Importing the required libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
import seaborn as sns
```

Reading the data set

In [2]:

```
df=pd.read_csv('emp_promotion.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1
1	65141	Operations	region_22	Bachelor's	m	other	1
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2
4	48945	Technology	region_26	Bachelor's	m	other	1

Removing unwanted coloumns

In [4]:

```
df.drop(['employee_id'],axis=1,inplace=True)
```

Replacing null values

In [5]:

```
df.isnull().any()
```

Out[5]:

```
department            False
region                False
education              True
gender                False
recruitment_channel   False
no_of_trainings       False
age                   False
previous_year_rating   True
length_of_service     False
KPIs_met >80%         False
awards_won?           False
avg_training_score    False
is_promoted           False
dtype: bool
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   department            54808 non-null  object 
 1   region                54808 non-null  object 
 2   education             52399 non-null  object 
 3   gender                54808 non-null  object 
 4   recruitment_channel   54808 non-null  object 
 5   no_of_trainings       54808 non-null  int64  
 6   age                   54808 non-null  int64  
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64  
 9   KPIs_met >80%         54808 non-null  int64  
 10  awards_won?           54808 non-null  int64  
 11  avg_training_score    54808 non-null  int64  
 12  is_promoted           54808 non-null  int64  
dtypes: float64(1), int64(7), object(5)
memory usage: 5.4+ MB
```

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
department                 0
region                     0
education               2409
gender                     0
recruitment_channel        0
no_of_trainings            0
age                        0
previous_year_rating    4124
length_of_service          0
KPIs_met >80%              0
awards_won?                0
avg_training_score         0
is_promoted                0
dtype: int64
```

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
118
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   department            54808 non-null  object 
 1   region                54808 non-null  object 
 2   education             52399 non-null  object 
 3   gender                54808 non-null  object 
 4   recruitment_channel   54808 non-null  object 
 5   no_of_trainings       54808 non-null  int64  
 6   age                   54808 non-null  int64  
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64  
 9   KPIs_met >80%         54808 non-null  int64  
 10  awards_won?           54808 non-null  int64  
 11  avg_training_score    54808 non-null  int64  
 12  is_promoted           54808 non-null  int64  
dtypes: float64(1), int64(7), object(5)
memory usage: 5.4+ MB
```

In [10]:

```
df['education'].unique()
```

Out[10]:

```
array(["Master's & above", "Bachelor's", nan, 'Below Secondary'],
      dtype=object)
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   department            54808 non-null  object 
 1   region                54808 non-null  object 
 2   education             52399 non-null  object 
 3   gender                54808 non-null  object 
 4   recruitment_channel   54808 non-null  object 
 5   no_of_trainings       54808 non-null  int64  
 6   age                   54808 non-null  int64  
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64  
 9   KPIs_met >80%         54808 non-null  int64  
 10  awards_won?           54808 non-null  int64  
 11  avg_training_score    54808 non-null  int64  
 12  is_promoted           54808 non-null  int64  
dtypes: float64(1), int64(7), object(5)
memory usage: 5.4+ MB
```

In [12]:

```
df.describe()
```

Out[12]:

	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	aw
count	54808.000000	54808.000000	50684.000000	54808.000000	54808.000000	54
mean	1.253011	34.803915	3.329256	5.865512	0.351974	
std	0.609264	7.660169	1.259993	4.265094	0.477590	
min	1.000000	20.000000	1.000000	1.000000	0.000000	
25%	1.000000	29.000000	3.000000	3.000000	0.000000	
50%	1.000000	33.000000	3.000000	5.000000	0.000000	
75%	1.000000	39.000000	4.000000	7.000000	1.000000	
max	10.000000	60.000000	5.000000	37.000000	1.000000	

In [13]:

```
df.tail()
```

Out[13]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	pr
54803	Technology	region_14	Bachelor's	m	sourcing	1	48	
54804	Operations	region_27	Master's & above	f	other	1	37	
54805	Analytics	region_1	Bachelor's	m	other	1	27	
54806	Sales & Marketing	region_9	NaN	m	sourcing	1	29	
54807	HR	region_22	Bachelor's	m	other	1	27	

In [14]:

```
df.shape
```

Out[14]:

```
(54808, 13)
```

In [15]:

```
df['education'].mode()
```

Out[15]:

```
0    Bachelor's
dtype: object
```

In [16]:

```
df['education'].mode()[0]
```

Out[16]:

```
"Bachelor's"
```

In [17]:

```
df['education'].head(30)
```

Out[17]:

```
0     Master's & above
1           Bachelor's
2           Bachelor's
3           Bachelor's
4           Bachelor's
5           Bachelor's
6           Bachelor's
7     Master's & above
8           Bachelor's
9     Master's & above
10                NaN
11          Bachelor's
12          Bachelor's
13    Master's & above
14    Master's & above
15          Bachelor's
16          Bachelor's
17          Bachelor's
18          Bachelor's
19          Bachelor's
20          Bachelor's
21                NaN
22          Bachelor's
23          Bachelor's
24    Master's & above
25          Bachelor's
26          Bachelor's
27          Bachelor's
28          Bachelor's
29          Bachelor's
Name: education, dtype: object
```

In [18]:

```
df['education'].fillna(df['education'].mode()[0],inplace=True)
```

In [19]:

```
df['education'].head(30)
```

Out[19]:

```
0     Master's & above
1           Bachelor's
2           Bachelor's
3           Bachelor's
4           Bachelor's
5           Bachelor's
6           Bachelor's
7     Master's & above
8           Bachelor's
9     Master's & above
10          Bachelor's
11          Bachelor's
12          Bachelor's
13    Master's & above
14    Master's & above
15          Bachelor's
16          Bachelor's
17          Bachelor's
18          Bachelor's
19          Bachelor's
20          Bachelor's
21          Bachelor's
22          Bachelor's
23          Bachelor's
24    Master's & above
25          Bachelor's
26          Bachelor's
27          Bachelor's
28          Bachelor's
29          Bachelor's
Name: education, dtype: object
```

In [20]:

```
df['previous_year_rating'].mean()
```

Out[20]:

```
3.329255780917055
```

In [21]:

```
df['previous_year_rating'].fillna(df['previous_year_rating'].mean(),inplace=True)
```

In [22]:

```
df['previous_year_rating'].isnull().any()
```

Out[22]:

```
False
```

In [23]:

```
df.isnull().any()
```

Out[23]:

```
department            False
region                False
education             False
gender                False
recruitment_channel   False
no_of_trainings       False
age                   False
previous_year_rating  False
length_of_service     False
KPIs_met >80%         False
awards_won?           False
avg_training_score    False
is_promoted           False
dtype: bool
```

univariate analysis of cateogarical data

In [24]:

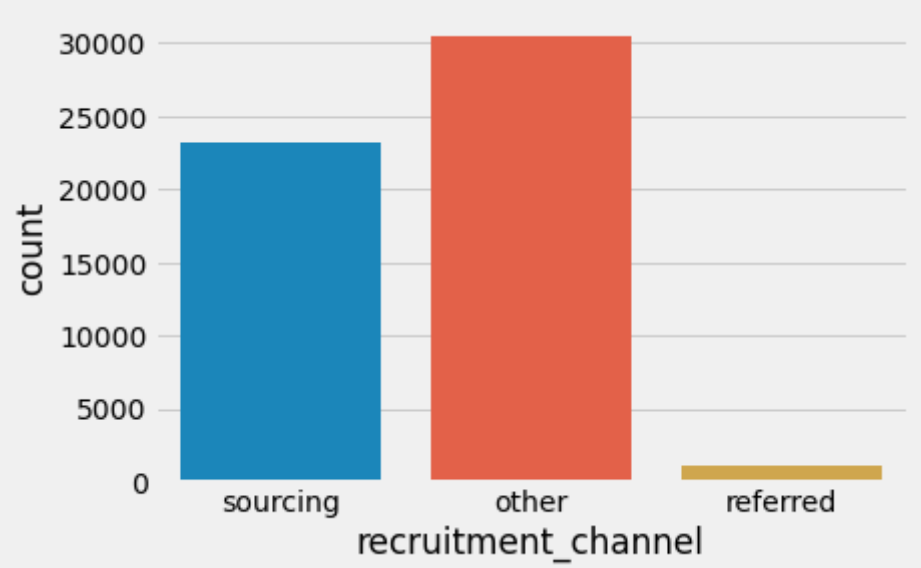
```
import warnings
warnings.filterwarnings('ignore')
```

In [25]:

```
plt.style.use('fivethirtyeight')
```

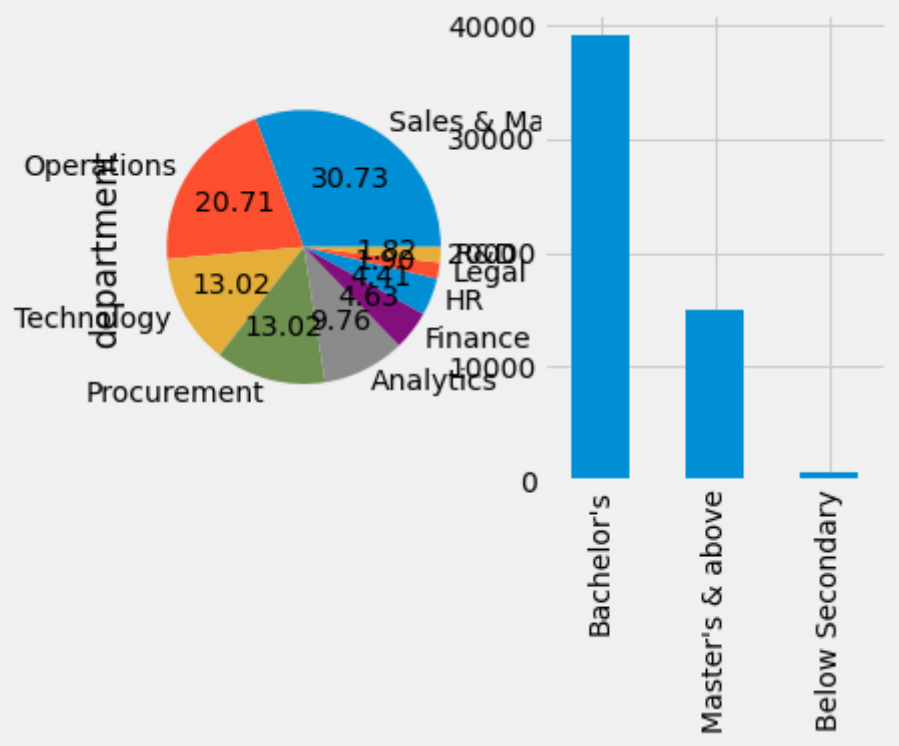
In [26]:

```
sns.countplot(df['recruitment_channel'])
plt.show()
```



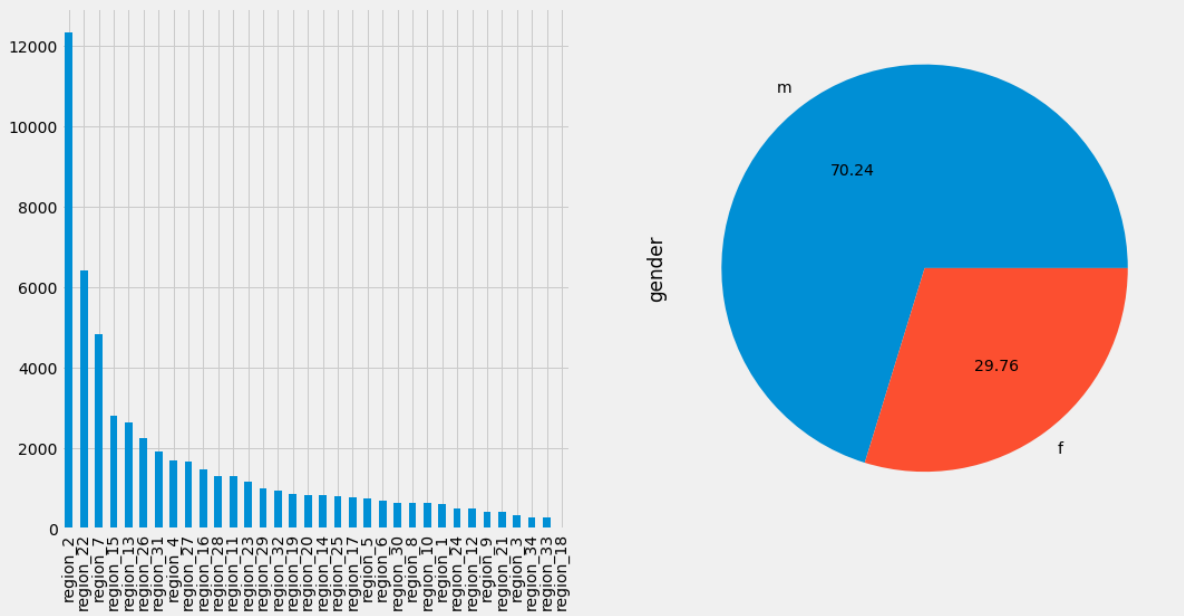
In [27]:

```
plt.figure(figsize=(6,4))
plt.subplot(121)
df['department'].value_counts().plot(kind='pie',autopct='%.2f')
plt.subplot(122)
df['education'].value_counts().plot(kind='bar')
plt.show()
```



In [28]:

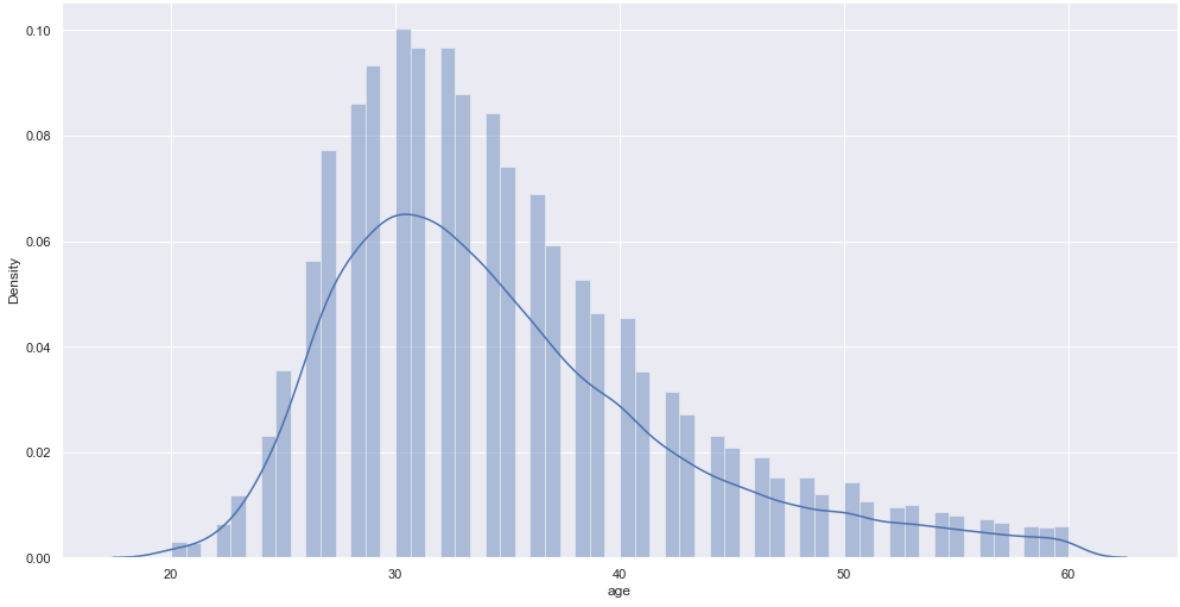
```
plt.figure(figsize=(16,8))
plt.subplot(121)
df['region'].value_counts().plot(kind='bar')
plt.subplot(122)
df['gender'].value_counts().plot(kind='pie',autopct='%.2f')
plt.show()
```



univariate anlysis of numeric data

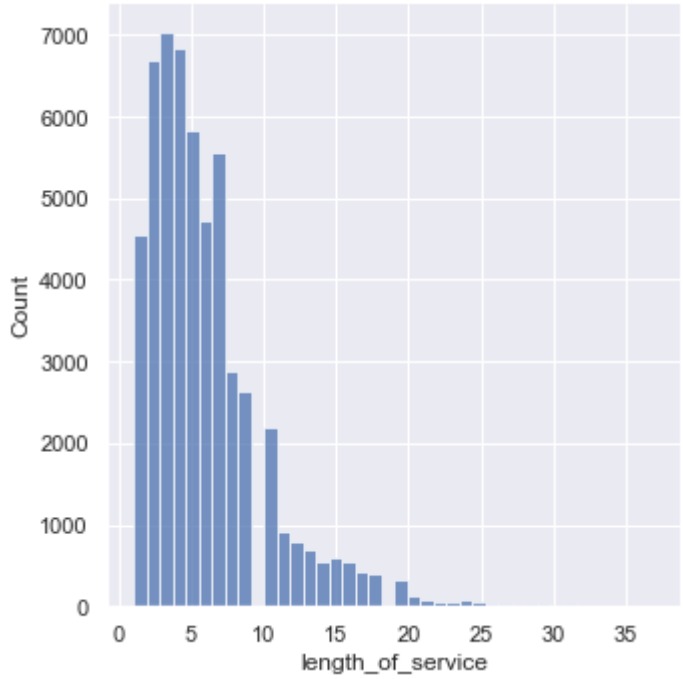
In [29]:

```
sns.set(rc = {'figure.figsize':(15,8)})
sns.distplot(df['age'],bins=60)
plt.show()
```



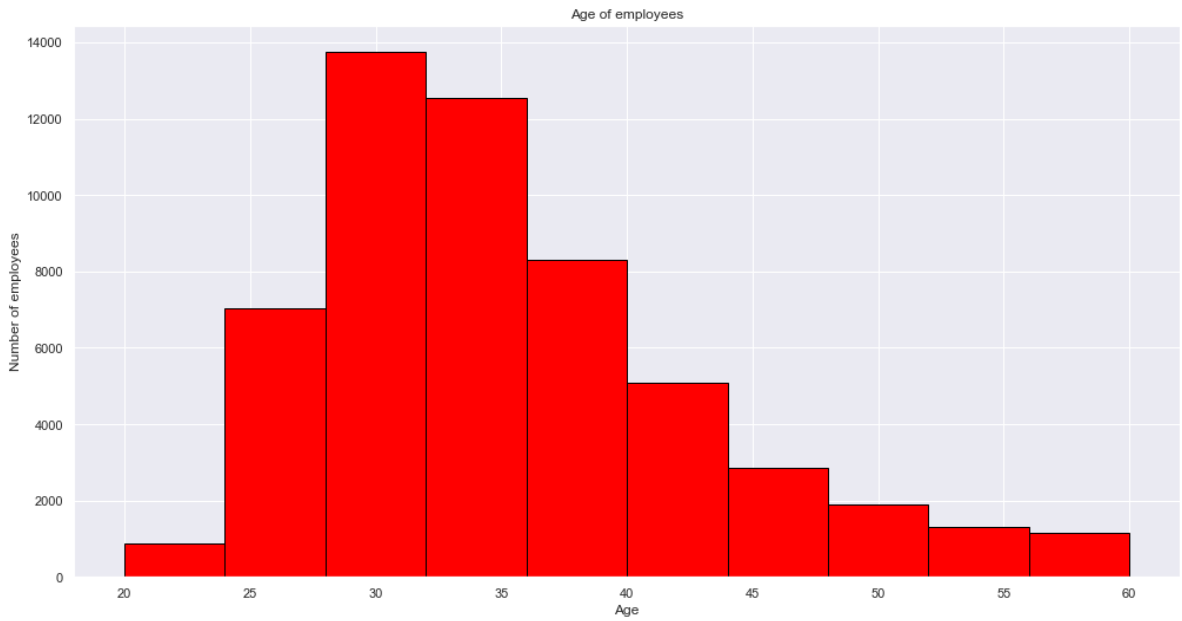
In [30]:

```
sns.displot(df['length_of_service'],kde=False,bins=40)
plt.show()
```



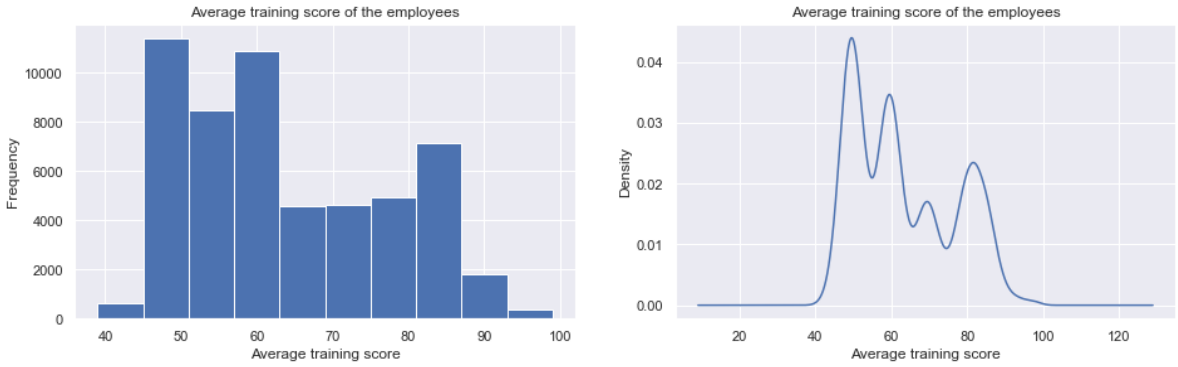
In [31]:

```
plt.hist(df['age'],bins=10,color='red',edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Number of employees')
plt.title('Age of employees')
plt.show()
```



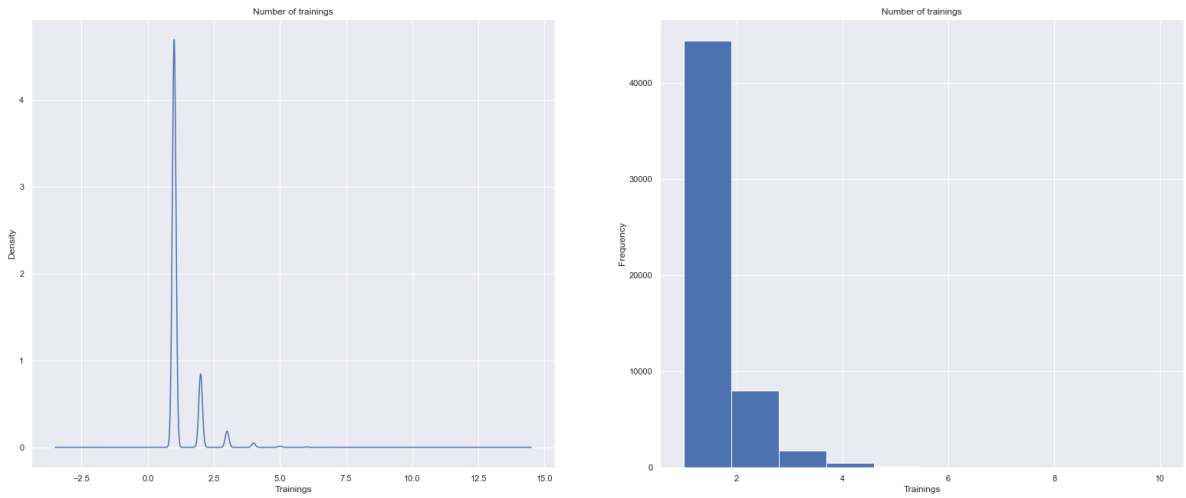
In [32]:

```
plt.figure(figsize=(14,4))
plt.subplot(121)
plt.xlabel('Average training score')
plt.ylabel('Employees')
plt.title('Average training score of the employees')
df.avg_training_score.plot(kind='hist')
plt.subplot(122)
plt.xlabel('Average training score')
plt.ylabel('Employees')
plt.title('Average training score of the employees')
df.avg_training_score.plot(kind='kde')
plt.show()
```



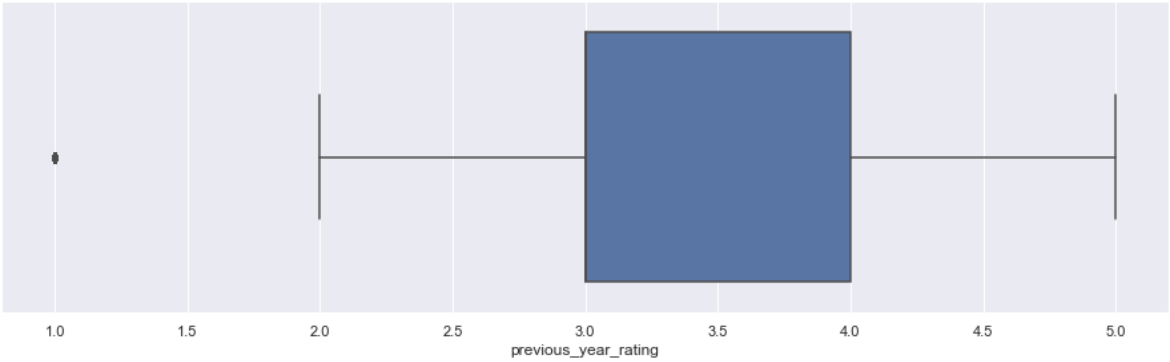
In [33]:

```
plt.figure(figsize=(24,10))
plt.subplot(121)
plt.xlabel('Trainings')
plt.ylabel('Employees')
plt.title('Number of trainings')
df.no_of_trainings.plot(kind='kde')
plt.subplot(122)
plt.xlabel('Trainings')
plt.ylabel('Employees')
plt.title('Number of trainings')
df.no_of_trainings.plot(kind='hist')
plt.show()
```



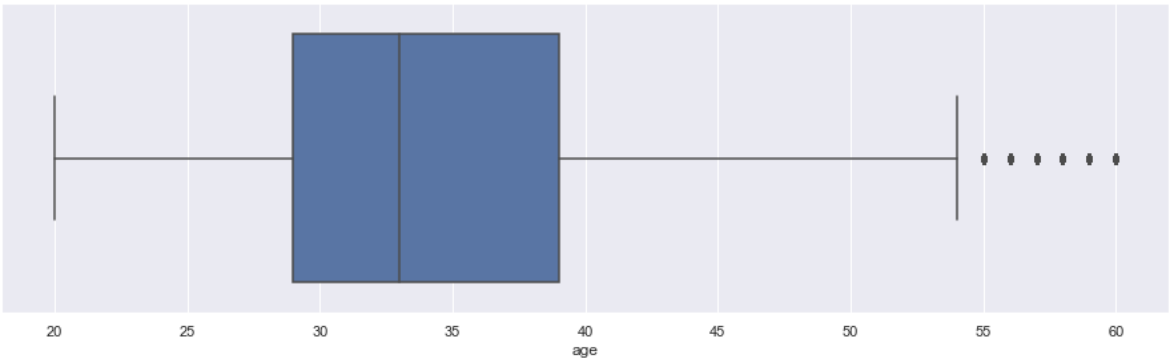
In [34]:

```
plt.figure(figsize=(14,4))
sns.boxplot(df['previous_year_rating'])
plt.show()
```



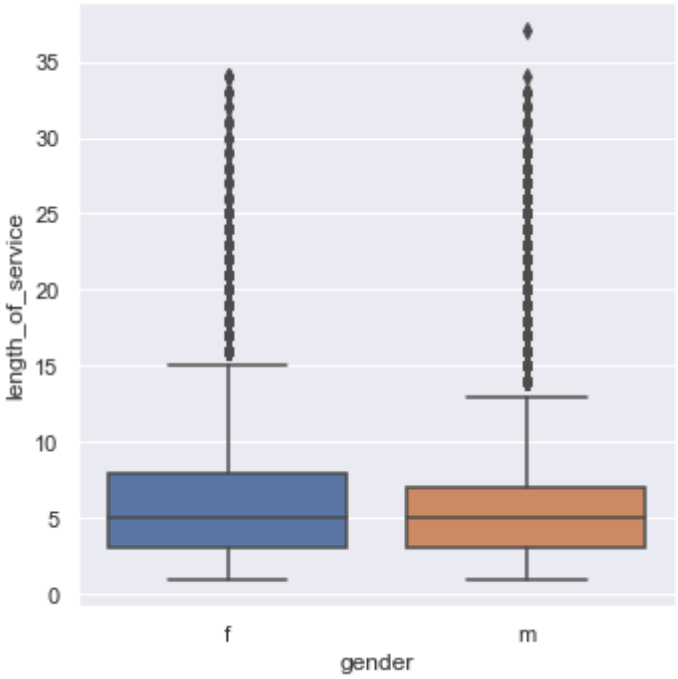
In [35]:

```
plt.figure(figsize=(14,4))
sns.boxplot(df['age'])
plt.show()
```



In [36]:

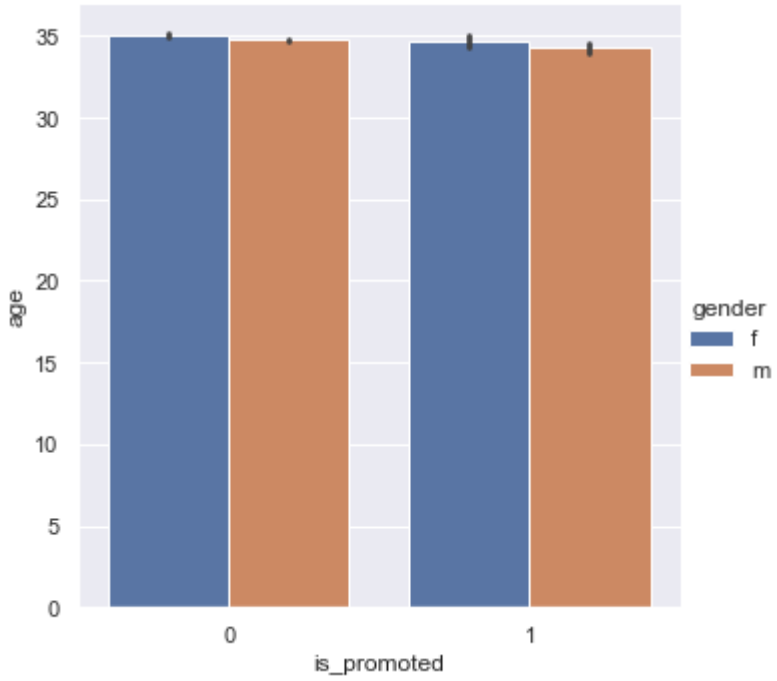
```
sns.catplot(x='gender',y='length_of_service',kind='box',data=df)
plt.show()
```



Multivarite Analysis

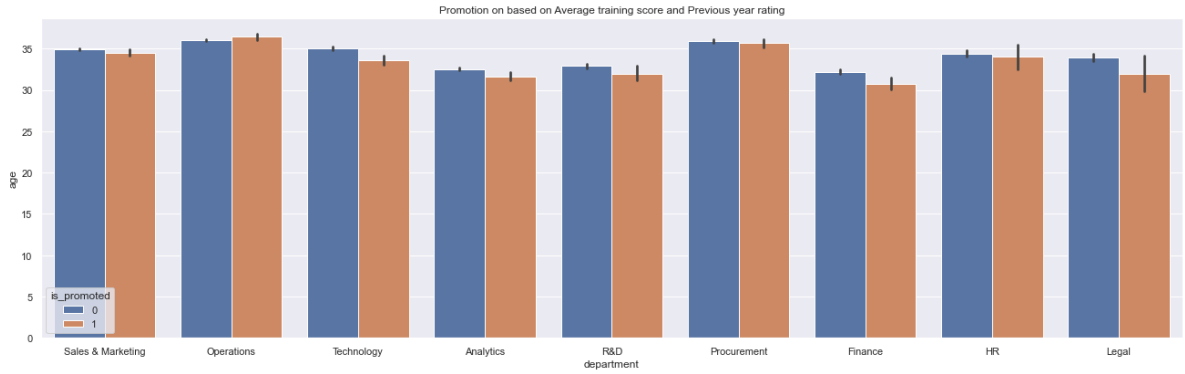
In [37]:

```
sns.catplot(y='age',x='is_promoted',hue='gender',kind='bar',data=df)
plt.show()
```



In [38]:

```
plt.figure(figsize=(20,6))
sns.barplot(x='department',y='age',hue='is_promoted',data=df)
plt.title('Promotion on based on Average training score and Previous year rating')
plt.show()
```

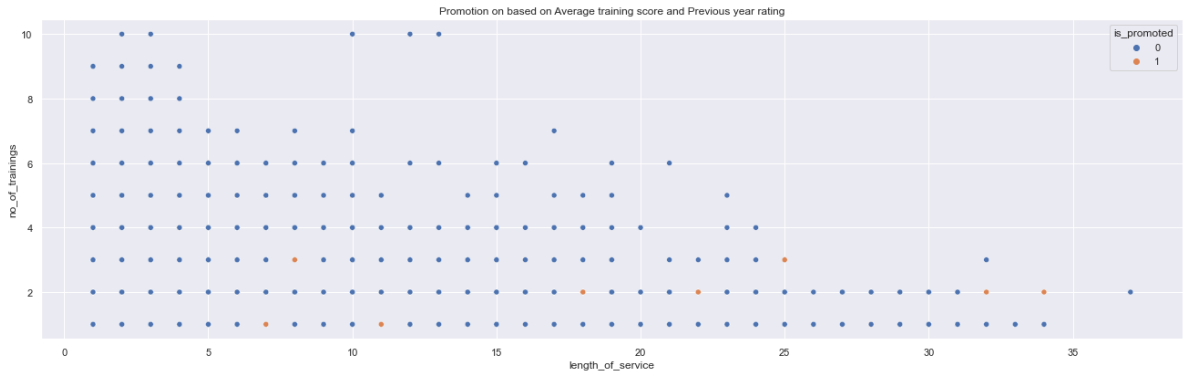


In [39]:

```
plt.figure(figsize=(20,6))
sns.scatterplot(df['length_of_service'],df['no_of_trainings'],df['is_promoted'],color='purp
plt.title('Promotion on based on Average training score and Previous year rating')
```

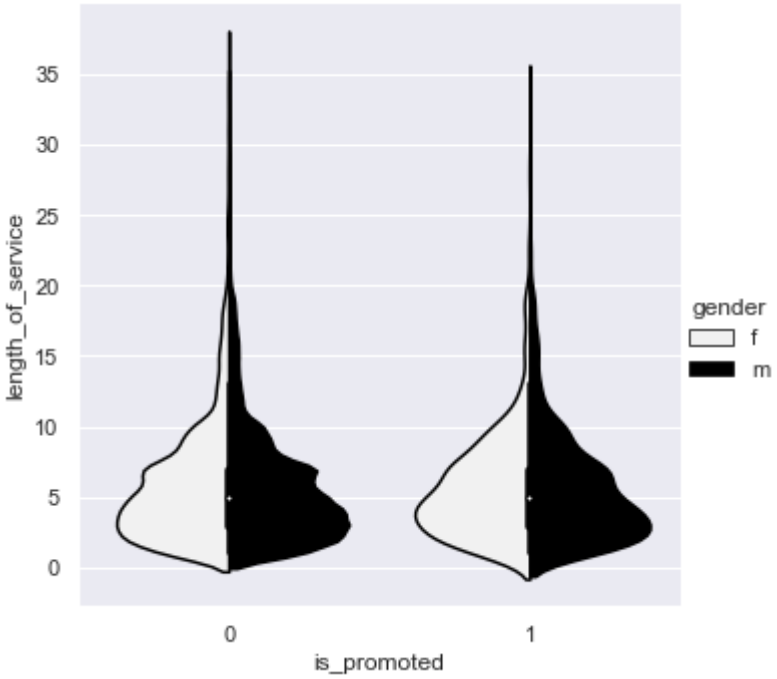
Out[39]:

```
Text(0.5, 1.0, 'Promotion on based on Average training score and Previous ye
ar rating')
```



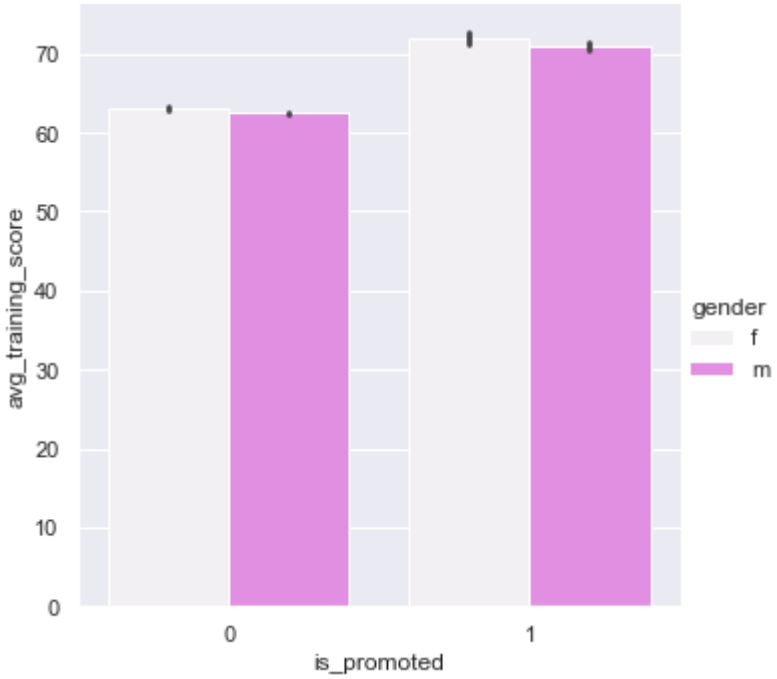
In [40]:

```
sns.catplot(x='is_promoted',y='length_of_service',hue='gender',kind='violin',split=True,col
plt.show()
```



In [41]:

```
sns.catplot(x='is_promoted',y='avg_training_score',hue='gender',kind='bar',color='violet',d
plt.show()
```

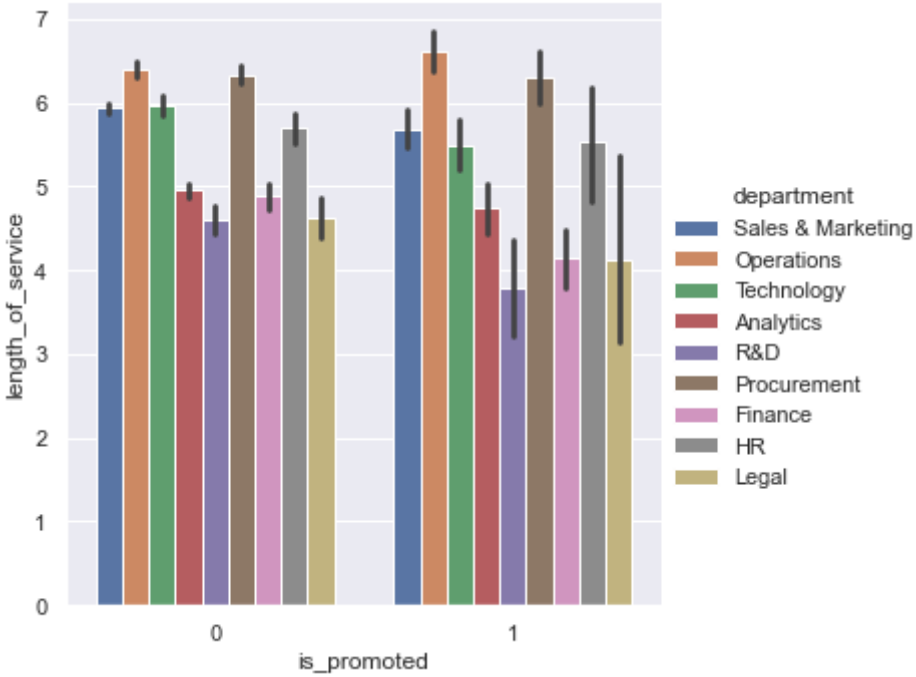


In [42]:

```
sns.catplot(x='is_promoted',y='length_of_service',hue='department',kind='bar',data=df)
```

Out[42]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79ecd0b20>
```

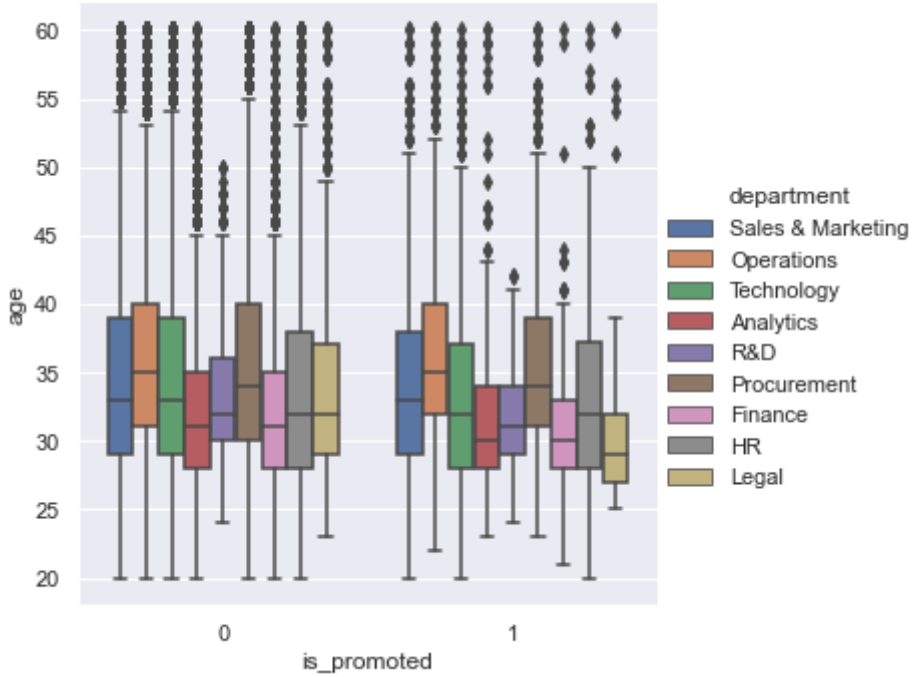


In [43]:

```
sns.catplot(x='is_promoted',y='age',hue='department',kind='box',data=df)
```

Out[43]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79edd5760>
```

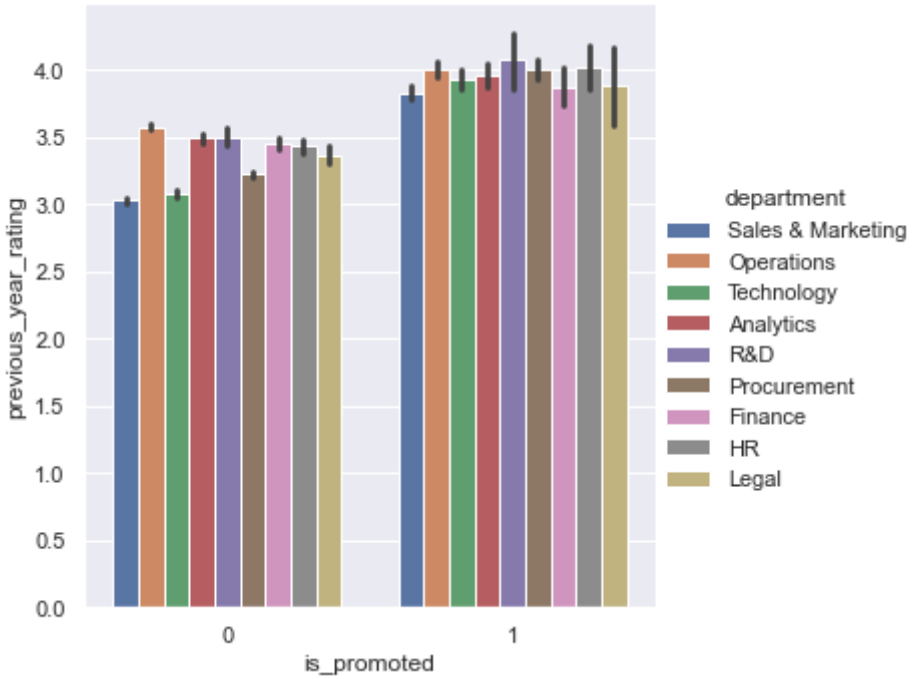


In [45]:

```
sns.catplot(x='is_promoted',y='previous_year_rating',hue='department',kind='bar',data=df)
```

Out[45]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79ede9160>
```

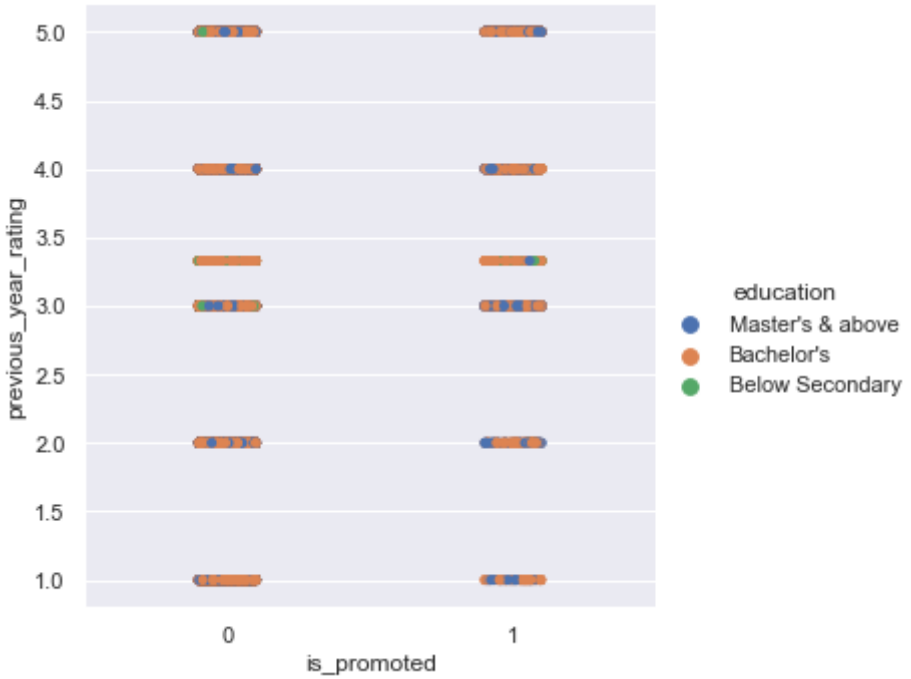


In [46]:

```
sns.catplot(x='is_promoted',y='previous_year_rating',hue='education',kind='strip',data=df)
```

Out[46]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79ece7910>
```

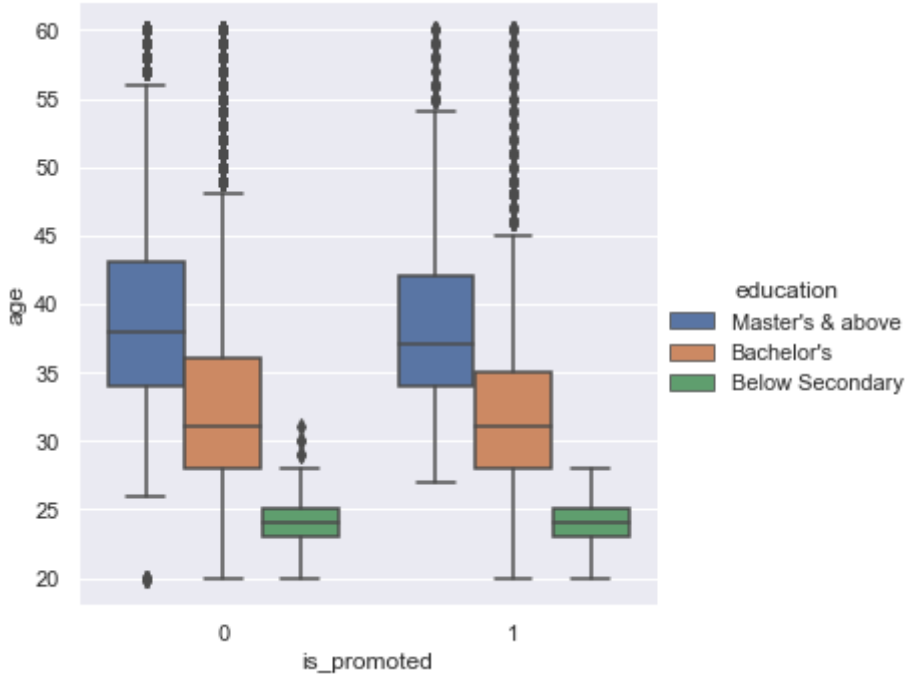


In [47]:

```
sns.catplot(x='is_promoted',y='age',hue='education',kind='box',data=df)
```

Out[47]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79ede90d0>
```

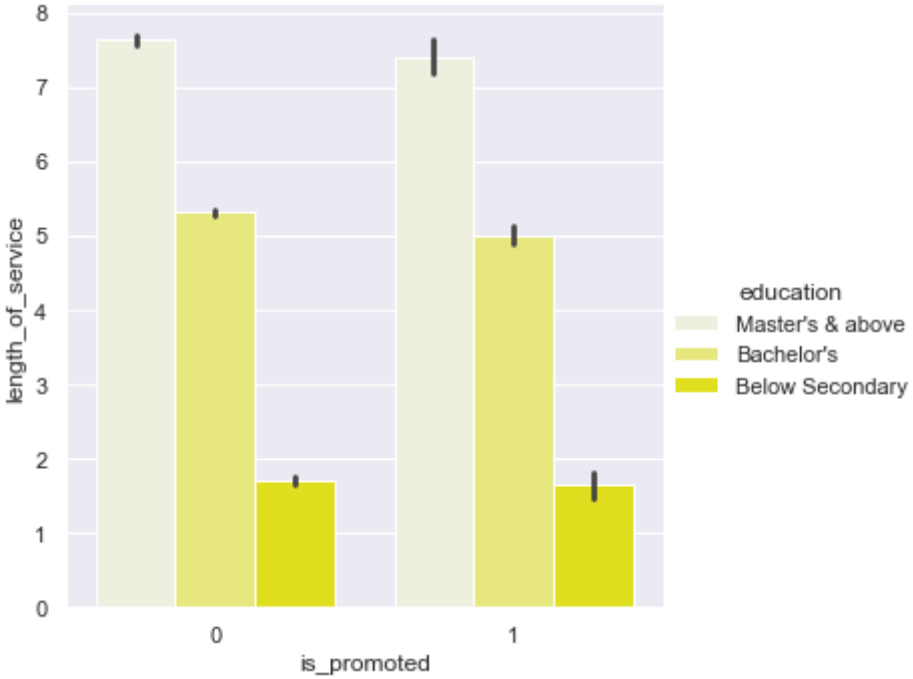


In [48]:

```
sns.catplot(x='is_promoted',y='length_of_service',hue='education',kind='bar',color='yellow'
```

Out[48]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79f942610>
```

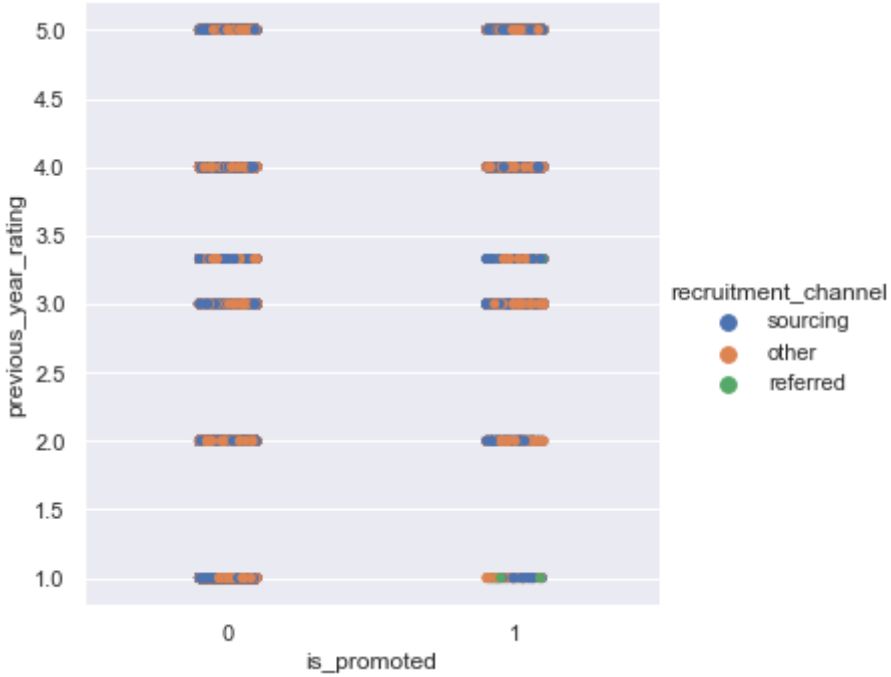


In [51]:

```
sns.catplot(x='is_promoted',y='previous_year_rating',hue='recruitment_channel',kind='strip'
```

Out[51]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79ea8e580>
```

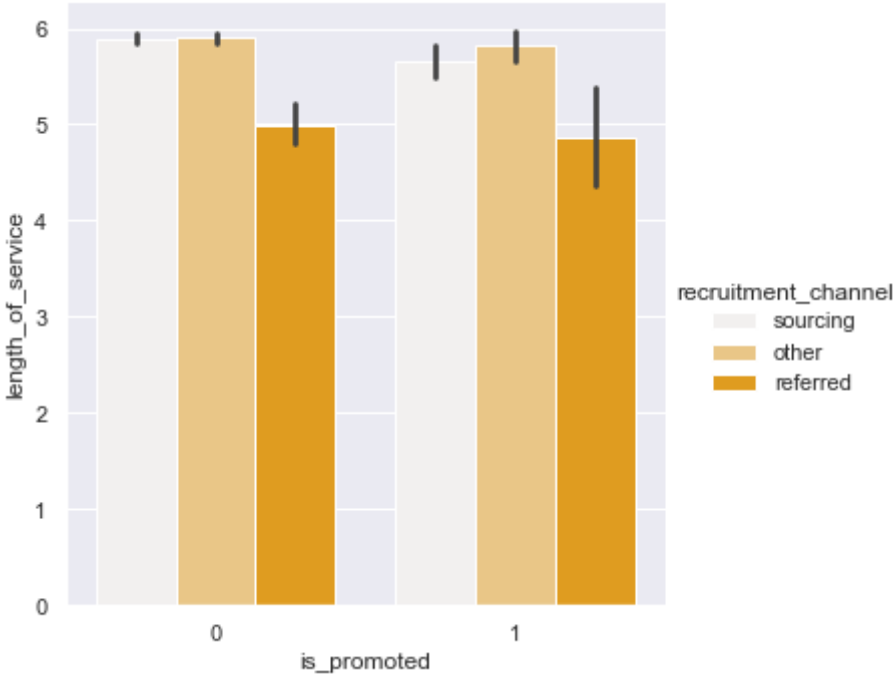


In [52]:

```
sns.catplot(x='is_promoted',y='length_of_service',hue='recruitment_channel',kind='bar',colo
```

Out[52]:

```
<seaborn.axisgrid.FacetGrid at 0x1d7a0469820>
```

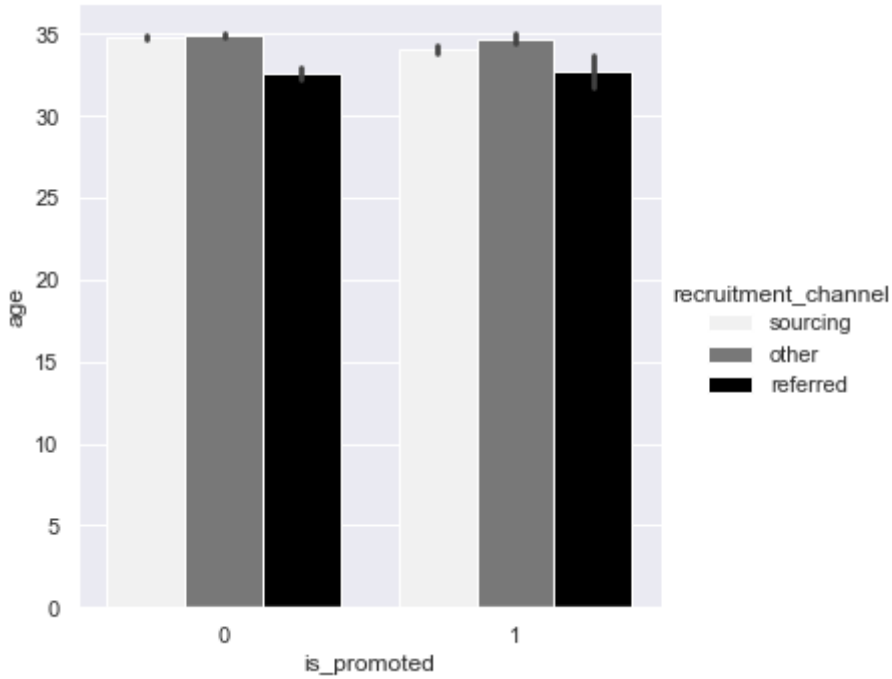


In [53]:

```
sns.catplot(x='is_promoted',y='age',hue='recruitment_channel',kind='bar',color='black',data
```

Out[53]:

```
<seaborn.axisgrid.FacetGrid at 0x1d79f2522b0>
```



In [54]:

```
df.shape
```

Out[54]:

```
(54808, 13)
```

DESCRIPTIVE ANALYSIS

In [55]:

```
df.describe(include='all')
```

Out[55]:

	department	region	education	gender	recruitment_channel	no_of_trainings	
count	54808	54808	54808	54808	54808	54808.000000	54808.0(
unique	9	34	3	2	3	NaN	
top	Sales & Marketing	region_2	Bachelor's	m	other	NaN	
freq	16840	12343	39078	38496	30446	NaN	
mean	NaN	NaN	NaN	NaN	NaN	1.253011	34.8(
std	NaN	NaN	NaN	NaN	NaN	0.609264	7.6(
min	NaN	NaN	NaN	NaN	NaN	1.000000	20.0(
25%	NaN	NaN	NaN	NaN	NaN	1.000000	29.0(
50%	NaN	NaN	NaN	NaN	NaN	1.000000	33.0(
75%	NaN	NaN	NaN	NaN	NaN	1.000000	39.0(
max	NaN	NaN	NaN	NaN	NaN	10.000000	60.0(

Data preprocessing

In [56]:

```
df.duplicated().sum()
```

Out[56]:

```
156
```

In [57]:

```
df.drop_duplicates(inplace=True)
```

Feature selection

which can be used for eliminating the highly correlated values between coloumns

In [58]:

```
df.corr()
```

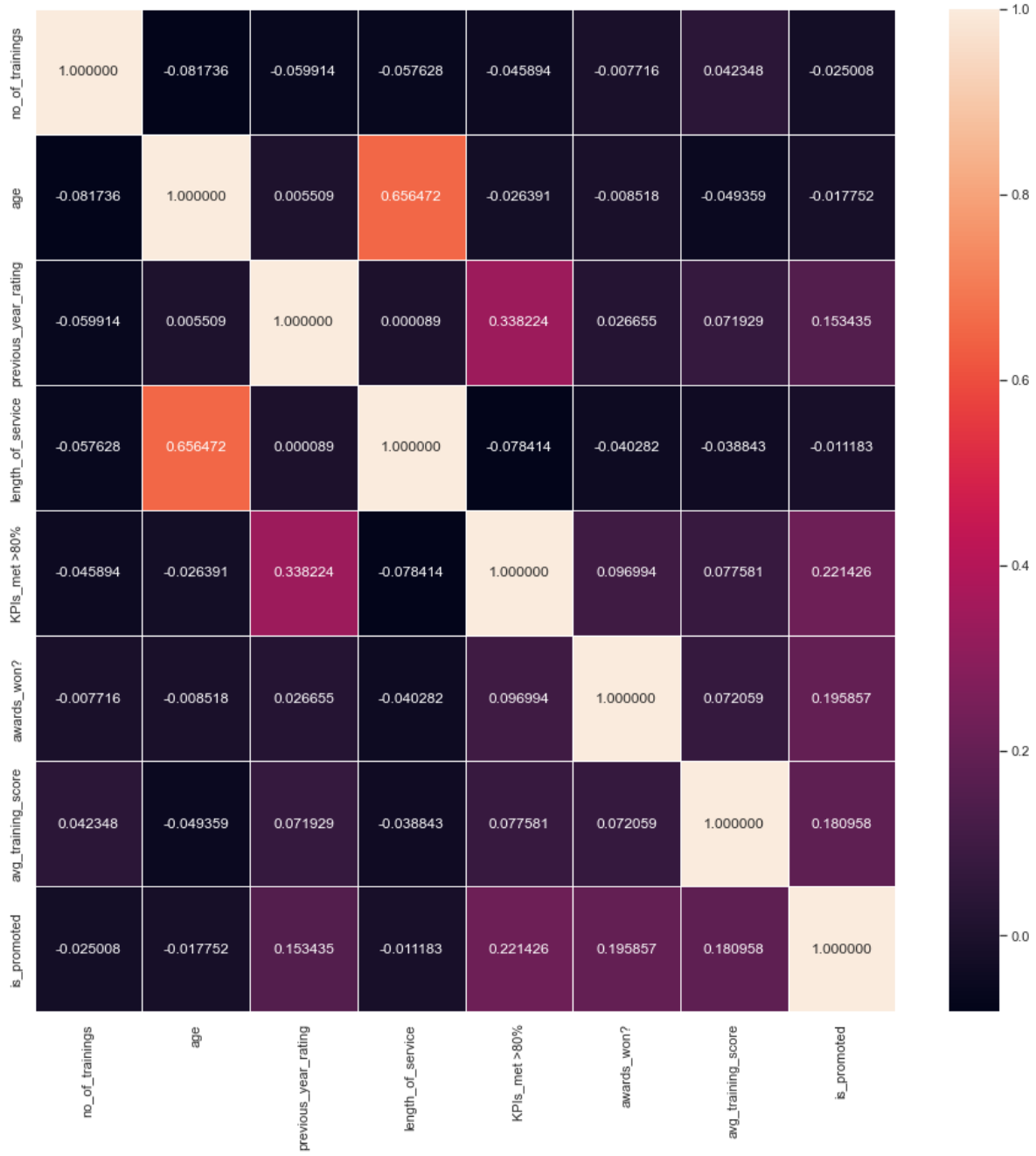
Out[58]:

	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_m >8
no_of_trainings	1.000000	-0.081736	-0.059914	-0.057628	-0.0458
age	-0.081736	1.000000	0.005509	0.656472	-0.0263
previous_year_rating	-0.059914	0.005509	1.000000	0.000089	0.3382
length_of_service	-0.057628	0.656472	0.000089	1.000000	-0.0784
KPIs_met >80%	-0.045894	-0.026391	0.338224	-0.078414	1.0000
awards_won?	-0.007716	-0.008518	0.026655	-0.040282	0.0969
avg_training_score	0.042348	-0.049359	0.071929	-0.038843	0.0775
is_promoted	-0.025008	-0.017752	0.153435	-0.011183	0.2214

plt.figure(figsize=(14,14)) sns.heatmap(data.corr(),annot=True,cbar=True,linewidth=True,fmt='f') plt.show()

In [59]:

```
plt.figure(figsize=(14,14))
sns.heatmap(df.corr(),annot=True,cbar=True,linewidth=True,fmt='f')
plt.show()
```



Here i do no not have any values between 80% to 90% correlated but in this case the highly correlated value is around 65% that's i am not going to eliminate any column

Removing duplicates rows

In [60]:

```
df.duplicated().sum()
```

Out[60]:

```
0
```

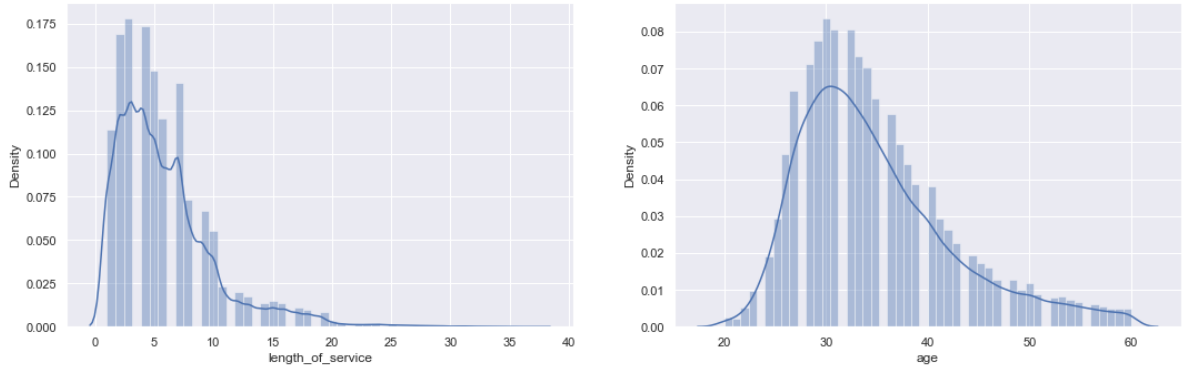
In [61]:

```
df.drop_duplicates(inplace=True)
```

Removing outlier

In [62]:

```
plt.figure(figsize=(16,5))
plt.subplot(121)
sns.distplot(df['length_of_service'])
plt.subplot(122)
sns.distplot(df['age'])
plt.show()
```



In [63]:

```
df['length_of_service'].describe()
```

Out[63]:

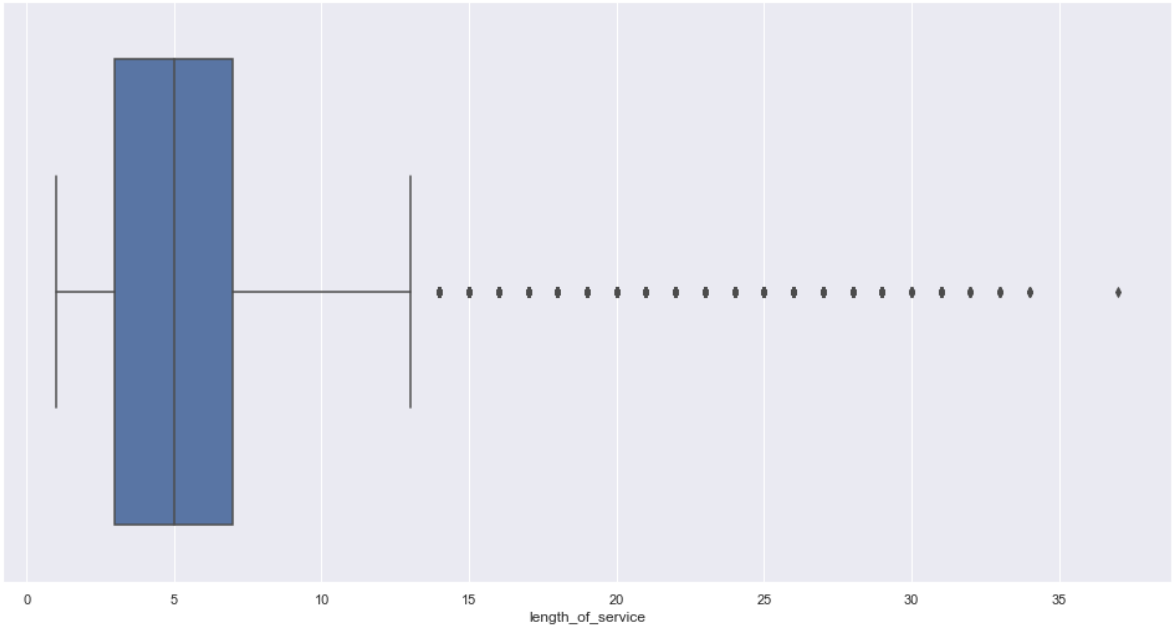
```
count    54652.000000
mean         5.873838
std          4.265650
min          1.000000
25%          3.000000
50%          5.000000
75%          7.000000
max         37.000000
Name: length_of_service, dtype: float64
```

In [64]:

```
sns.boxplot(df['length_of_service'])
```

Out[64]:

```
<AxesSubplot:xlabel='length_of_service'>
```



In [65]:

```
df['age'].describe()
```

Out[65]:

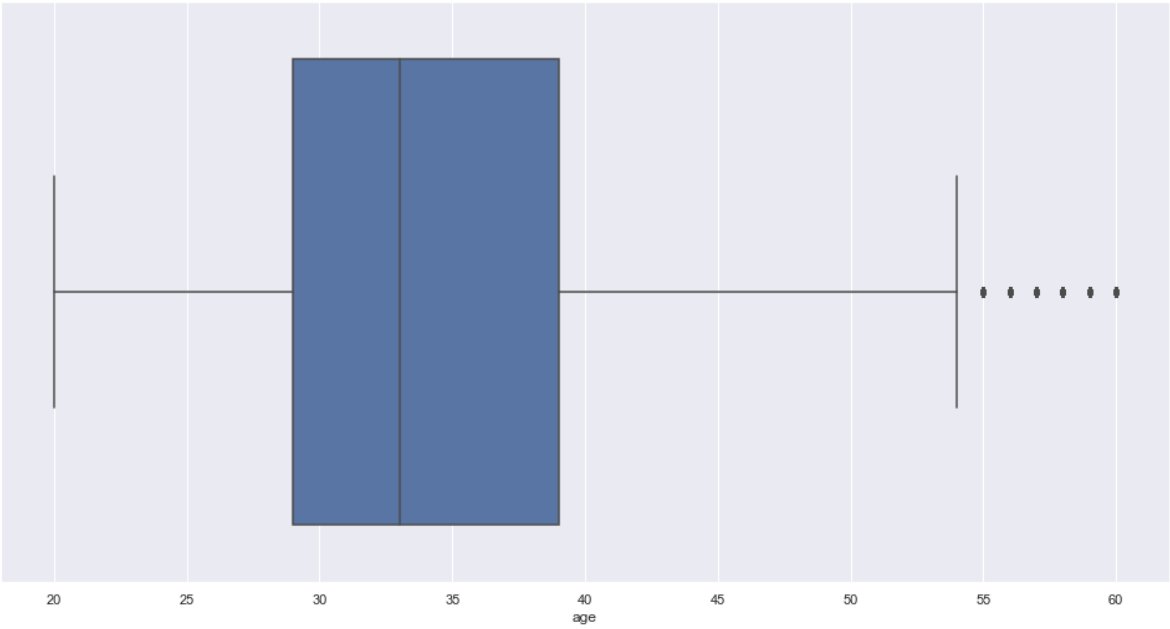
```
count    54652.000000
mean        34.820684
std          7.660173
min         20.000000
25%         29.000000
50%         33.000000
75%         39.000000
max         60.000000
Name: age, dtype: float64
```

In [66]:

```
sns.boxplot(df['age'])
```

Out[66]:

```
<AxesSubplot:xlabel='age'>
```



In [67]:

```
percentile25_los=df['length_of_service'].quantile(0.25)
percentile75_los=df['length_of_service'].quantile(0.75)
percentile25_age=df['age'].quantile(0.25)
percentile75_age=df['age'].quantile(0.75)
print("Q2 of los:",percentile25_los)
print("Q4 of los:",percentile75_los)
print("Q2 of age:",percentile25_age)
print("Q4 of age:",percentile75_age)
```

```
Q2 of los: 3.0
Q4 of los: 7.0
Q2 of age: 29.0
Q4 of age: 39.0
```

In [68]:

```
iqr_los=percentile75_los-percentile25_los
print("IQR of los",iqr_los)
iqr_age=percentile75_age-percentile25_age
print("IQR of age",iqr_age)
```

```
IQR of los 4.0
IQR of age 10.0
```

In [69]:

```
upper_limit_los=percentile75_los+1.5*iqr_los
lower_limit_los=percentile25_los-1.5*iqr_los
upper_limit_age=percentile75_age+1.5*iqr_age
lower_limit_age=percentile25_age-1.5*iqr_age
```

In [70]:

```
print("upper limit of los",upper_limit_los)
print("lower limit of los",lower_limit_los)
print("upper limit of age",upper_limit_age)
print("lower limit of age",lower_limit_age)
```

```
upper limit of los 13.0
lower limit of los -3.0
upper limit of age 54.0
lower limit of age 14.0
```

In [71]:

```
df[df['length_of_service']>13]
```

Out[71]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	pr
13	Technology	region_29	Master's & above	m	other	2	39	
42	HR	region_2	Bachelor's	m	sourcing	1	59	
60	Sales & Marketing	region_4	Master's & above	m	other	1	50	
74	Sales & Marketing	region_7	Bachelor's	m	other	1	50	
99	Finance	region_2	Master's & above	m	other	1	60	
...	...	...	...	...	...	...	...	
54691	Analytics	region_2	Master's & above	m	sourcing	1	47	
54695	Operations	region_2	Bachelor's	f	other	2	52	
54697	Sales & Marketing	region_2	Bachelor's	m	sourcing	1	47	
54754	Technology	region_26	Bachelor's	f	other	1	42	
54803	Technology	region_14	Bachelor's	m	sourcing	1	48	

3488 rows × 13 columns

In [72]:

```
df[(df['age']>54) | (df['age']<14)]
```

Out[72]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	pr
33	Operations	region_2	Bachelor's	m	sourcing	2	57	
42	HR	region_2	Bachelor's	m	sourcing	1	59	
49	Procurement	region_2	Master's & above	f	sourcing	1	56	
99	Finance	region_2	Master's & above	m	other	1	60	
307	Procurement	region_2	Master's & above	f	other	1	58	
...	...	...	...	...	...	...	...	
54580	Technology	region_7	Master's & above	f	other	1	57	
54617	Sales & Marketing	region_2	Master's & above	f	other	1	57	
54628	Sales & Marketing	region_2	Bachelor's	m	sourcing	1	57	
54749	Procurement	region_2	Master's & above	f	sourcing	1	55	
54792	Sales & Marketing	region_14	Bachelor's	m	other	1	59	

1435 rows × 13 columns

CAPPING OF OUTLIERS

In [73]:

```
df['length_of_service']=np.where(
df['length_of_service']>upper_limit_los,
upper_limit_los,
np.where(
df['length_of_service']<lower_limit_los,
lower_limit_los,df['length_of_service'])
)
```

In [74]:

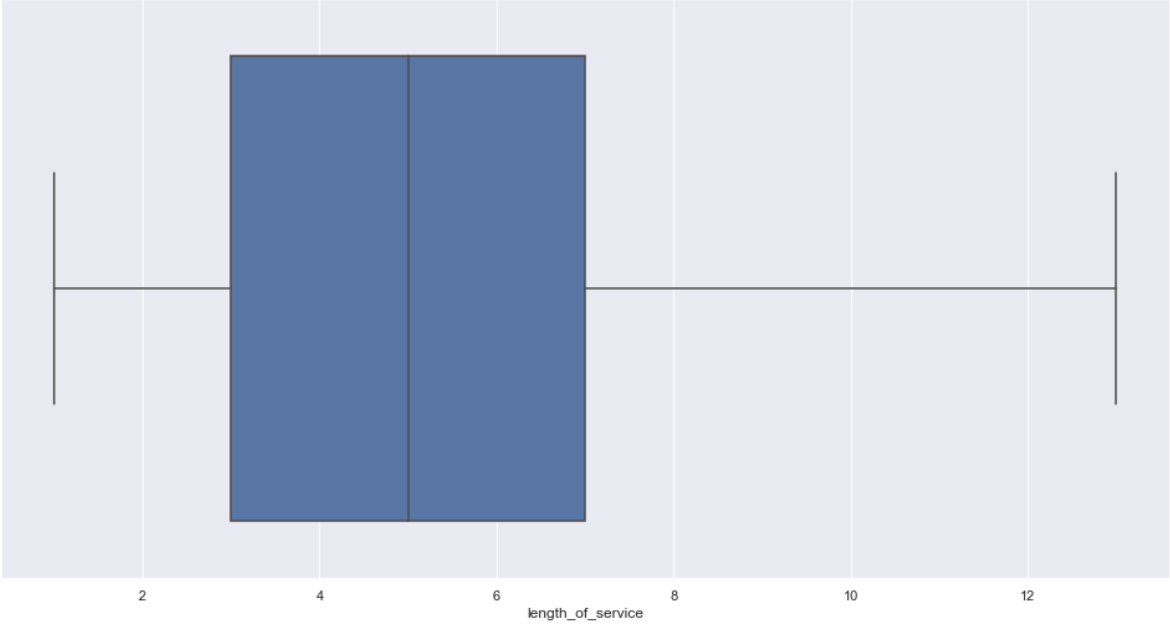
```
df['length_of_service'].describe()
```

Out[74]:

```
count    54652.000000
mean         5.578369
std          3.413165
min          1.000000
25%          3.000000
50%          5.000000
75%          7.000000
max         13.000000
Name: length_of_service, dtype: float64
```

In [75]:

```
sns.boxplot(df['length_of_service'])
plt.show()
```



In [76]:

```
df['age'].describe()
```

Out[76]:

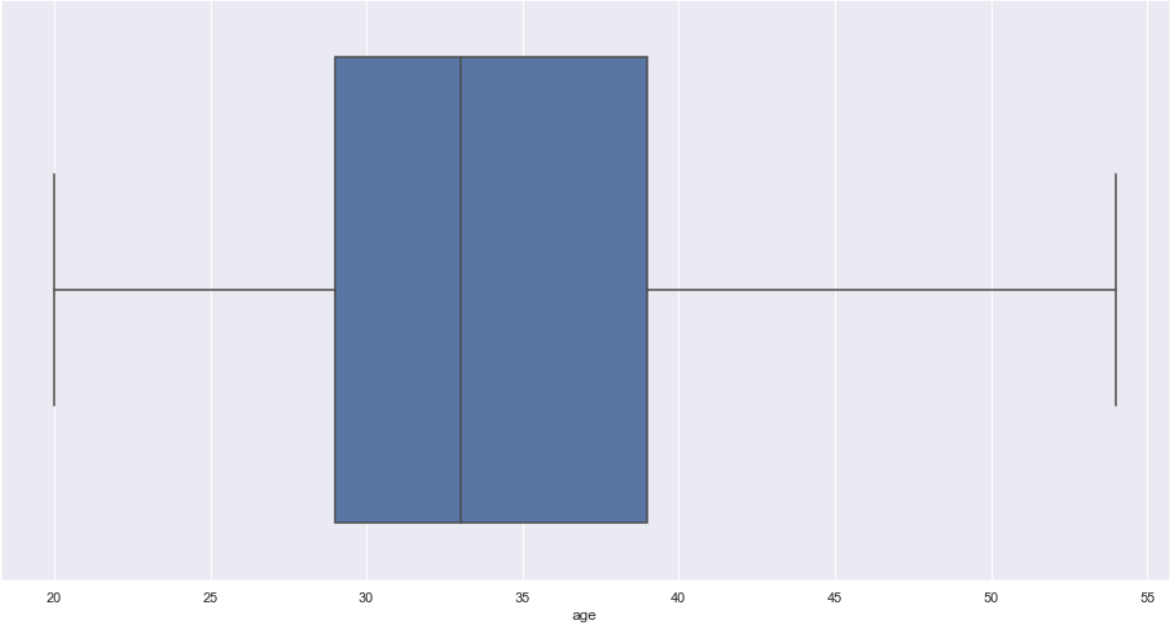
```
count    54652.000000
mean        34.820684
std          7.660173
min         20.000000
25%         29.000000
50%         33.000000
75%         39.000000
max         60.000000
Name: age, dtype: float64
```

In [77]:

```
df['age']=np.where(
df['age']>upper_limit_age,upper_limit_age
    ,np.where(
    df['age']<lower_limit_age,lower_limit_age,
    df['age']))
```

In [78]:

```
sns.boxplot(df['age'])
plt.show()
```



In [79]:

```
df
```

Out[79]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	pr
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35.0	
1	Operations	region_22	Bachelor's	m	other	1	30.0	
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34.0	
3	Sales & Marketing	region_23	Bachelor's	m	other	2	39.0	
4	Technology	region_26	Bachelor's	m	other	1	45.0	
...	...	...	...	...	...	...	...	
54803	Technology	region_14	Bachelor's	m	sourcing	1	48.0	
54804	Operations	region_27	Master's & above	f	other	1	37.0	
54805	Analytics	region_1	Bachelor's	m	other	1	27.0	
54806	Sales & Marketing	region_9	Bachelor's	m	sourcing	1	29.0	
54807	HR	region_22	Bachelor's	m	other	1	27.0	

54652 rows × 13 columns

SPLITING THE DATA INTO TRAIN AND TEST DATA

In [80]:

```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [81]:

```
y.value_counts()
```

Out[81]:

```
0    49987
1     4665
Name: is_promoted, dtype: int64
```

In [82]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

IMPUTING MISSING VALUES

In [83]:

```
X_train.shape,X_test.shape
```

Out[83]:

```
((43721, 12), (10931, 12))
```

In [84]:

```
X_train.sample(5)
```

Out[84]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	p
31833	Procurement	region_14	Bachelor's	m	sourcing	1	29.0	
10024	Operations	region_2	Bachelor's	f	other	2	48.0	
26657	Sales & Marketing	region_7	Bachelor's	m	sourcing	1	33.0	
45270	Procurement	region_2	Master's & above	m	other	2	34.0	
36164	Technology	region_22	Bachelor's	m	other	1	33.0	

In [85]:

```
X_test.sample(5)
```

Out[85]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age	p
46500	Sales & Marketing	region_16	Bachelor's	m	other	1	26.0	
9727	Technology	region_1	Bachelor's	f	sourcing	1	29.0	
22751	Technology	region_14	Bachelor's	f	other	1	32.0	
7452	Operations	region_26	Bachelor's	m	other	2	44.0	
7737	Finance	region_26	Bachelor's	m	other	1	38.0	

In [86]:

```
X_train.isnull().mean()*100
```

Out[86]:

```
department            0.0
region                0.0
education             0.0
gender                0.0
recruitment_channel   0.0
no_of_trainings       0.0
age                   0.0
previous_year_rating  0.0
length_of_service     0.0
KPIs_met >80%         0.0
awards_won?           0.0
avg_training_score    0.0
dtype: float64
```

In [87]:

```
X_test.isnull().mean()*100
```

Out[87]:

```
department            0.0
region                0.0
education             0.0
gender                0.0
recruitment_channel   0.0
no_of_trainings       0.0
age                   0.0
previous_year_rating  0.0
length_of_service     0.0
KPIs_met >80%         0.0
awards_won?           0.0
avg_training_score    0.0
dtype: float64
```

In [88]:

```
df[['education','previous_year_rating']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54652 entries, 0 to 54807
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   education             54652 non-null  object 
 1   previous_year_rating  54652 non-null  float64
dtypes: float64(1), object(1)
memory usage: 1.3+ MB
```

In [89]:

```
df_cat=X_train.select_dtypes(include='object')
df_cat.sample(5)
```

Out[89]:

	department	region	education	gender	recruitment_channel
17661	Technology	region_22	Bachelor's	m	referred
9748	Sales & Marketing	region_7	Bachelor's	m	other
50645	Legal	region_5	Bachelor's	m	sourcing
22218	Technology	region_13	Bachelor's	m	other
34439	Sales & Marketing	region_17	Bachelor's	m	other

In [90]:

```
df_num=X_train.select_dtypes(exclude='object')
df_num.sample(5)
```

Out[90]:

	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?
6086	1	27.0	3.0	3.0	0	0
43570	1	35.0	3.0	6.0	1	0
5849	2	39.0	3.0	5.0	1	1
13339	1	39.0	2.0	3.0	1	0
6231	2	31.0	5.0	5.0	1	0

In [91]:

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PowerTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline,make_pipeline
```

In [92]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [93]:

```
tr1=ColumnTransformer(transformers=[
    ('imputer2',SimpleImputer(strategy='most_frequent'),[2]),
    ('imputer7',SimpleImputer(strategy='median'),[7])
    #('ohe',OneHotEncoder(sparse=False,drop='first',dtype='int32'),[0,1,3,4]),
    #('le',LabelEncoder(),[2]),
    #('scaler',StandardScaler(),slice(0,5))
],remainder='passthrough')
```

In [96]:

```
tr1.fit(X_train)
```

Out[96]:

```
ColumnTransformer(remainder='passthrough',
                  transformers=[('imputer2',
                                 SimpleImputer(strategy='most_frequent'),
[2]),
                                ('imputer7', SimpleImputer(strategy='media
n'),
                                 [7])])
```

In [97]:

```
tr2=ColumnTransformer(transformers=[
    ('le',OrdinalEncoder(),[0])
],remainder='passthrough')
```

In [98]:

```
tr3=ColumnTransformer(transformers=[
    ('ohe',OneHotEncoder(sparse=False,dtype='int32'),[2])
],remainder='passthrough')
```

In [99]:

```
tr4=ColumnTransformer(transformers=[
    ('ohe',OneHotEncoder(sparse=False,dtype='int32'),[11])
],remainder='passthrough')
```

In [100]:

```
tr5=ColumnTransformer(transformers=[
    ('ohe',OneHotEncoder(sparse=False,dtype='int32'),[45])
],remainder='passthrough')
```

In [101]:

```
tr6=ColumnTransformer(transformers=[
    ('ohe',OneHotEncoder(sparse=False,dtype='int32'),[47])
],remainder='passthrough')
```

In [102]:

```
tr7=ColumnTransformer(transformers=[
    ('scaler',MinMaxScaler(),slice(0,55))
])
```

In [103]:

```
tr8=ColumnTransformer(transformers=[
    ('pt',PowerTransformer(method='yeo-johnson'),slice(0,55))
],remainder='passthrough')
```

In [104]:

```
tr9=RandomForestClassifier()
```

In [105]:

```
pipe1=Pipeline([
    ('tr1',tr1),
    ('tr2',tr2),
    ('tr3',tr3),
    ('tr4',tr4),
    ('tr5',tr5),
    ('tr6',tr6),
    ('tr7',tr7),
    ('tr8',tr8),
    ('tr9',tr9)
])
```

In [107]:

```
pipe1.fit(X_train,y_train)
```

Out[107]:

```
Pipeline(steps=[('tr1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('imputer2',
                                                  SimpleImputer(strategy='mo
st_frequent'),
                                                  [2]),
                                                 ('imputer7',
                                                  SimpleImputer(strategy='me
dian'),
                                                  [7])])),
                ('tr2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('le', OrdinalEncoder(),
                                                  [0])])),
                ('tr3',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('oh...
                                                  [45])])),
                ('tr6',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(dtype='int3
2',
                                                                sparse=Fals
e),
                                                  [47])])),
                ('tr7',
                 ColumnTransformer(transformers=[('scaler', MinMaxScaler(),
                                                  slice(0, 55, None))])),
                ('tr8',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('pt', PowerTransformer(),
                                                  slice(0, 55, None))])),
                ('tr9', RandomForestClassifier())])
```

In [108]:

```
y_pred1=pipe1.predict(X_test)
print('RandomForestClassifier:')
print('confusion_matrix:')
print(confusion_matrix(y_test,y_pred1))
print('classification_report:')
print(classification_report(y_test,y_pred1))
```

```
RandomForestClassifier:
confusion_matrix:
[[9807  183]
 [ 866   75]]
classification_report:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      9990
           1       0.29      0.08      0.13       941

    accuracy                           0.90     10931
   macro avg       0.60      0.53      0.54     10931
weighted avg       0.86      0.90      0.88     10931
```

In [109]:

```
tr10=DecisionTreeClassifier()
```

In [110]:

```
pipe2=Pipeline([
    ('tr1',tr1),
    ('tr2',tr2),
    ('tr3',tr3),
    ('tr4',tr4),
    ('tr5',tr5),
    ('tr6',tr6),
    ('tr7',tr7),
    ('tr8',tr8),
    ('tr10',tr10)
])
```

In [111]:

```
pipe2.fit(X_train,y_train)
```

Out[111]:

```
Pipeline(steps=[('tr1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('imputer2',
                                                  SimpleImputer(strategy='mo
st_frequent'),
                                                  [2]),
                                                 ('imputer7',
                                                  SimpleImputer(strategy='me
dian'),
                                                  [7])])),
                ('tr2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('le', OrdinalEncoder(),
                                                  [0])])),
                ('tr3',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('oh...
                ('tr6',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(dtype='int3
2',
                                                                sparse=Fals
e),
                                                  [47])])),
                ('tr7',
                 ColumnTransformer(transformers=[('scaler', MinMaxScaler(),
                                                  slice(0, 55, None))])),
                ('tr8',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('pt', PowerTransformer(),
                                                  slice(0, 55, None))])),
                ('tr10', DecisionTreeClassifier())])
```

In [112]:

```
y_pred2=pipe2.predict(X_test)
print('decision tree:')
print('confusion matrix:')
print(confusion_matrix(y_test,y_pred2))
print('classification report:')
print(classification_report(y_test,y_pred2))
```

```
decision tree:
confusion matrix:
[[9177  813]
 [ 737  204]]
classification report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92      9990
           1       0.20      0.22      0.21       941

    accuracy                           0.86     10931
   macro avg       0.56      0.57      0.57     10931
weighted avg       0.86      0.86      0.86     10931
```

In [113]:

```
tr11=KNeighborsClassifier()
```

In [114]:

```
pipe3=Pipeline([
    ('tr1',tr1),
    ('tr2',tr2),
    ('tr3',tr3),
    ('tr4',tr4),
    ('tr5',tr5),
    ('tr6',tr6),
    ('tr7',tr7),
    ('tr8',tr8),
    ('tr11',tr11)
])
```

In [115]:

```
pipe3.fit(X_train,y_train)
```

Out[115]:

```
Pipeline(steps=[('tr1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('imputer2',
                                                  SimpleImputer(strategy='mo
st_frequent'),
                                                  [2]),
                                                 ('imputer7',
                                                  SimpleImputer(strategy='me
dian'),
                                                  [7])])),
                ('tr2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('le', OrdinalEncoder(),
                                                  [0])])),
                ('tr3',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('oh...
                                                  [45])])),
                ('tr6',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(dtype='int3
2',
                                                                sparse=Fals
e),
                                                  [47])])),
                ('tr7',
                 ColumnTransformer(transformers=[('scaler', MinMaxScaler(),
                                                  slice(0, 55, None))])),
                ('tr8',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('pt', PowerTransformer(),
                                                  slice(0, 55, None))])),
                ('tr11', KNeighborsClassifier())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [116]:

```
y_pred3=pipe3.predict(X_test)
print('KNeighborsClassifer:')
print('confusion_matrix:')
print(confusion_matrix(y_test,y_pred3))
print('classification_report:')
print(classification_report(y_test,y_pred3))
```

```
KNeighborsClassifer:
confusion_matrix:
[[9784  206]
 [ 859   82]]
classification_report:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      9990
           1       0.28      0.09      0.13       941

    accuracy                           0.90     10931
   macro avg       0.60      0.53      0.54     10931
weighted avg       0.86      0.90      0.88     10931
```

In [117]:

```
tr12=GradientBoostingClassifier()
```

In [118]:

```
pipe4=Pipeline([
    ('tr1',tr1),
    ('tr2',tr2),
    ('tr3',tr3),
    ('tr4',tr4),
    ('tr5',tr5),
    ('tr6',tr6),
    ('tr7',tr7),
    ('tr8',tr8),
    ('tr12',tr12)
])
```

In [119]:

```
pipe4.fit(X_train,y_train)
```

Out[119]:

```
Pipeline(steps=[('tr1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('imputer2',
                                                  SimpleImputer(strategy='mo
st_frequent'),
                                                  [2]),
                                                 ('imputer7',
                                                  SimpleImputer(strategy='me
dian'),
                                                  [7])])),
                ('tr2',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('le', OrdinalEncoder(),
                                                  [0])])),
                ('tr3',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('oh...
                ('tr6',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(dtype='int3
2',
                                                                sparse=Fals
e),
                                                  [47])])),
                ('tr7',
                 ColumnTransformer(transformers=[('scaler', MinMaxScaler(),
                                                  slice(0, 55, None))])),
                ('tr8',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('pt', PowerTransformer(),
                                                  slice(0, 55, None))])),
                ('tr12', GradientBoostingClassifier())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org

In [120]:

```
y_pred4=pipe4.predict(X_test)
print('GradientBoosterClassifier:')
print('confusion_matrix:')
print(confusion_matrix(y_test,y_pred4))
print('classification_report')
print(classification_report(y_test,y_pred4))
```

```
GradientBoosterClassifier:
confusion_matrix:
[[9954   36]
 [ 889   52]]
classification_report
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      9990
           1       0.59      0.06      0.10       941

    accuracy                           0.92     10931
   macro avg       0.75      0.53      0.53     10931
weighted avg       0.89      0.92      0.88     10931
```

In [121]:

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score
```

In [122]:

```
y_pred1=pipe1.predict(X_test)
```

In [123]:

```
y_pred2=pipe2.predict(X_test)
```

In [124]:

```
y_pred3=pipe3.predict(X_test)
```

In [125]:

```
y_pred4=pipe4.predict(X_test)
```

In [126]:

```
print('rf:',y_pred1,'dt:',y_pred2,'knc:',y_pred3,'gbc:',y_pred4)
```

```
rf: [0 0 0 ... 0 0 0] dt: [0 0 0 ... 0 0 0] knc: [0 0 0 ... 0 0 0] gbc: [0 0
0 ... 0 0 0]
```

In [127]:

```
score1=f1_score(y_pred1,y_test,average='weighted')
```

In [128]:

```
score2=f1_score(y_pred2,y_test,average='weighted')
```

In [129]:

```
score3=f1_score(y_pred3,y_test,average='weighted')
```

In [130]:

```
score4=f1_score(y_pred4,y_test,average='weighted')
```

In [131]:

```
print('rf:',score1,'dt:',score2,'knc:',score3,'gbc:',score4)
```

```
rf: 0.9297813505142158 dt: 0.8557201983481194 knc: 0.9269123144274146 gbc:
0.948719894696188
```

In [132]:

```
cv1=cross_val_score(pipe1,X_train,y_train,cv=5)
```

In [133]:

```
cv2=cross_val_score(pipe2,X_train,y_train,cv=5)
```

In [134]:

```
cv3=cross_val_score(pipe3,X_train,y_train,cv=5)
```

In [135]:

```
cv4=cross_val_score(pipe4,X_train,y_train,cv=5)
```

In [136]:

```
print('rf:',np.mean(cv1),'dt:',np.mean(cv2),'knc:',np.mean(cv3),'gbc:',np.mean(cv4))
```

```
rf: 0.9074586575939094 dt: 0.856544832571952 knc: 0.9084878720398063 gbc: 0.
9169964198598388
```

In [137]:

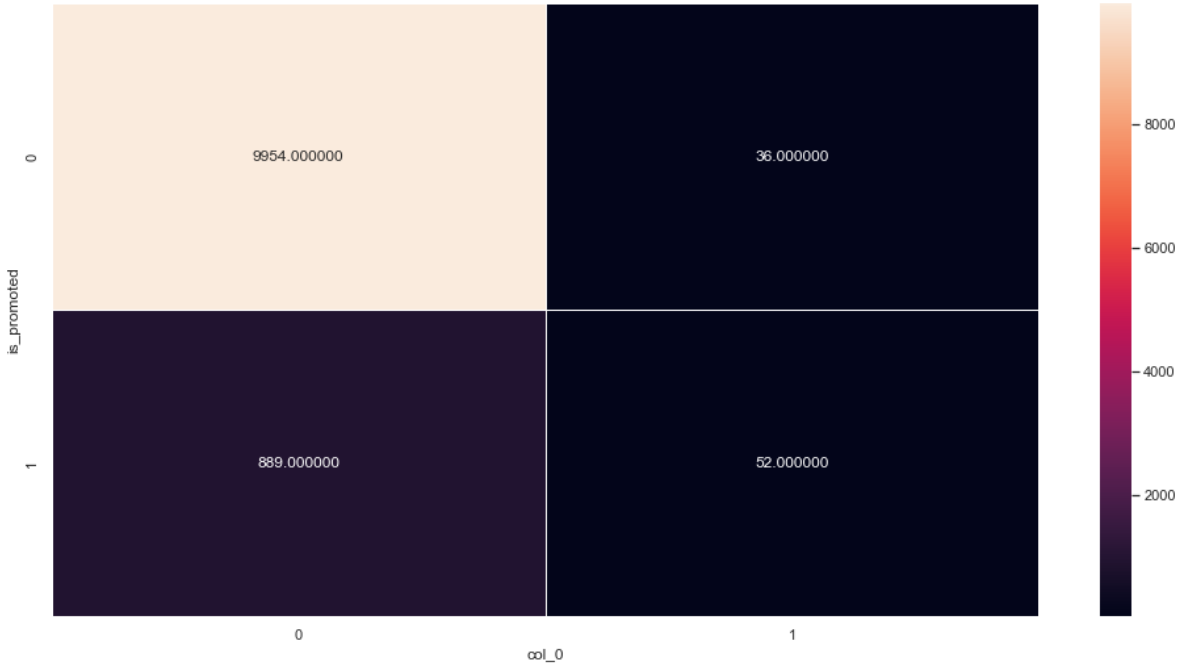
```
pd.crosstab(y_test,y_pred4)
```

Out[137]:

col_0	0	1
is_promoted		
0	9954	36
1	889	52

In [138]:

```
sns.heatmap(pd.crosstab(y_test,y_pred4),annot=True,cbar=True,linewidth=True,fmt='f')
plt.show()
```



In [139]:

```
import joblib
```

In [140]:

```
joblib.dump(pipe4,'model.pkl')
```

Out[140]:

```
['model.pkl']
```