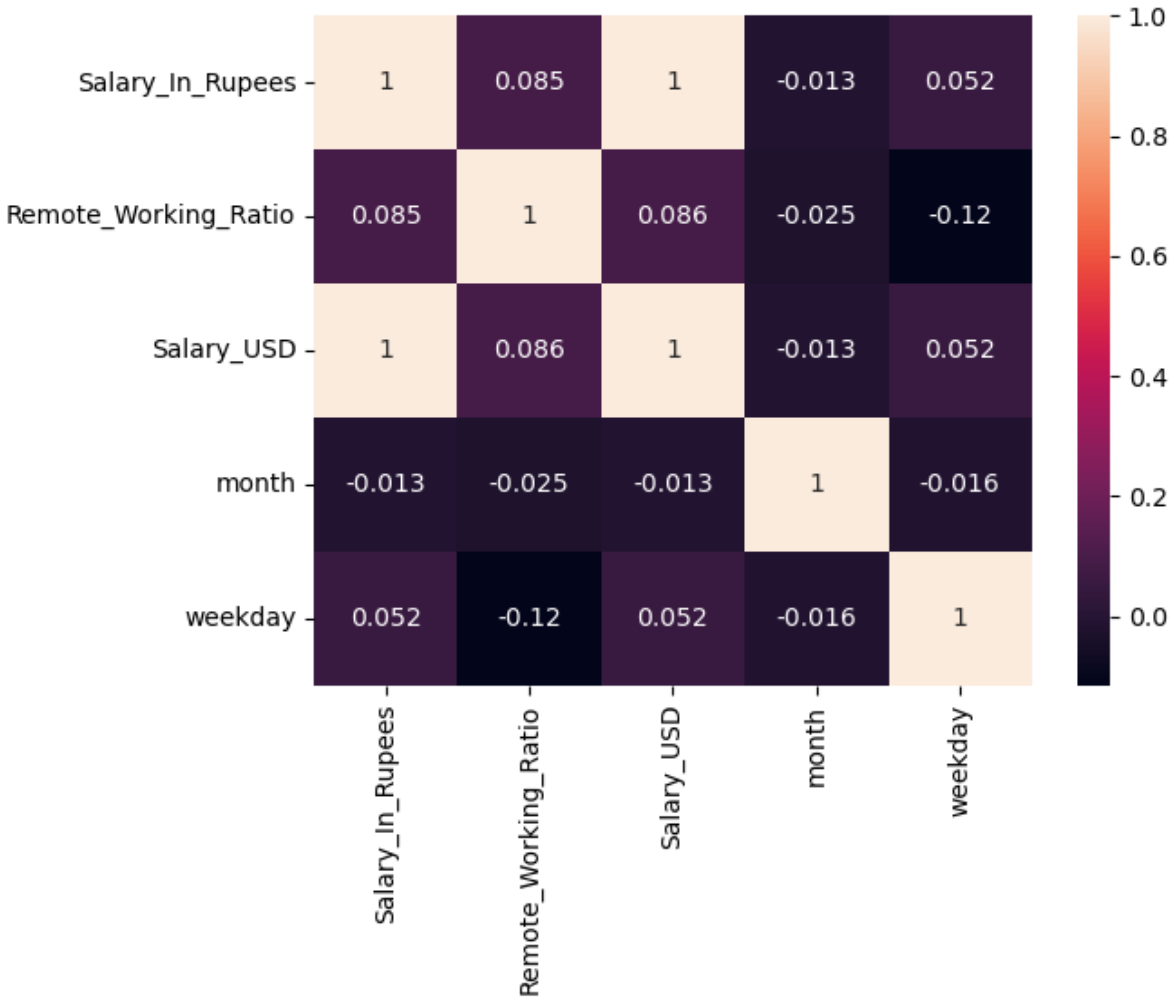
In [58]:
```
salaries = pd.read_csv("Salaries_with_date_of_response.csv", index_col = 0 ,
```

```
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '21/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '29/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '13/10/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '15/10/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '15/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '17/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '23/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '20/10/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '25/10/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '24/11/2020' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '20/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '13/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '24/11/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '25/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '30/11/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '17/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
```

```
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '20/11/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '17/11/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '26/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '28/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '29/11/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '23/10/2021' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '17/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '15/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '18/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '14/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '27/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '28/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '23/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '18/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '16/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '14/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
```

```
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '25/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '15/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '13/11/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
/Users/richard/opt/anaconda3/lib/python3.9/site-packages/pandas/io/parsers/b
ase_parser.py:1070: UserWarning: Parsing '13/10/2022' in DD/MM/YYYY format.
Provide format or specify infer_datetime_format=True for consistent parsing.
  return tools.to_datetime(
```

In [59]:
```
salaries["month"]  = salaries["date_of_response"].dt.month
salaries["weekday"] = salaries["date_of_response"].dt.weekday
sns.heatmap(salaries.corr(), annot = True)
plt.show()
```

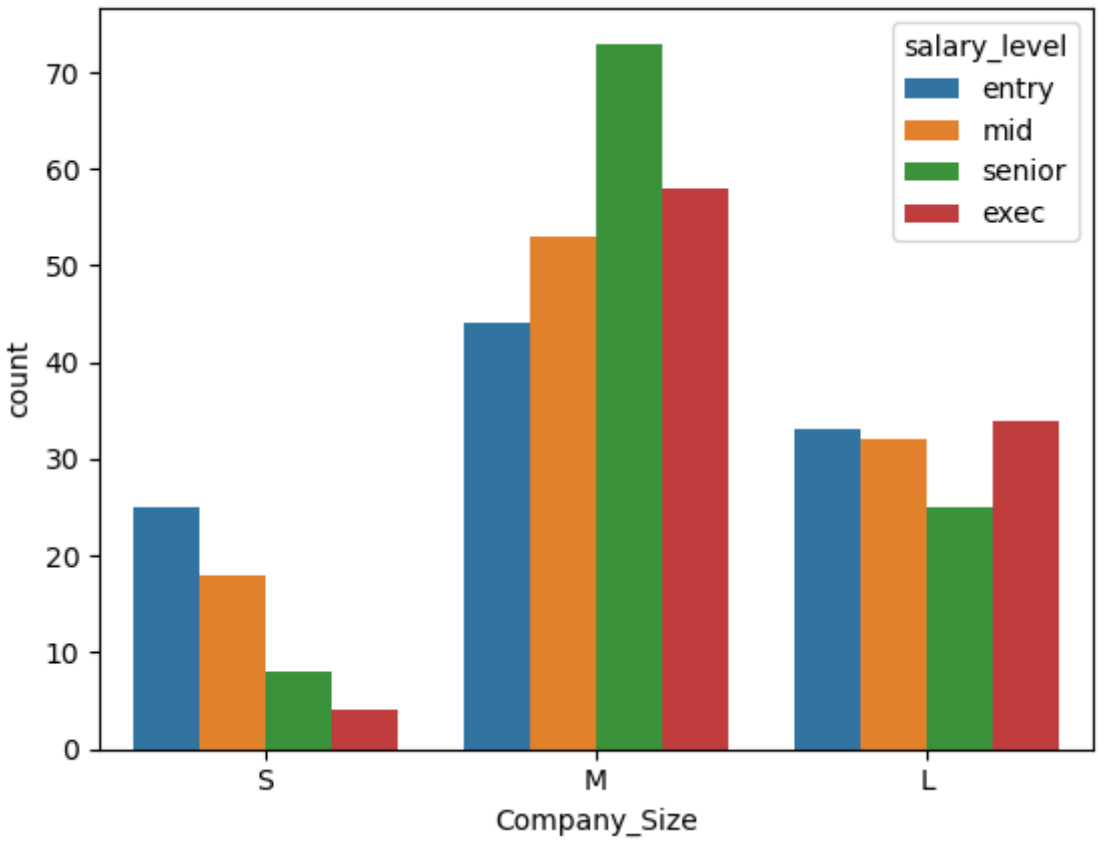


10

In [55]:
```
salaries = pd.read_csv("Salaries_with_date_of_response.csv", index_col = 0)
twenty_fifth = salaries["Salary_USD"].quantile(0.25)
salaries_median = salaries["Salary_USD"].median()
seventy_fifth = seventy_fifth = salaries["Salary_USD"].quantile(0.75)
print(twenty_fifth, salaries_median,seventy_fifth)
```

```
60880.691999999995 97488.552 143225.1
```

In [56]:
```
salary_labels = ["entry" , "mid" , "senior", "exec"]
salary_ranges = [0 ,twenty_fifth, salaries_median,seventy_fifth, salaries["S
salaries["salary_level"] = pd.cut(salaries["Salary_USD"], bins = salary_rang
sns.countplot(data=salaries , x = "Company_Size"  , hue="salary_level")
plt.show()
```



11

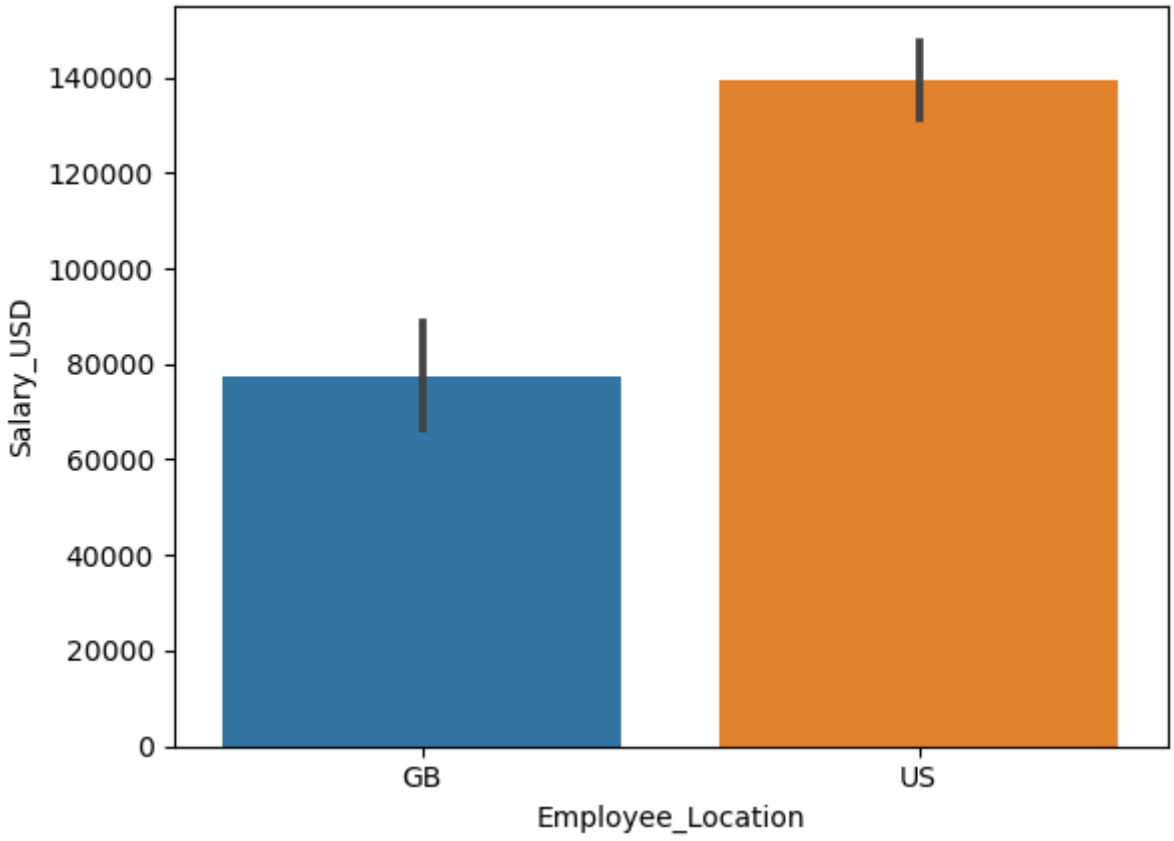
In [72]:
```
usa = salaries[salaries["Employee_Location"].isin(["US", "GB"])]
usa
```

Out[72]:

	Designation	date_of_response	Experience	Employment_Status	Salary_In_Rupees	Em
1	Big Data Engineer	2020-09-19	SE	FT	8680000.0	
3	Machine Learning Engineer	2020-11-29	SE	FT	11900000.0	
4	Data Analyst	2020-07-09	EN	FT	5730000.0	
5	Lead Data Scientist	2020-04-08	SE	FT	15100000.0	
7	Business Data Analyst	2020-10-13	MI	FT	10700000.0	
...	...	...	...	...	...	
400	Data Scientist	2022-02-26	MI	FT	10300000.0	
402	Data Engineer	2022-10-27	SE	FT	12300000.0	
403	Data Engineer	2022-08-25	SE	FT	10000000.0	
404	Data Analyst	2022-11-08	SE	FT	10300000.0	
405	Data Analyst	2022-06-15	SE	FT	11900000.0	

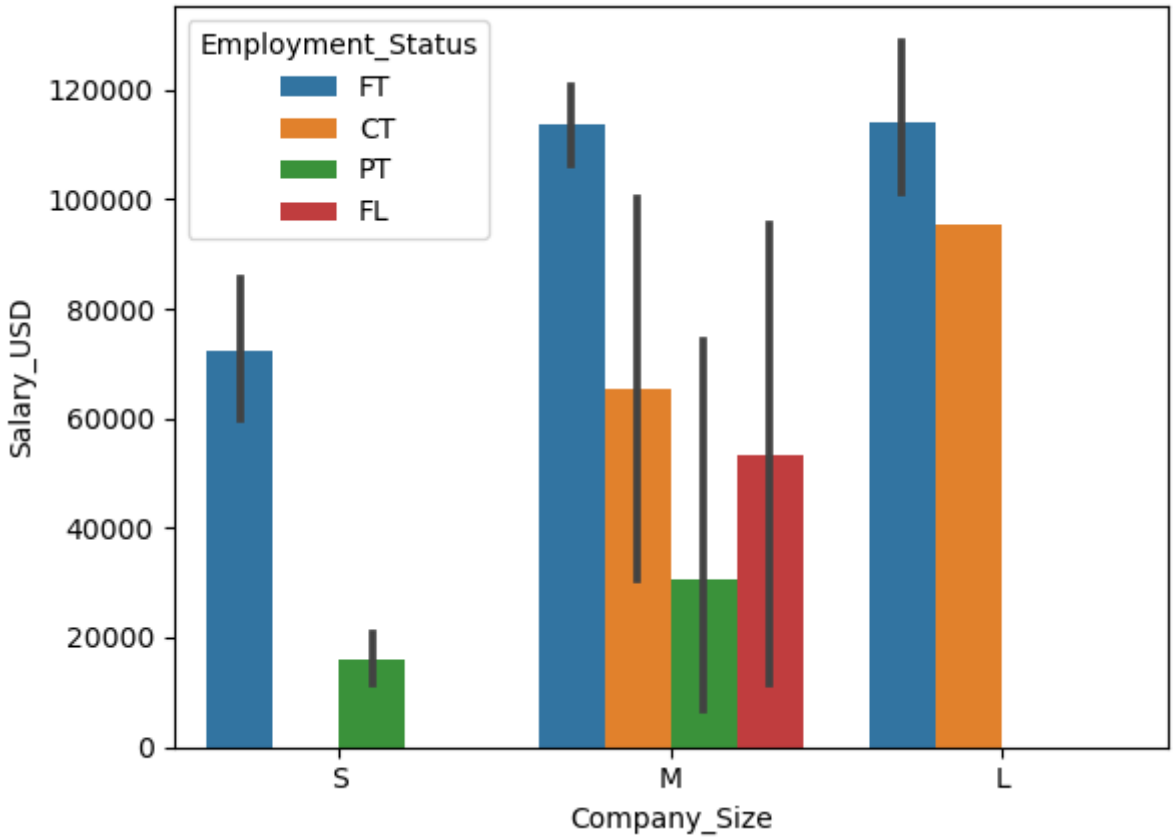
257 rows × 13 columns

In [73]:
```
sns.barplot(data = usa , x =  "Employee_Location" , y = "Salary_USD")
plt.show()
```



In [75]:
```
sns.barplot(data = salaries , y = "Salary_USD" , x=  "Company_Size" , hue =
```

Out[75]:
```
<AxesSubplot:xlabel='Company_Size', ylabel='Salary_USD'>
```



The correct hypothesis is a. On average, small companies pay part-time employees less than large companies.

The plot shows that the salaries for part-time employees (PT) are lower for small companies (S) than for large companies (L). This suggests that there is a negative correlation between company size and salary for part-time employees. We can therefore hypothesize that, on average, small companies pay part-time employees less than large companies.

The other hypotheses are not supported by the data. For example, the plot does not show any difference in salary between freelancers (FL) at small and large companies. Additionally, the plot shows that the salaries for contractors (CT) are similar for medium-sized (M) and large companies.

Therefore, the correct hypothesis is a. On average, small companies pay part-time employees less than large companies. The answer is (a).

In [ ]: