Python [conda env:DataScience]

Professor evaluations and beauty.

Openintro - Professor evaluations and beauty: Teaching ratings

Evaluation of Teacher Productivity and Beauty.

- Beauty in the classroom: teachers' neatness and supposed pedagogical productivity.

Introduction

Data were obtained from end-of-semester evaluations of students from 463 courses taught by a sample of 94 professors at the University of Texas at Austin. In addition, six students rated the physical appearance of the professors. The result is a data frame where each row contains a different course and each column has information about the course and the professor who taught it.

Data Description

Variable	Description
minority	Is the instructor a member of a minority group (non-Caucasian)?
age	The teacher's age
gender	Indicates whether the instructor was male or female.
credits	Is the course a single credit elective?
beauty	Rating of instructor physical appearance by a panel of six students, averaged across the six panelists, and standardized to have a mean of zero.
eval	Overall score of the teaching evaluation of the course, on a scale of 1 (very unsatisfactory) to 5 (excellent).
division	Is the course upper or lower division?
native	Is the instructor a native English speaker?
tenure	Does the instructor have the possibility of permanence?
students	Number of students who participated in the assessment.
allstudents	Number of students enrolled in the course.
prof	Indicating instructor ID.

- **Necessary packages and libraries:**

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

- **Read data from .csv file with pandas**

In [2]:
```
teaching_ratings = 'teachingratings.csv'
data = pd.read_csv(teaching_ratings)
```

- # Display dataset information

- **First five rows of data.**

In [3]:
```
data.head()
```

Out[3]:

	minority	age	gender	credits	beauty	eval	division	native	tenure	students
0	yes	36	female	more	0.289916	4.3	upper	yes	yes	24
1	yes	36	female	more	0.289916	3.7	upper	yes	yes	86
2	yes	36	female	more	0.289916	3.6	upper	yes	yes	76
3	yes	36	female	more	0.289916	4.4	upper	yes	yes	77
4	no	59	male	more	-0.737732	4.5	upper	yes	yes	17

- **Get each variable information**

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   minority         463 non-null    object 
 1   age              463 non-null    int64  
 2   gender           463 non-null    object 
 3   credits          463 non-null    object 
 4   beauty           463 non-null    float64
 5   eval             463 non-null    float64
 6   division         463 non-null    object 
 7   native           463 non-null    object 
 8   tenure           463 non-null    object 
 9   students         463 non-null    int64  
 10  allstudents      463 non-null    int64  
 11  prof             463 non-null    int64  
 12  PrimaryLast      463 non-null    int64  
 13  vismin           463 non-null    int64  
 14  female           463 non-null    int64  
 15  single_credit    463 non-null    int64  
 16  upper_division   463 non-null    int64  
 17  English_speaker  463 non-null    int64  
 18  tenured_prof     463 non-null    int64  
dtypes: float64(2), int64(11), object(6)
memory usage: 68.9+ KB
```

- **Get the number of rows and columns of the dataset - (number of rows, number of columns)**

In [5]:
```
data.shape
```

Out[5]: (463, 19)

- **Descriptive statistics table**

In [6]:
```
data.describe()
```

Out[6]:

	age	beauty	eval	students	allstudents	prof	P
count	463.000000	4.630000e+02	463.000000	463.000000	463.000000	463.000000	4
mean	48.365011	6.271140e-08	3.998272	36.624190	55.177106	45.434125	
std	9.802742	7.886477e-01	0.554866	45.018481	75.072800	27.508902	
min	29.000000	-1.450494e+00	2.100000	5.000000	8.000000	1.000000	
25%	42.000000	-6.562689e-01	3.600000	15.000000	19.000000	20.000000	
50%	48.000000	-6.801430e-02	4.000000	23.000000	29.000000	44.000000	
75%	57.000000	5.456024e-01	4.400000	40.000000	60.000000	70.500000	
max	73.000000	1.970023e+00	5.000000	380.000000	581.000000	94.000000	

- Unique values in the "prof" variable

In [7]:
```
data.prof.unique()
```

Out[7]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 1
7,
       18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35, 3
6,
       37, 38, 39, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 5
5,
       56, 57, 58, 59, 60, 63, 64, 65, 66, 67, 68, 70, 71, 72, 73, 74, 7
5,
       76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 9
2,
       93, 94, 22, 30, 40, 47, 61, 62, 69])
```

- Number of unique values in the "prof" variable

In [8]:
```
data.prof.nunique()
```

Out[8]:
```
94
```

- Drop duplicates using "prof" as a subset and assign it a new DataFrame called "not_duplicates"

In [9]:
```
not_duplicates = data.drop_duplicates(subset =['prof'])
not_duplicates.shape
```

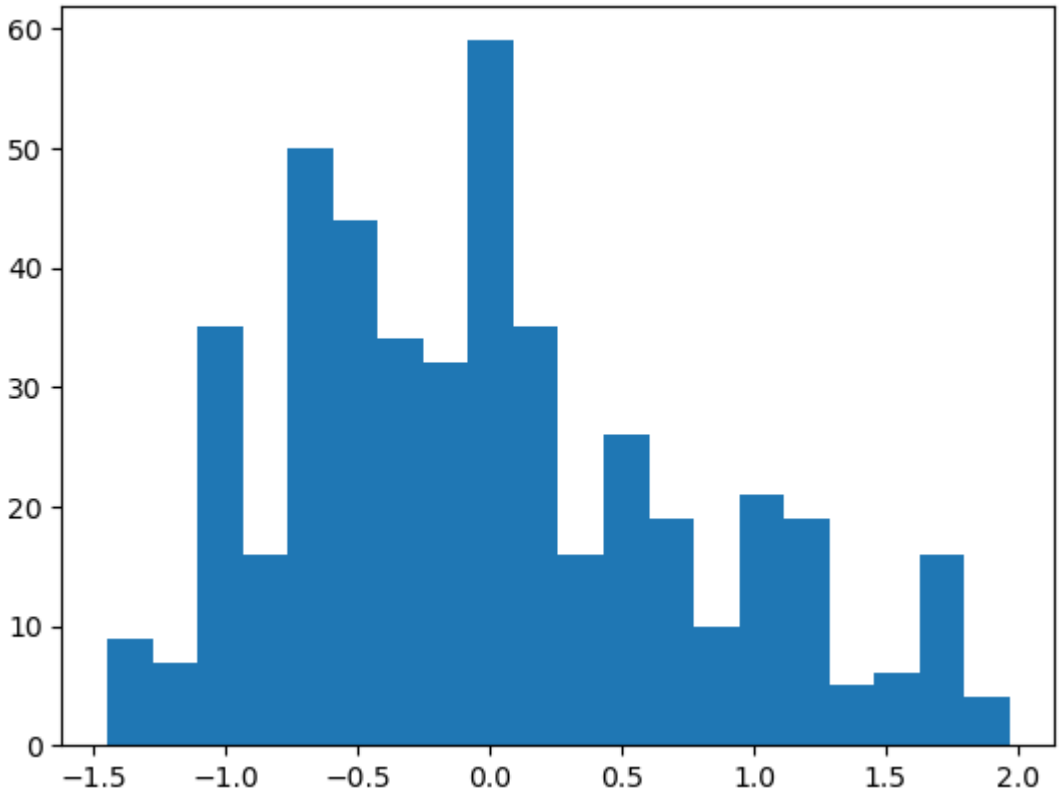
Out[9]:
```
(94, 19)
```

Data Visualization:

- **Beatuty variable distribution data: "beauty".**

In [10]:
```
plt.hist(data['beauty'], bins=20)
```

Out[10]:
```
(array([ 9.,  7., 35., 16., 50., 44., 34., 32., 59., 35., 16., 26., 19.,
        10., 21., 19.,  5.,  6., 16.,  4.]),
 array([-1.45049405, -1.2794682 , -1.10844234, -0.93741649, -0.76639063,
        -0.59536478, -0.42433892, -0.25331307, -0.08228722,  0.08873864,
         0.25976449,  0.43079035,  0.6018162 ,  0.77284206,  0.94386791,
         1.11489376,  1.28591962,  1.45694547,  1.62797133,  1.79899718,
         1.97002304]),
 <BarContainer object of 20 artists>)
```



In [11]:
```
data['beauty'].mean()
```

Out[11]:
```
6.271139975345787e-08
```

- **Mean, standard deviation and variance of beauty respect to gender.**

In [12]:
```
data.groupby('gender').agg({'beauty':['mean', 'std', 'var']}).reset_index
```

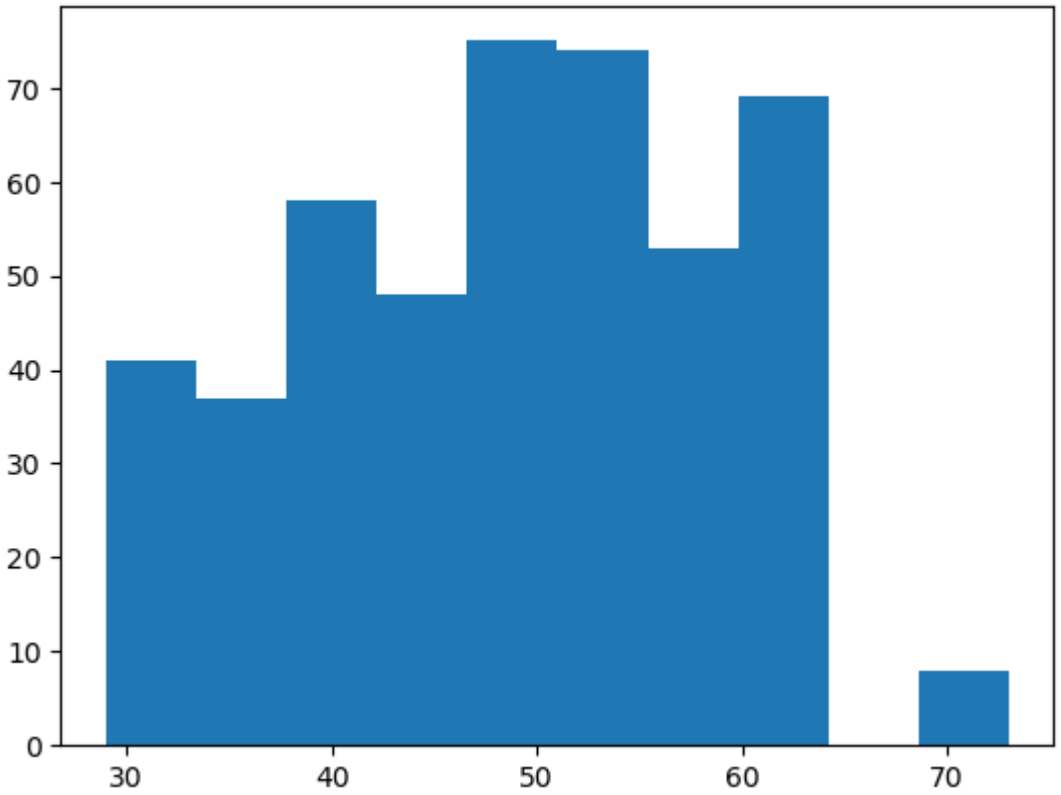
Out[12]:

	gender	beauty		
		mean	std	var
0	female	0.116109	0.81781	0.668813
1	male	-0.084482	0.75713	0.573246

- **Display age variable: "age"**

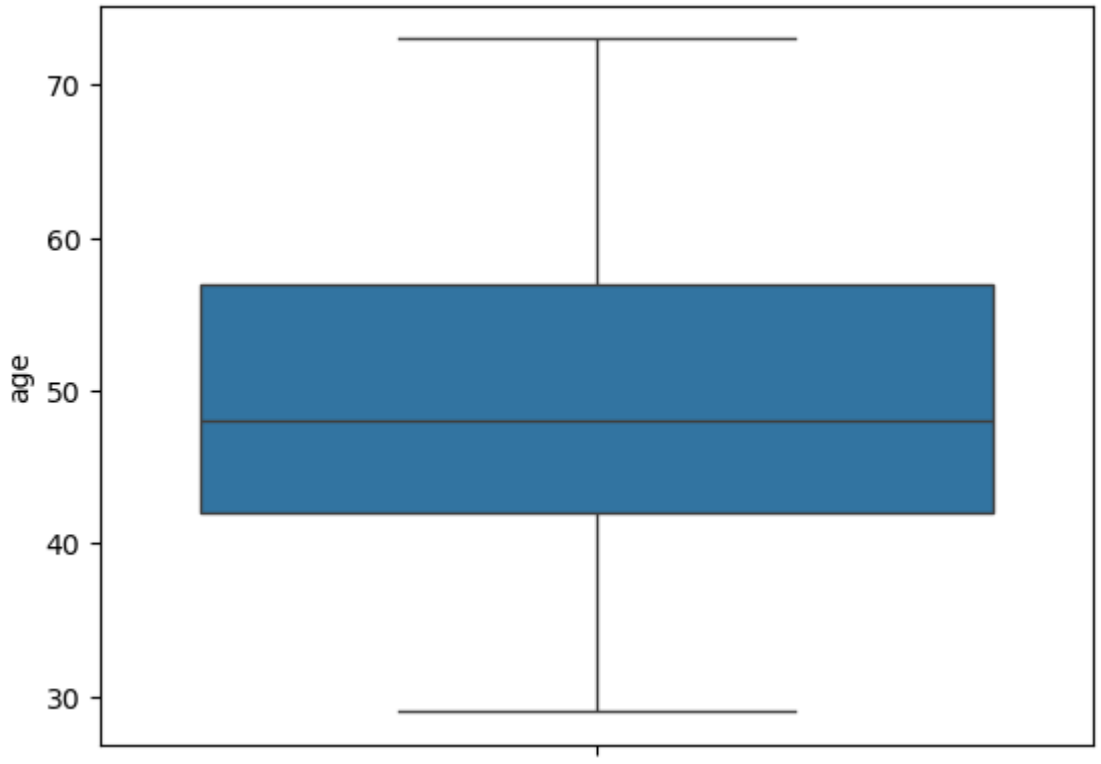
In [13]:
```
plt.hist(data['age'])
```

Out[13]:
```
(array([41., 37., 58., 48., 75., 74., 53., 69.,  0.,  8.]),
 array([29. , 33.4, 37.8, 42.2, 46.6, 51. , 55.4, 59.8, 64.2, 68.6, 73.
]),
 <BarContainer object of 10 artists>)
```



- **Age variable Box plot.**

In [14]:
```
ax = sns.boxplot(y="age", data=data)
```



- **Mean age and standard deviation of teachers in the dataset.**

In [15]:
```
data.agg({'age':['mean', 'std']}).reset_index()
```

Out[15]:

	index	age
0	mean	48.365011
1	std	9.802742

- **Mean age and standard deviation of teachers in the dataset by gender.**

In [16]:
```
data.groupby('gender').agg({'age':['mean', 'std']}).reset_index()
```

Out[16]:

	gender	age	
		mean	std
0	female	45.092308	8.532031
1	male	50.746269	9.993396

- **Use the new dataset without duplicates to get the actual average age and actual standard deviation.**

In [17]:
```
not_duplicates.agg({'age':['mean', 'std']}).reset_index()
```

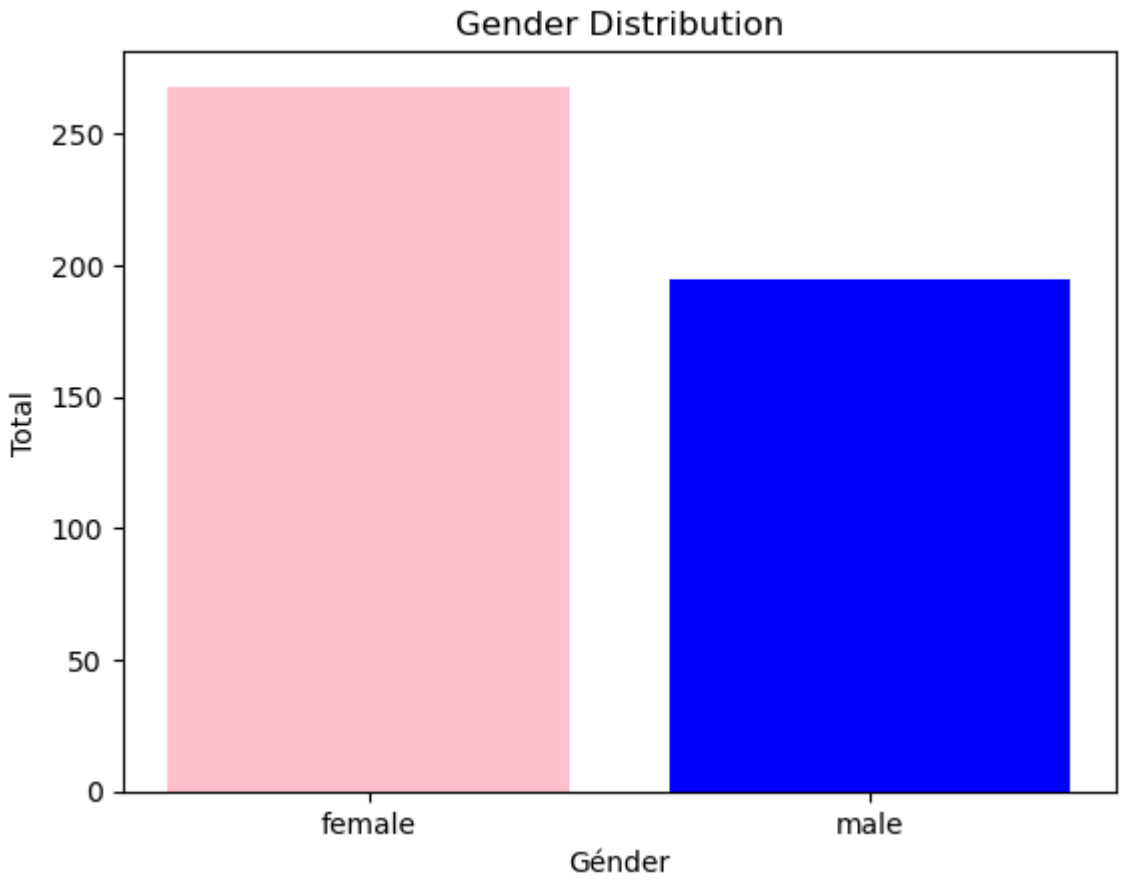
Out[17]:

	index	age
0	mean	47.553191
1	std	10.256513

- **Display the gender variable: "gender"**

In [18]:
```
plt.bar(data.gender.unique(),data.gender.value_counts(),color=['pink','bl
plt.xlabel('Génder')
plt.ylabel('Total')
plt.title('Gender Distribution')
```

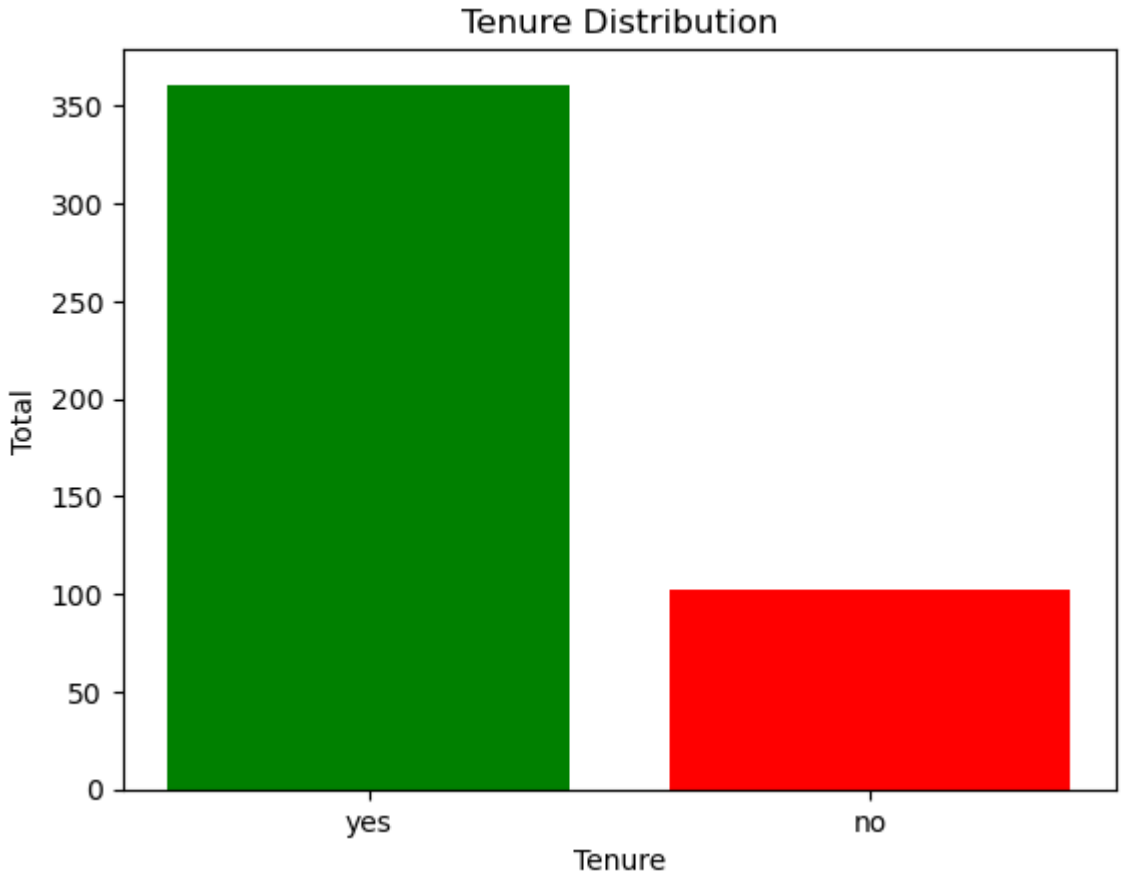
Out[18]:
```
Text(0.5, 1.0, 'Gender Distribution')
```



- **Display the tenure variable: "tenure"**

In [19]:
```
plt.bar(data.tenure.unique(),data.tenure.value_counts(),color=['green','r
plt.xlabel('Tenure')
plt.ylabel('Total')
plt.title('Tenure Distribution')
```

Out[19]:
```
Text(0.5, 1.0, 'Tenure Distribution')
```



- **Percentage of tenured professors by gender**

In [20]:
```
tenure_count = data[data.tenure == 'yes'].groupby('gender').agg({'tenure'
tenure_count
```

Out[20]:

	gender	tenure
0	female	145
1	male	216

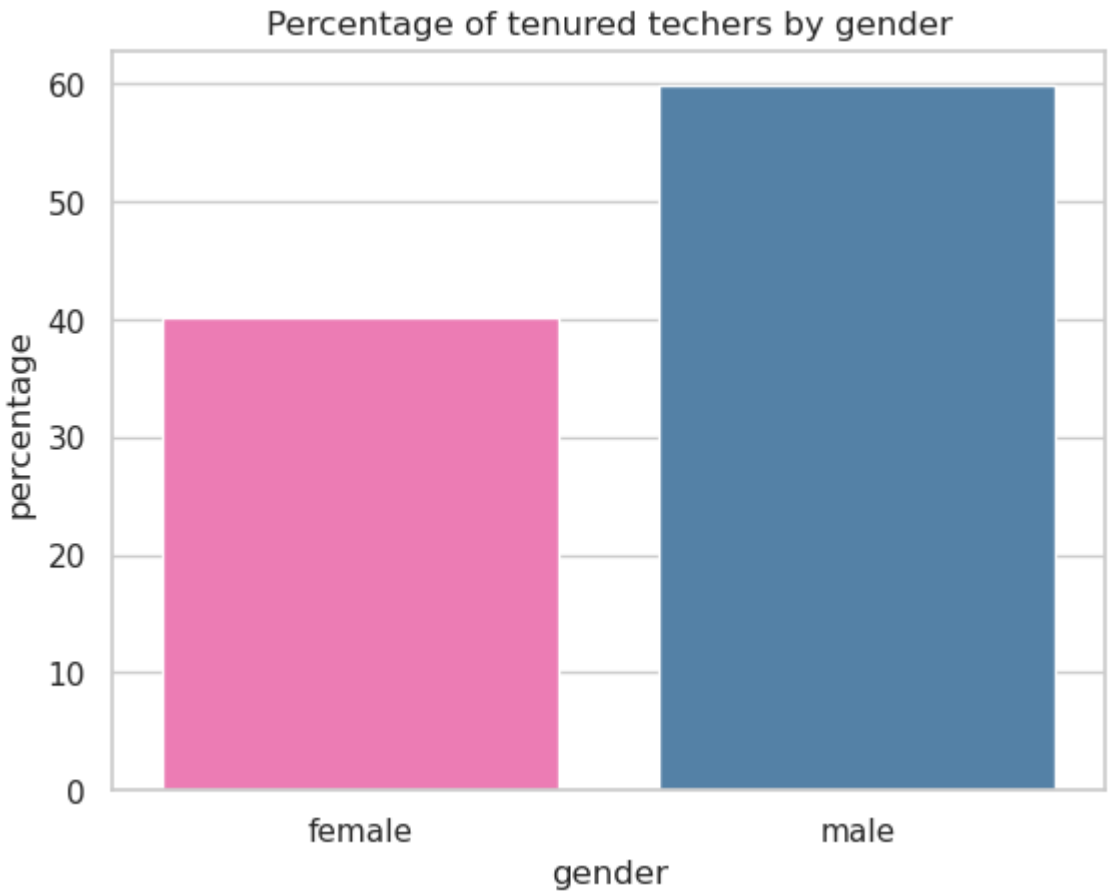
In [21]:
```
tenure_count['percentage'] = 100 * tenure_count.tenure/tenure_count.tenur
tenure_count
```

Out[21]:

	gender	tenure	percentage
0	female	145	40.166205
1	male	216	59.833795

In [22]:
```
sns.set(style='whitegrid')
ax = sns.barplot(x='gender', y='percentage', data=tenure_count, hue='gend
ax.set_title('Percentage of tenured techers by gender')
```

Out[22]:
```
Text(0.5, 1.0, 'Percentage of tenured techers by gender')
```



- **Percentage of visible minorities who are tenured professors.**

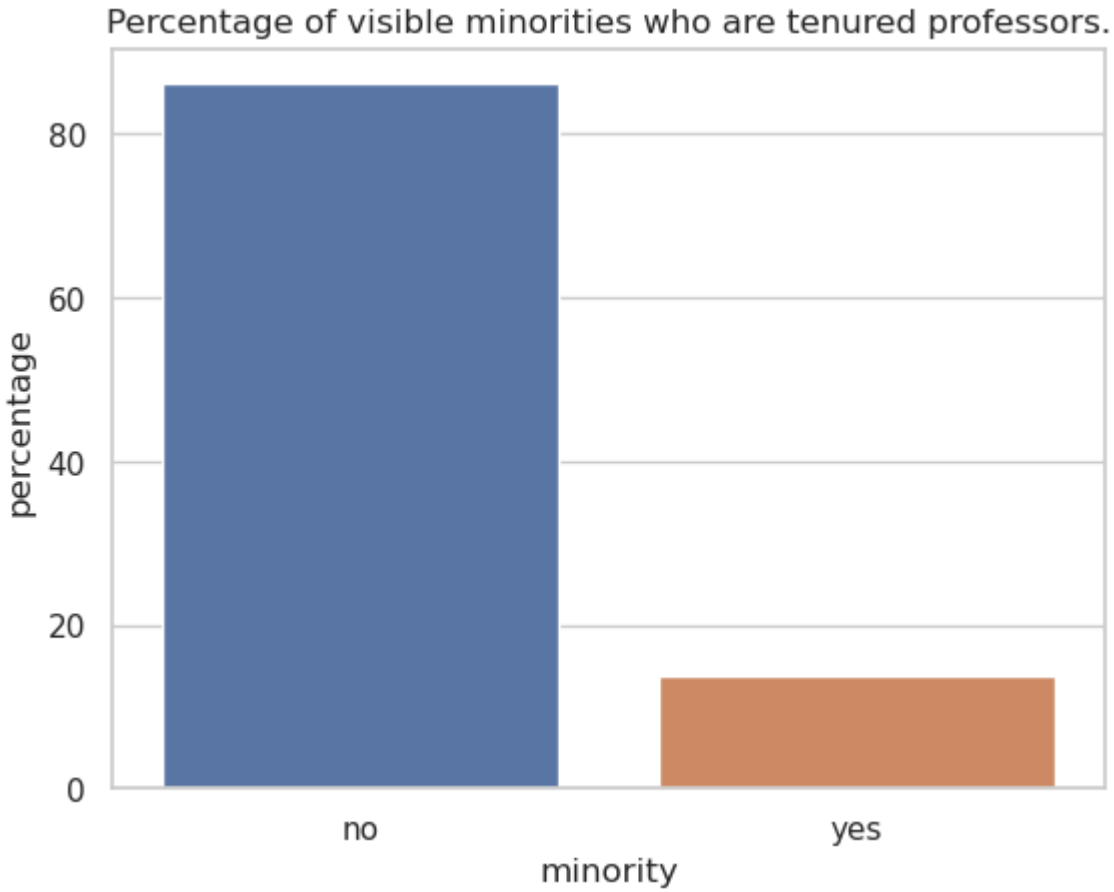
In [23]:
```
tenure_min = data.groupby('minority').agg({'tenure': 'count'}).reset_inde
tenure_min['percentage'] = 100 * tenure_min.tenure/tenure_min.tenure.sum(
tenure_min
```

Out[23]:

	minority	tenure	percentage
0	no	399	86.177106
1	yes	64	13.822894

In [24]:
```
sns.set(style='whitegrid')
ax = sns.barplot(x='minority', y='percentage', data=tenure_min, hue='mino
ax.set_title('Percentage of visible minorities who are tenured professors
```

Out[24]:
```
Text(0.5, 1.0, 'Percentage of visible minorities who are tenured profess
ors.')
```



- **Average evaluation score of the tenured professors**

In [25]:
```
data[data['tenure'] == 'yes']['eval'].median()
```

Out[25]: 4.0

- **Average teacher evaluation in both upper and lower division groups**

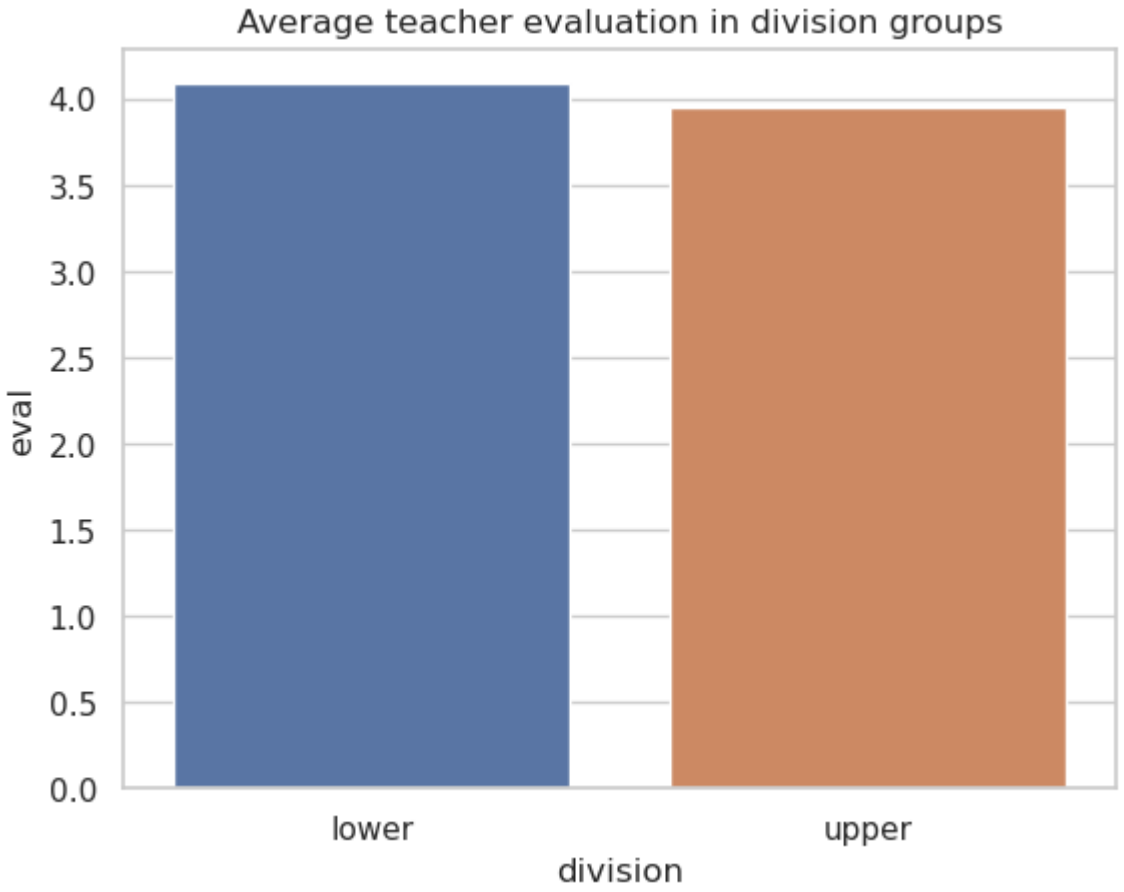
In [26]:
```
division_eval = data.groupby('division')[['eval']].mean().reset_index()
division_eval
```

Out[26]:

	division	eval
0	lower	4.087261
1	upper	3.952614

In [27]:
```
sns.set(style='whitegrid')
ax = sns.barplot(x='division', y='eval', data=division_eval, hue='divisio
ax.set_title('Average teacher evaluation in division groups')
```

Out[27]: Text(0.5, 1.0, 'Average teacher evaluation in division groups')



- **Relationship between age and teacher evaluation scores.**

In [28]:
```
ax = sns.scatterplot(x='age', y='eval', data=data)
ax.set_title('Relation: Age and average score')
```

Out[28]:
```
Text(0.5, 1.0, 'Relation: Age and average score')
```

- **Relationship between age and teacher evaluation scores, broken down by gender.**

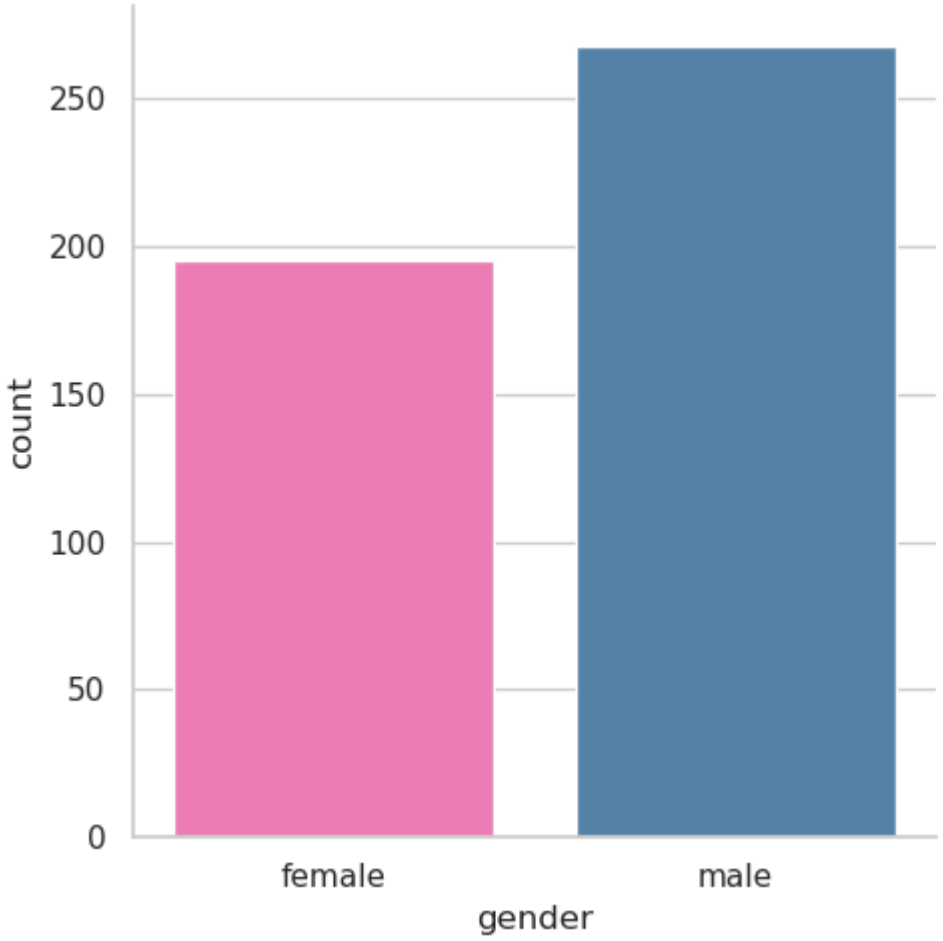
In [29]:
```
ax = sns.scatterplot(x='age', y='eval', hue='gender', data=data, palette=
ax.set_title('Relation: Age and average score by gender')
```

Out[29]:
```
Text(0.5, 1.0, 'Relation: Age and average score by gender')
```

- **Courses taught by gender**

In [30]:
```
sns.catplot(x='gender', kind='count', data=data, hue='gender', palette=['
```

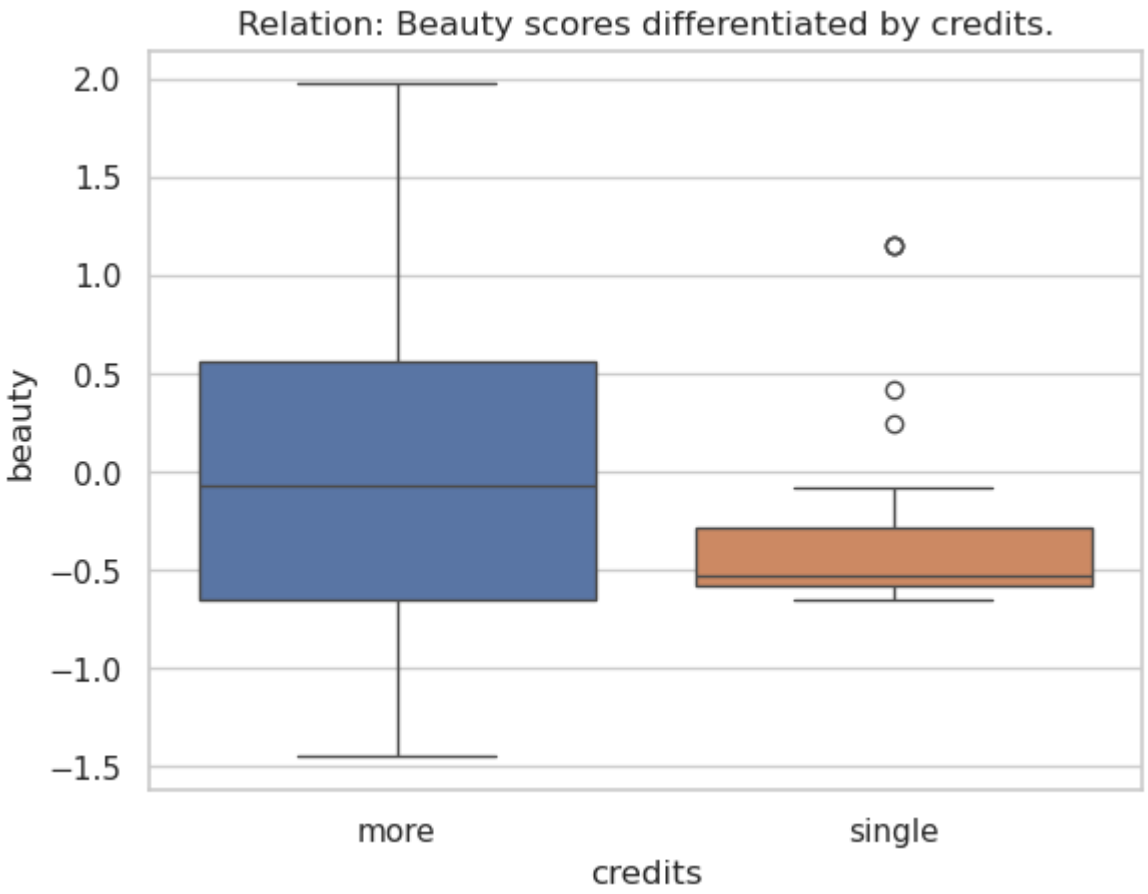
Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x7f3b5c8dae90>
```



- **Beauty scores differentiated by credits Box plot**

In [31]:
```
ax = sns.boxplot(x='credits', y='beauty', data=data, hue='credits')
ax.set_title('Relation: Beauty scores differentiated by credits.')
```

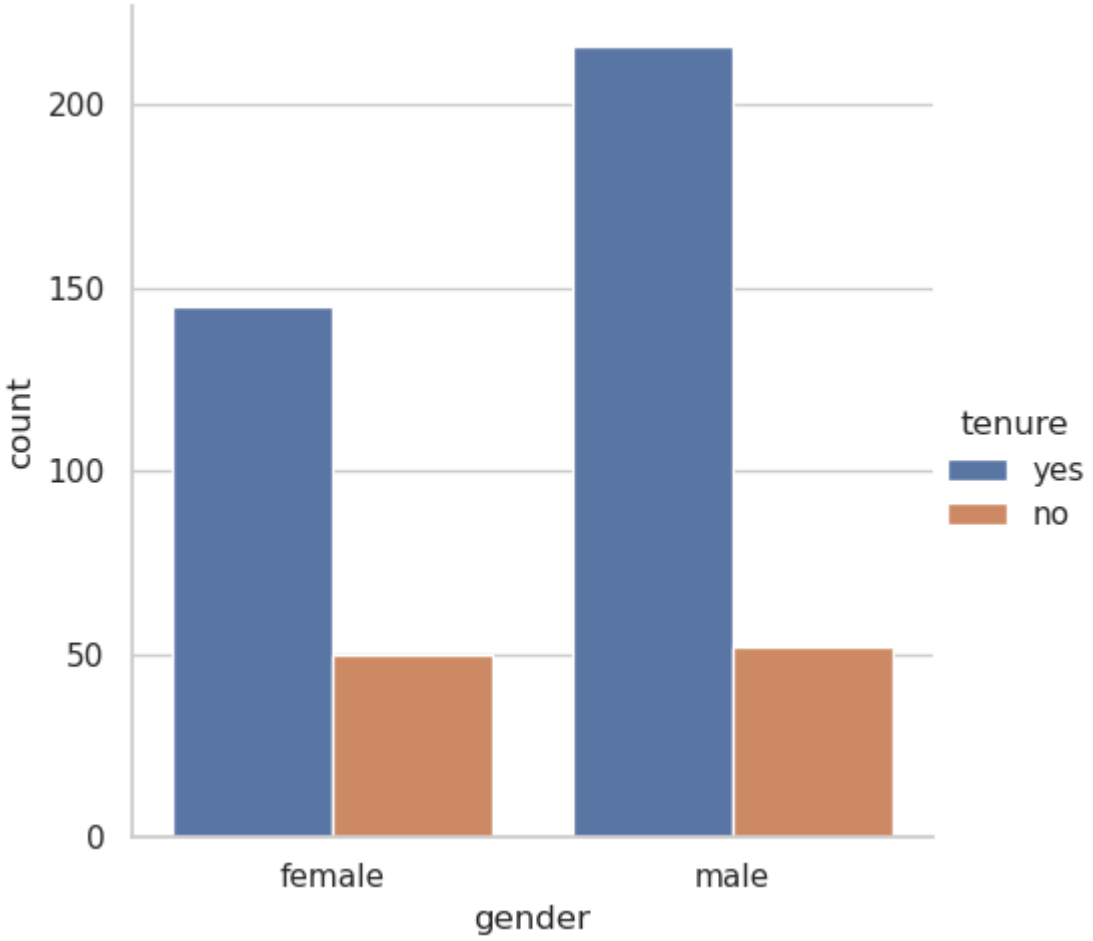
Out[31]:
```
Text(0.5, 1.0, 'Relation: Beauty scores differentiated by credits.')
```



- **Histogram group of teachers by gender and seniority**

In [32]:
```
sns.catplot(x='gender', hue='tenure', kind='count', data=data)
```

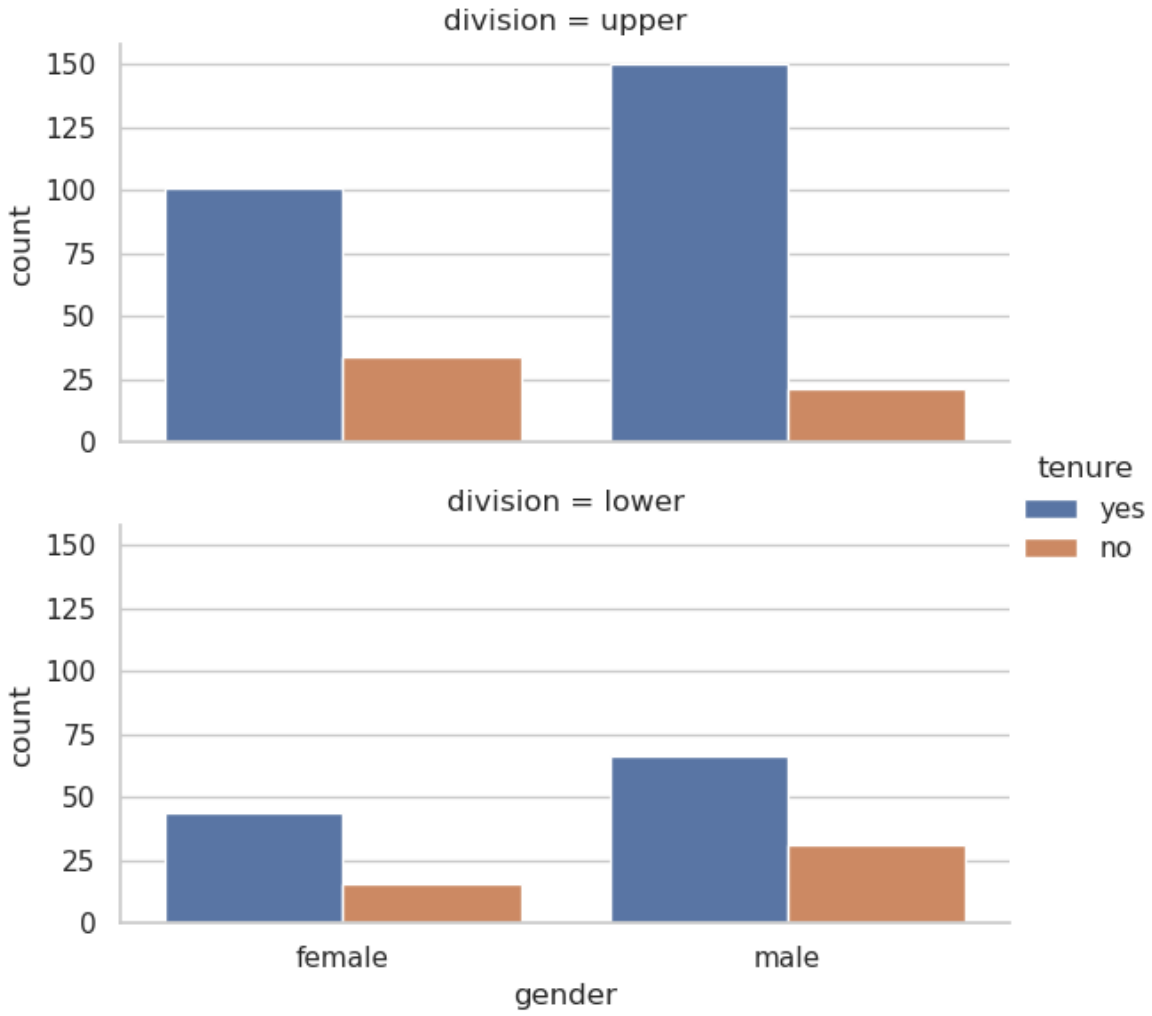
Out[32]:
```
<seaborn.axisgrid.FacetGrid at 0x7f3b5c909490>
```



- **Add division as another factor to the previous histogram**

In [33]:
```
sns.catplot(x='gender', hue = 'tenure', row = 'division',
            kind='count', data=data, height = 3, aspect = 2)
```

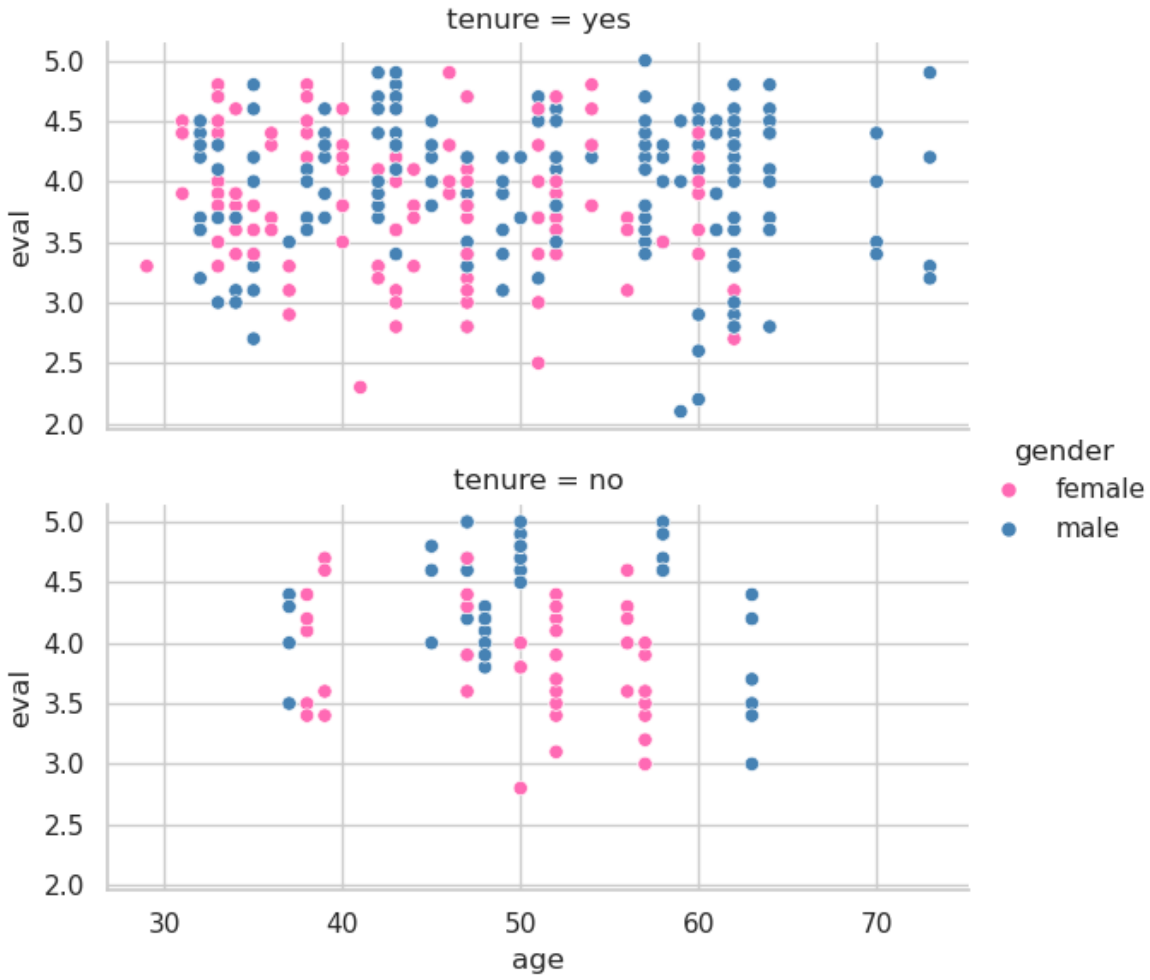
Out[33]:
```
<seaborn.axisgrid.FacetGrid at 0x7f3b56ff6d50>
```



- **Scatterplot of age and assessment scores, broken down by gender and tenure.**

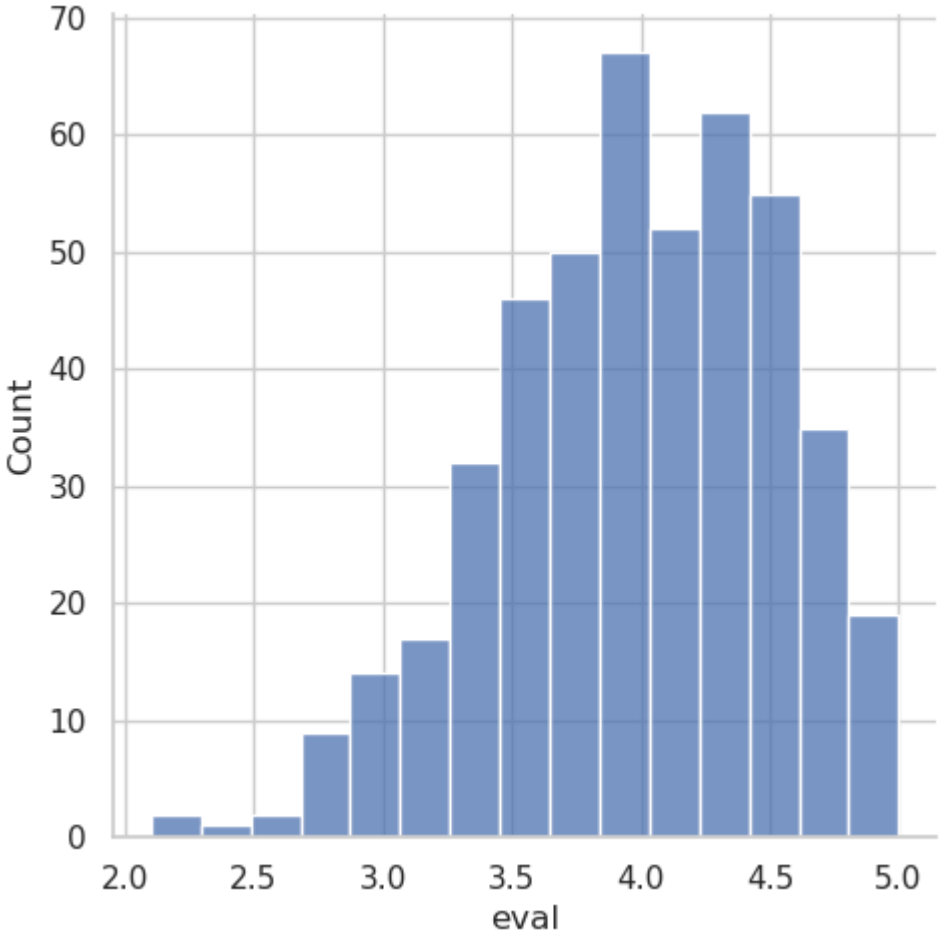
In [34]:
```
sns.relplot(x="age", y="eval", hue="gender", row="tenure", palette=['#FF6
            data=data, height=3, aspect=2)
```

Out[34]:
```
<seaborn.axisgrid.FacetGrid at 0x7f3b56e1abd0>
```



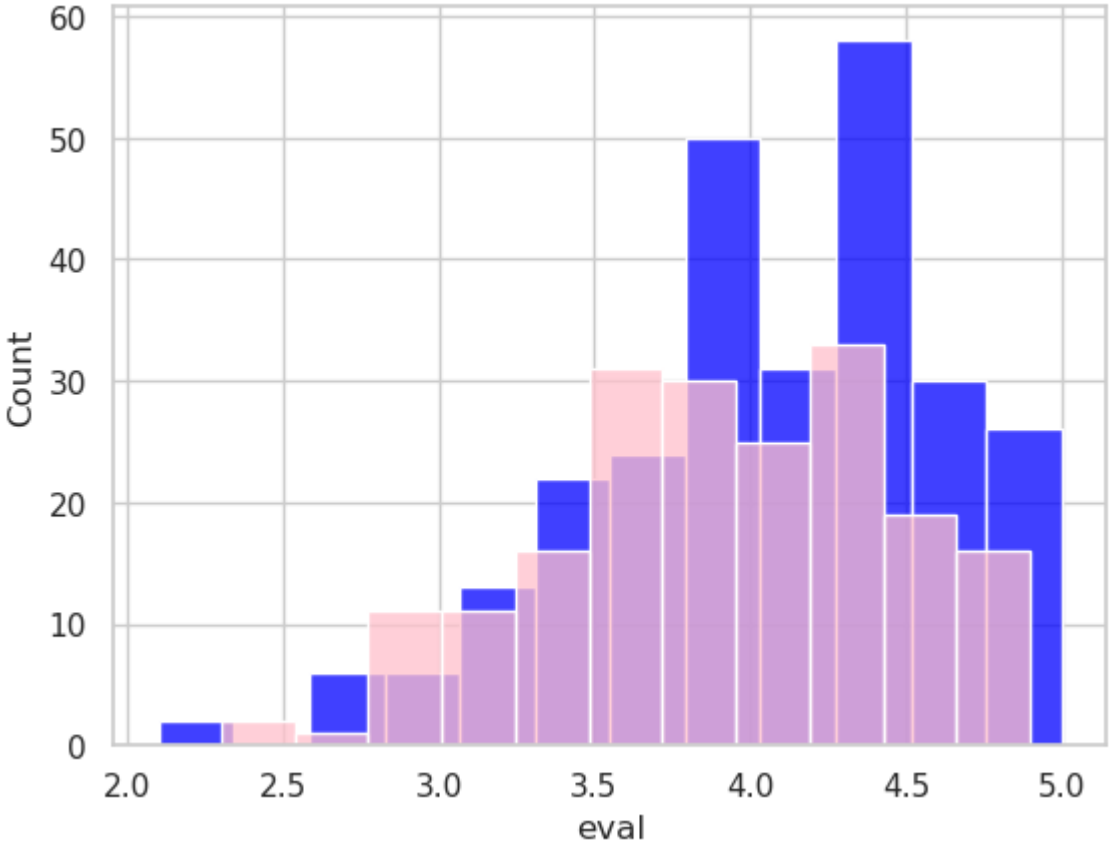
- **Teacher evaluation score distribution chart**

In [35]:
```
ax = sns.displot(data['eval'], kde = False)
```



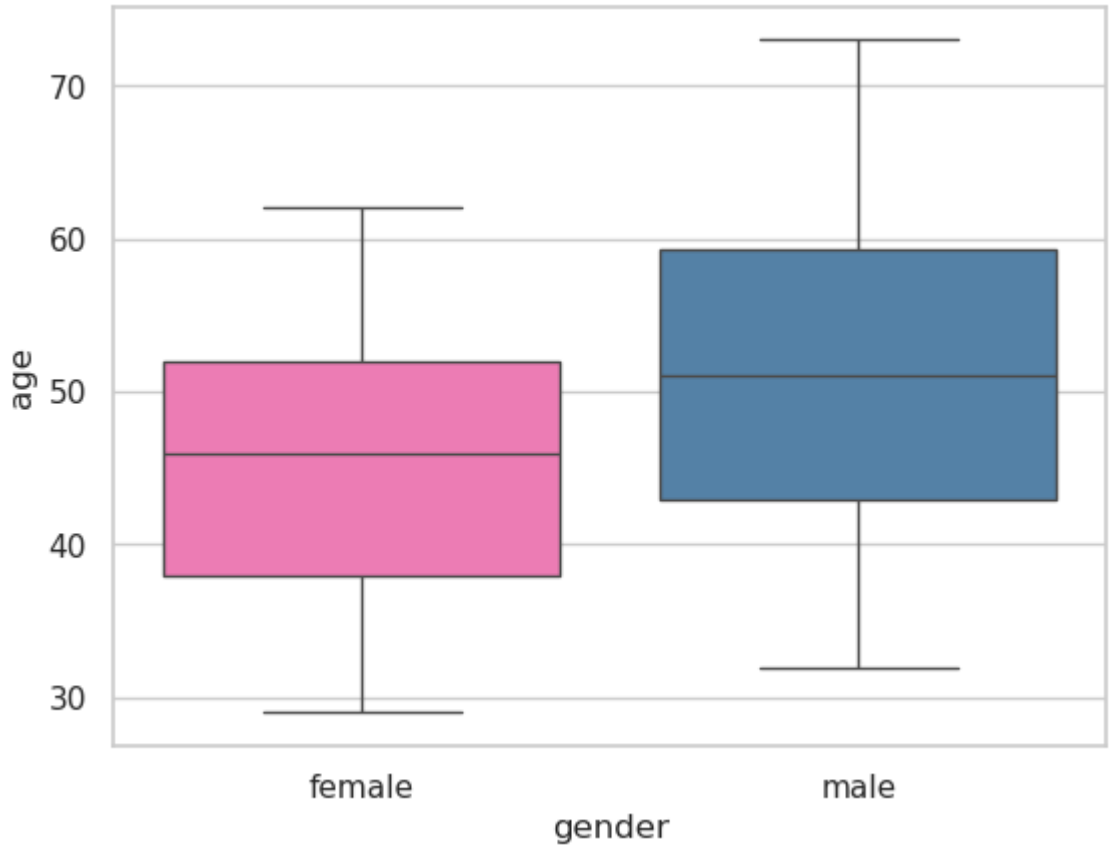
- **Teacher evaluation score distribution graph with gender as a factor**

In [36]:
```
sns.histplot(data[data['gender'] == 'male']['eval'], color="blue", kde=Fa
sns.histplot(data[data['gender'] == 'female']['eval'], color='pink', kde=
plt.show()
```



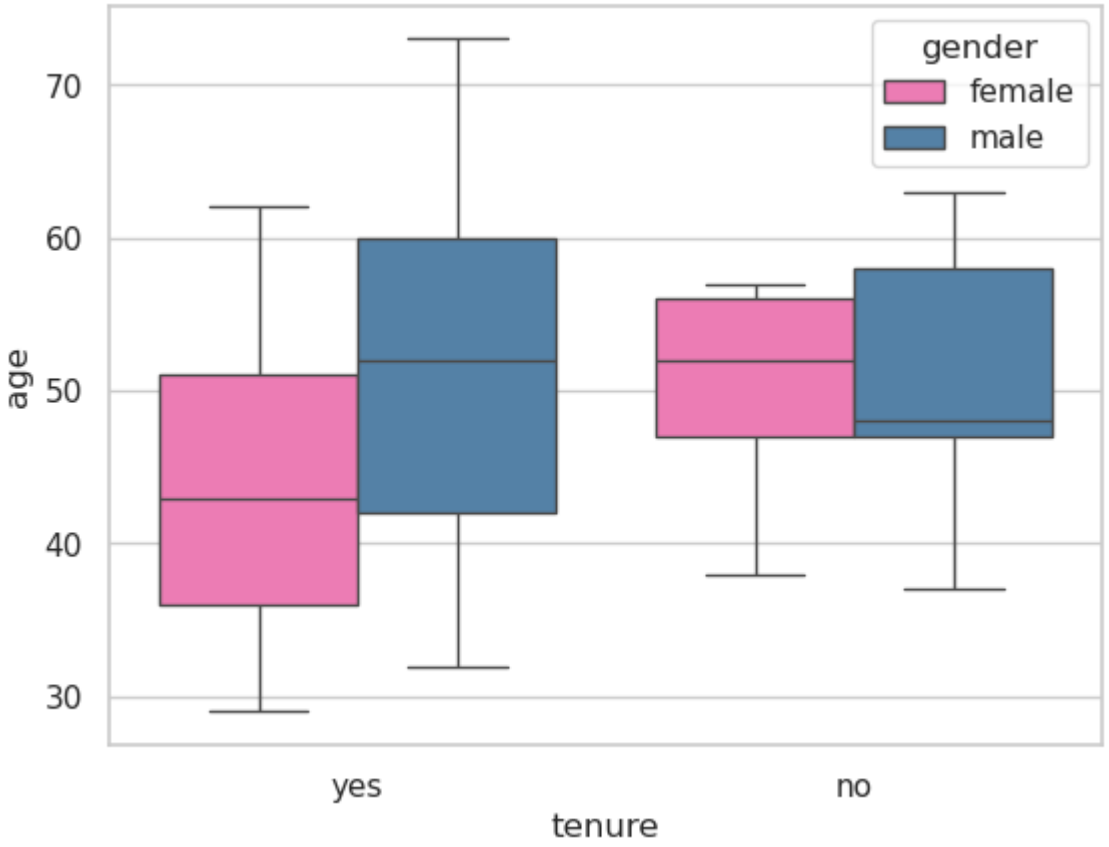
- **Box plot: teacher age by gender**

```
In [37]: ax = sns.boxplot(x="gender", y="age", data=data, hue='gender', palette=['
```



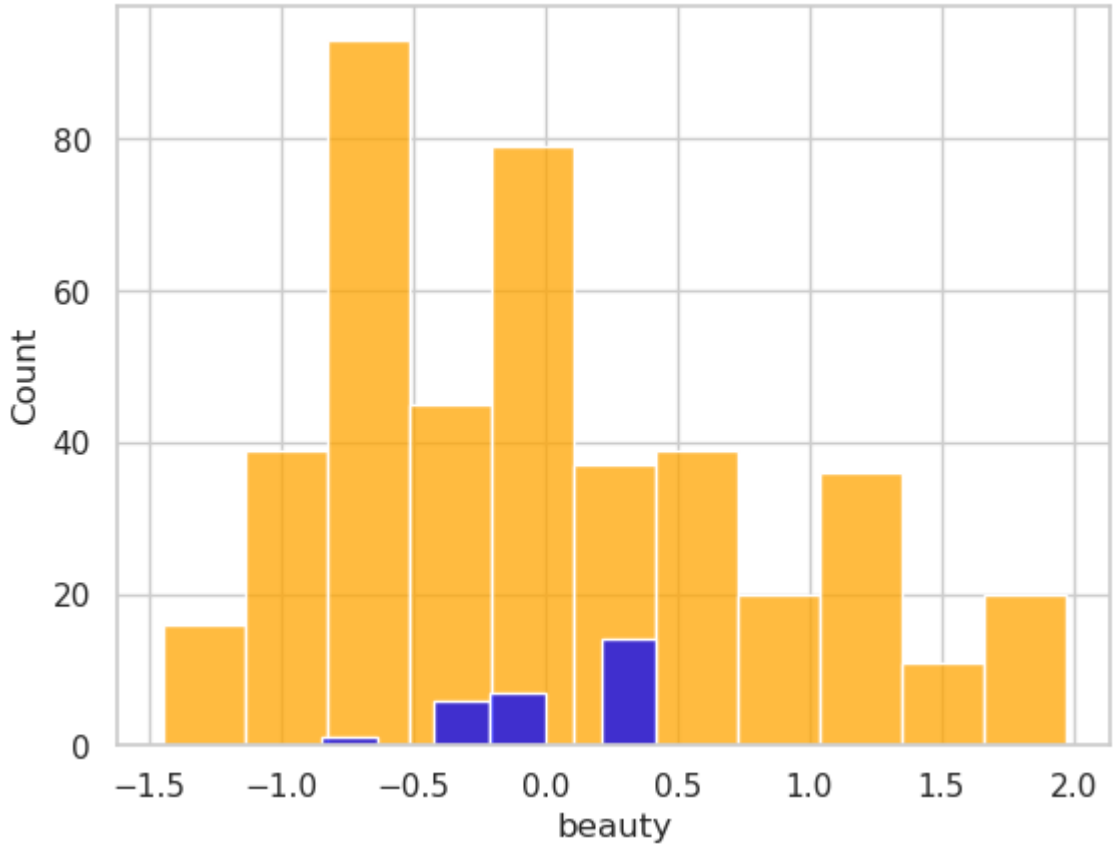
- **Compare age with tenure and gender**

In [38]:
```
ax = sns.boxplot(x="tenure", y="age", hue="gender", data=data, palette=['
```



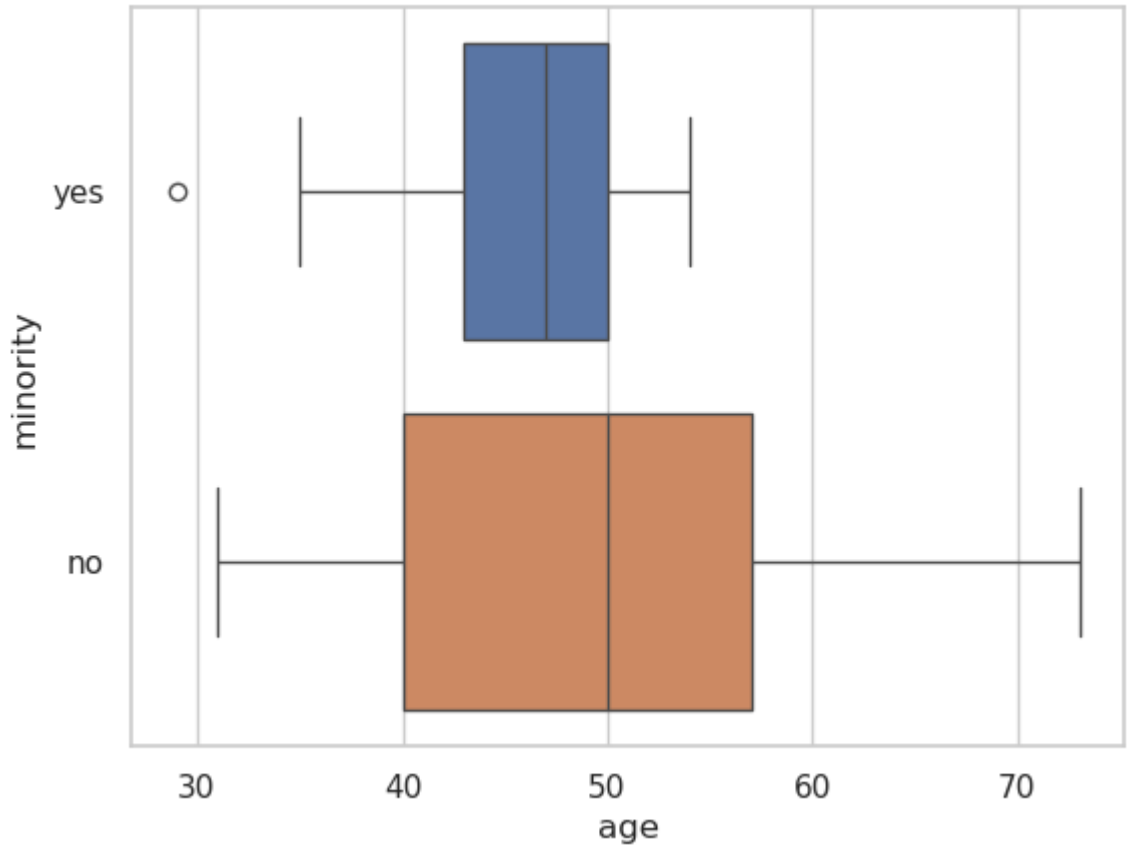
- **Beauty score distribution graph with native English speaker as factor**

In [39]:
```
sns.histplot(data[data['native'] == 'yes']['beauty'], color="orange", kde
sns.histplot(data[data['native'] == 'no']['beauty'], color="blue", kde=Fa
plt.show()
```



- **Horizontal box plot of instructors' age by visible minority**

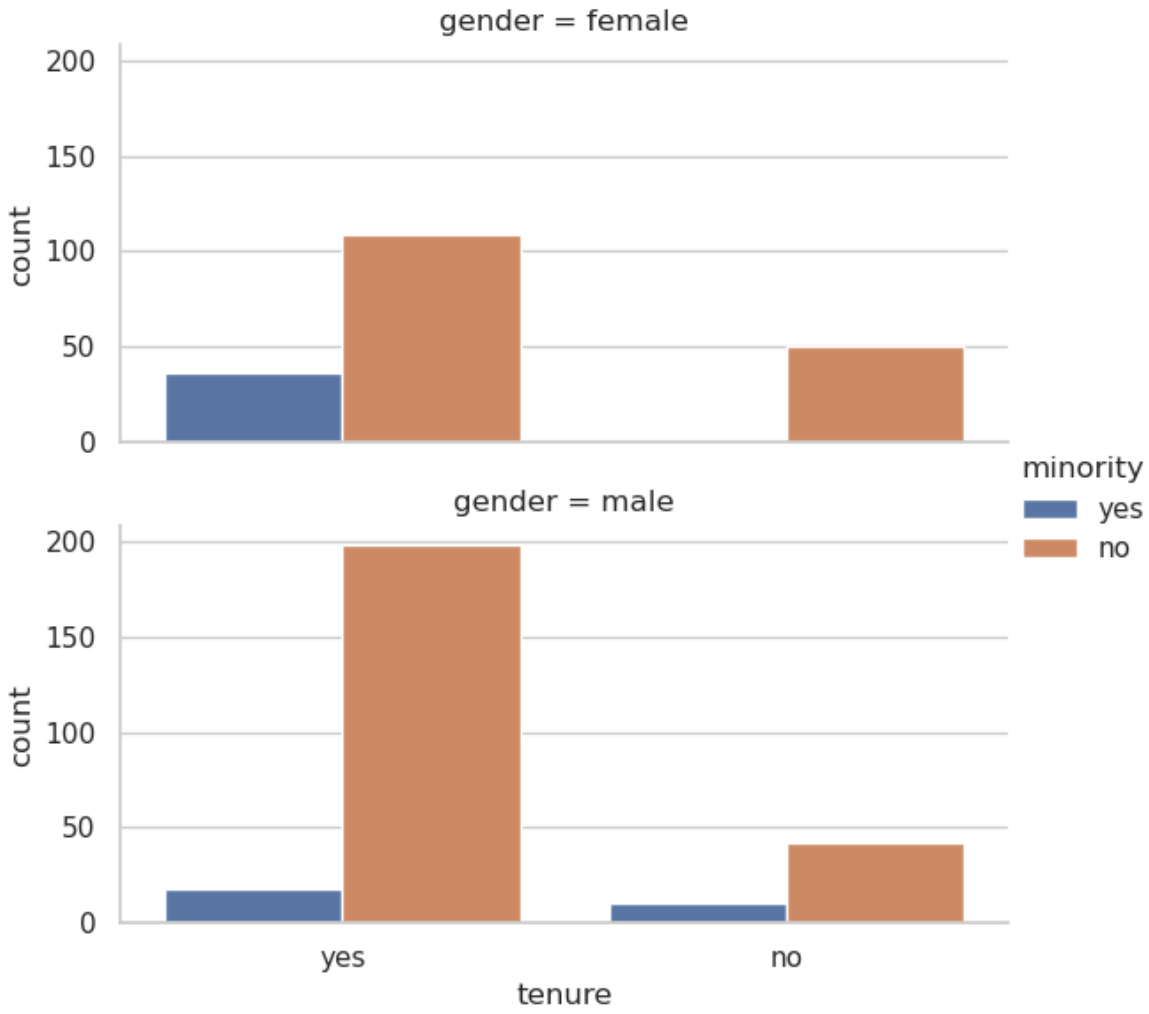
In [40]:
```
ax = sns.boxplot(x="age", y="minority", data=data, hue='minority')
```



- **Group histogram of minority tenure and gender factor**

In [41]:
```
sns.catplot(x='tenure', hue='minority', row='gender',
            kind='count', data=data, height = 3, aspect = 2)
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x7f3b567f4250>



Changelog:

Date (DD/MM/YYYY)	Version	Description of change
15/03/2024	00.0	Download and process
18/03/2024	01.0	Added catplots
19/03/2024	02.0	Bug fixes