Predicting Employee Promotion

HR Analytics--Employee Promotion Data

INTRODUCTION

There are many advantages that comes with promotion in the workplace. It comes with increase in prestige, respect and wages/salary. Promotion in a business environment also comes with increase freedom and agency. It is because of this that many employees at different levels in an organisation tries to climb the corporate ladder in order to enjoy the various advantages that comes with it.

Organisations also pay much attention to who is going to get promoted and how that will affect the business. This is because the activities of those in key positions in a business usually affects the welfare of the organisation in general. Firm and businesses pay attention to many factors when making decision on who to promote. For this reason resources are expended by businesses in the process of selection of employee to promote. Employees in organizations also make effort to signal to their employers that they are the best fit for promotion. This involves on working to improve various factors that can lead them to be viable for promotion.

This project is important for both employers looking to start training in arears to promotion of employees. It is also important to employees to determine their strength and weaknesses, oppurtunities and threts, as well as, key factors to work on in order to get promoted. Although this is a case study of a given firm, the findings of this study can be applied to various firms.

RESEARCH QUESTIONS

1. What are the most important features and the influence these features have in predicting employee promotion with a particular row of data?
2. Which features are the strongest in predicting employee promotion?
3. Does the contribution of each feature vary by gender?

In [ ]:

In [1]:
```
# importing the liabrary

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
train_data = pd.read_csv(r'C:\Users\Mukul\Downloads\train.csv')
```

In [3]:
```
test_data = pd.read_csv(r'C:\Users\Mukul\Downloads\test.csv')
```

In [4]:
```
# check the top 5 dataset

train_data.head()
```

Out[4]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previ
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	
4	48945	Technology	region_26	Bachelor's	m	other	1	45	

In [5]:
```
# check the last 5 dataset

train_data.tail()
```

Out[5]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age
54803	3030	Technology	region_14	Bachelor's	m	sourcing	1	48
54804	74592	Operations	region_27	Master's & above	f	other	1	37
54805	13918	Analytics	region_1	Bachelor's	m	other	1	27
54806	13614	Sales & Marketing	region_9	NaN	m	sourcing	1	29
54807	51526	HR	region_22	Bachelor's	m	other	1	27

In [6]:
```
# check the shape

train_data.shape
```

Out[6]:
```
(54808, 13)
```

In [7]:
```
# check the test dataset

test_data.head()
```

Out[7]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	prev
0	8724	Technology	region_26	Bachelor's	m	sourcing	1	24	
1	74430	HR	region_4	Bachelor's	f	other	1	31	
2	72255	Sales & Marketing	region_13	Bachelor's	m	other	1	31	
3	38562	Procurement	region_2	Bachelor's	f	other	3	31	
4	64486	Finance	region_29	Bachelor's	m	sourcing	1	30	

In [8]:
```
# check the shape of test_dataset

test_data.shape
```

Out[8]:
```
(23490, 12)
```

- Here is indicated the TARGET Columns are left

In [9]:
```
# check the columns

train_data.columns
```

Out[9]:
```
Index(['employee_id', 'department', 'region', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'awards_won?', 'avg_training_score',
       'is_promoted'],
      dtype='object')
```

Provided the information of dataset

- employee_id: Unique ID for employee.
- department: Department of employee.
- region: Region of employment (unordered).
- education: Education Level.
- gender: Gender of Employee.
- recruitment_channel: Channel of recruitment for employee.
- no_of_trainings: no of other trainings completed in previous year on soft skills, technical skills etc.
- age: Age of Employee
- previous_ year_rating: Employee Rating for the previous year.
- length_ of_ service: Length of service in years.
- awards_ won?: if awards won during previous year then 1 else 0.
- avg_ training_ score: Average score in current training evaluations.
- is_promoted: (Target) Recommended for promotion.

The columns in the test and train data are the same except that is_promoted is not included in the test

In [10]:
```
# check the info() of dataset

train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   employee_id           54808 non-null  int64  
 1   department            54808 non-null  object 
 2   region                54808 non-null  object 
 3   education             52399 non-null  object 
 4   gender                54808 non-null  object 
 5   recruitment_channel   54808 non-null  object 
 6   no_of_trainings       54808 non-null  int64  
 7   age                   54808 non-null  int64  
 8   previous_year_rating  50684 non-null  float64
 9   length_of_service     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
```

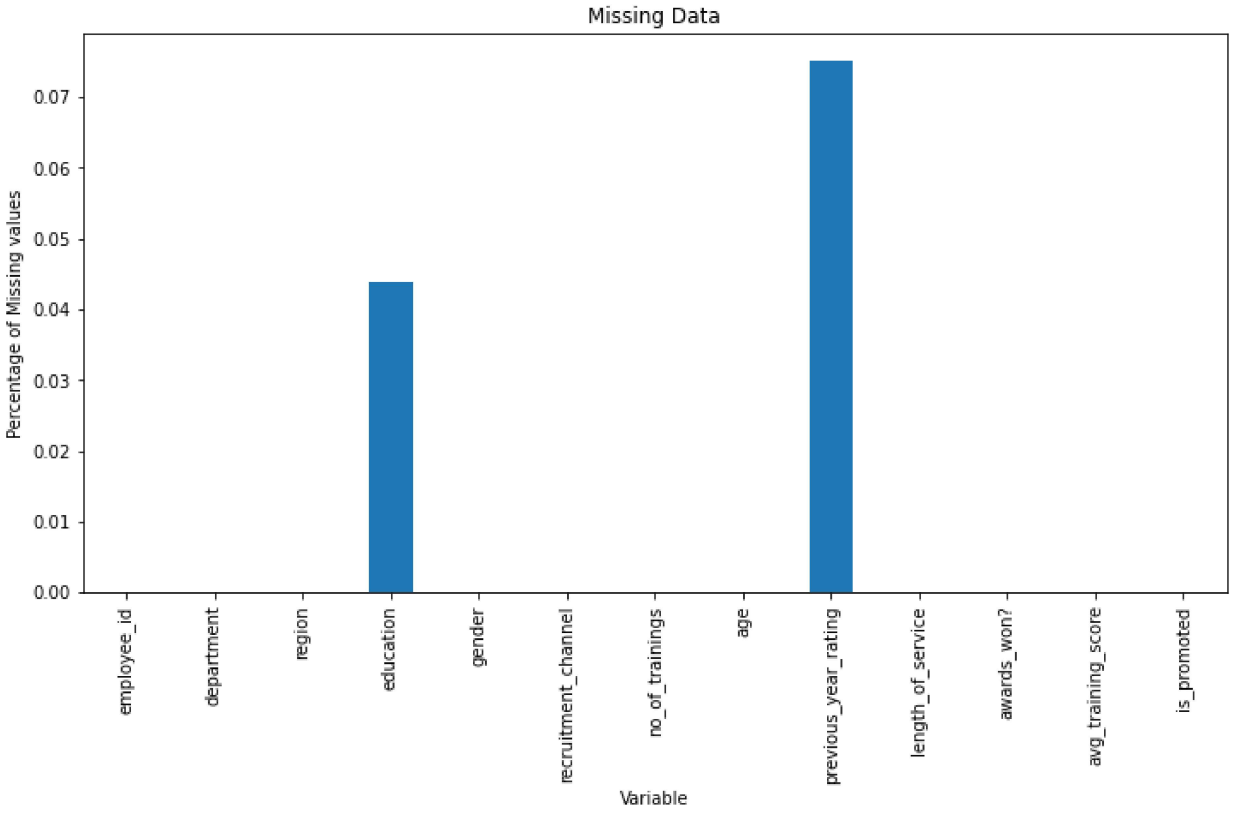
In [ ]:
```

```

In [11]:
```
# check the null value
train_data.isnull().sum()
```

Out[11]:
```
employee_id              0
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
awards_won?              0
avg_training_score       0
is_promoted              0
dtype: int64
```

In [12]:
```
# indicated the null values
train_data.isnull().mean().plot.bar(figsize=(12,6))
plt.ylabel('Percentage of Missing values')
plt.xlabel('Variable')
plt.title('Missing Data')
```

Out[12]:
```
Text(0.5, 1.0, 'Missing Data')
```



In [13]:
```
# check the % of missing  values

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

Out[13]:
```
employee_id           0.000000
department            0.000000
region                0.000000
education             4.395344
gender                0.000000
recruitment_channel   0.000000
no_of_trainings       0.000000
age                   0.000000
previous_year_rating  7.524449
length_of_service     0.000000
awards_won?           0.000000
avg_training_score    0.000000
is_promoted           0.000000
dtype: float64
```

In [14]:
```
# check the total missing value

train_data.isnull().sum().sum()
```

Out[14]:
```
6533
```

Here are indicated the two columns of missing values 1. EDUCATION and 2. PREVIOUS_YEAR_RATING

In [15]:
```
train_data['education'].unique()
```

Out[15]:
```
array(["Master's & above", "Bachelor's", nan, 'Below Secondary'],
      dtype=object)
```

In [16]:
```
train_data['education'].value_counts()
```

Out[16]:
```
Bachelor's          36669
Master's & above    14925
Below Secondary       805
Name: education, dtype: int64
```

In [17]:
```
# filling the missing values

train_data['education'] = train_data['education'].fillna(train_data['education'].mode(

train_data['previous_year_rating'] = train_data['previous_year_rating'].fillna(train_d
```

In [18]:
```
# Filling the missing values after that check the missing values

train_data.isnull().sum()
```

Out[18]:
```
employee_id             0
department              0
region                  0
education               0
gender                  0
recruitment_channel     0
no_of_trainings         0
age                     0
previous_year_rating    0
length_of_service       0
awards_won?             0
avg_training_score      0
is_promoted             0
dtype: int64
```

- Here are indicated the no missing values

In [19]:
```
# check the unique values

train_data.nunique()
```

Out[19]:
```
employee_id           54808
department                9
region                   34
education                 3
gender                    2
recruitment_channel       3
no_of_trainings          10
age                      41
previous_year_rating      5
length_of_service        35
awards_won?               2
avg_training_score       61
is_promoted               2
dtype: int64
```

In [ ]:
```

```

In [20]:
```
# check the duplicated values
train_data.duplicated().sum()
```

Out[20]:
```
0
```

In [21]:
```
# check the sample of dataset

train_data.sample()
```

Out[21]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age
16371	36056	Sales & Marketing	region_24	Master's & above	m	other	1	39

In [22]:
```
train_data.columns
```

Out[22]:
```
Index(['employee_id', 'department', 'region', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'awards_won?', 'avg_training_score',
       'is_promoted'],
      dtype='object')
```

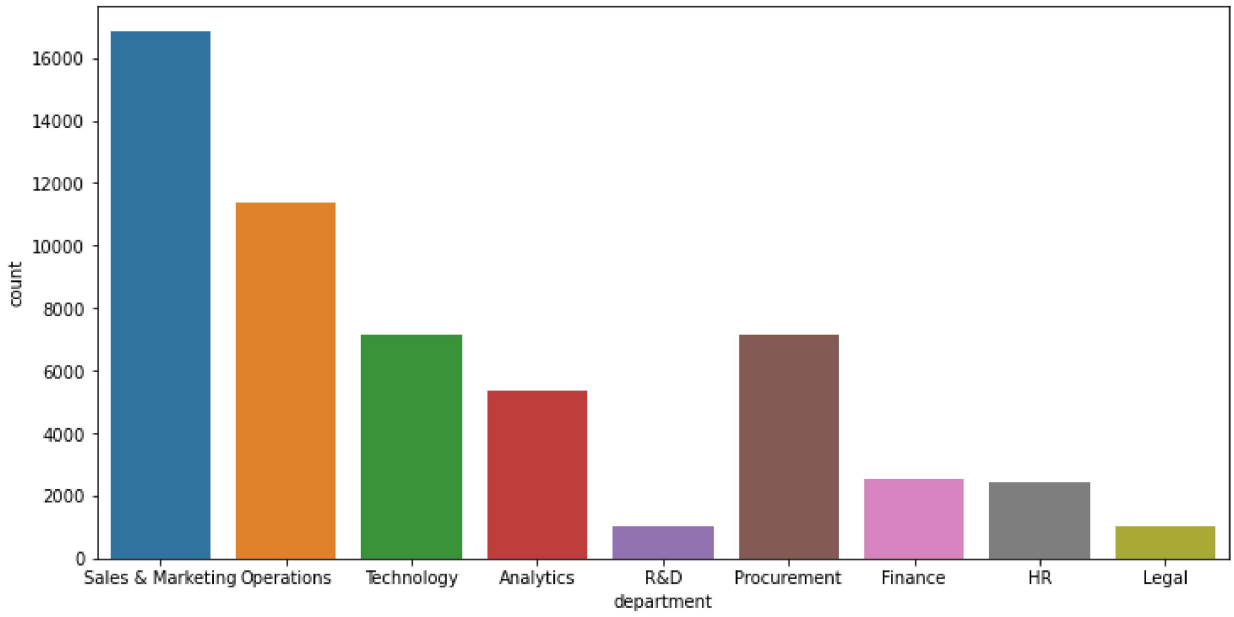
In [23]:
```
# check the department of unique categories and value_counts
train_data['department'].unique()
```

Out[23]:
```
array(['Sales & Marketing', 'Operations', 'Technology', 'Analytics',
       'R&D', 'Procurement', 'Finance', 'HR', 'Legal'], dtype=object)
```

In [24]:
```
train_data['department'].value_counts()
```

Out[24]:
```
Sales & Marketing    16840
Operations           11348
Technology            7138
Procurement           7138
Analytics             5352
Finance               2536
HR                    2418
Legal                 1039
R&D                    999
Name: department, dtype: int64
```

In [25]:
```
plt.figure(figsize=(12,6))
sns.countplot(train_data['department'])
plt.show()
```



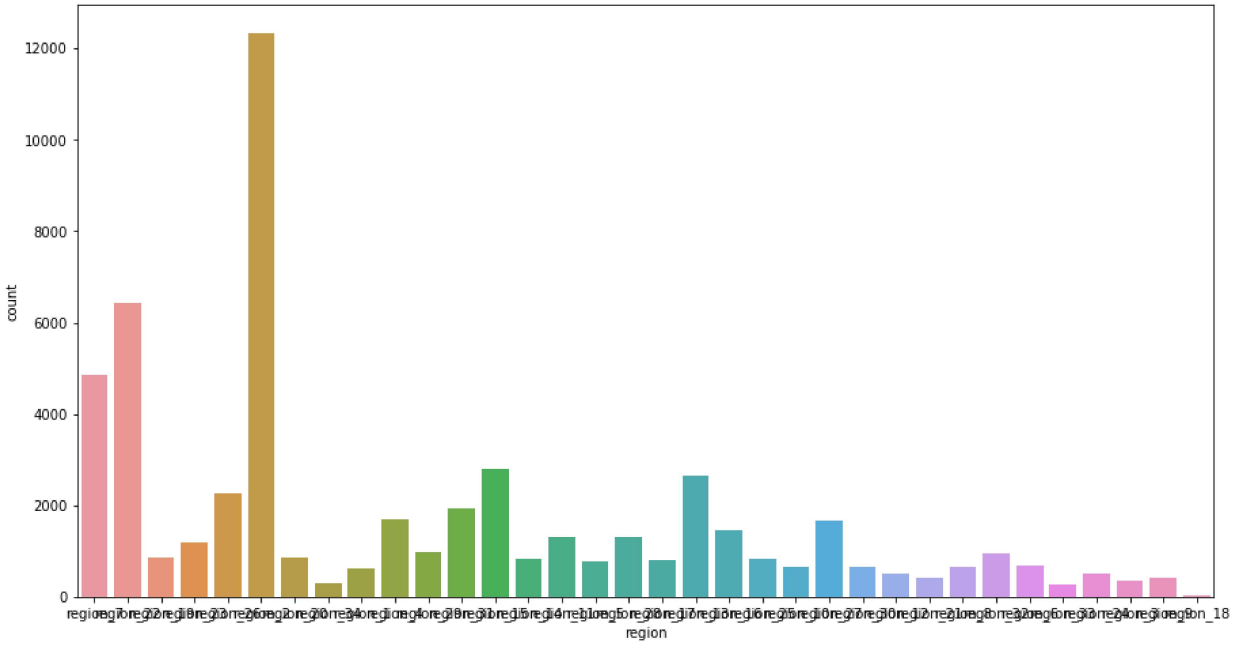
In [26]:
```
# check the region of unique categories and value_counts
train_data['region'].unique()
```

Out[26]:
```
array(['region_7', 'region_22', 'region_19', 'region_23', 'region_26',
       'region_2', 'region_20', 'region_34', 'region_1', 'region_4',
       'region_29', 'region_31', 'region_15', 'region_14', 'region_11',
       'region_5', 'region_28', 'region_17', 'region_13', 'region_16',
       'region_25', 'region_10', 'region_27', 'region_30', 'region_12',
       'region_21', 'region_8', 'region_32', 'region_6', 'region_33',
       'region_24', 'region_3', 'region_9', 'region_18'], dtype=object)
```

In [27]:
```
train_data['region'].value_counts()
```

Out[27]:
```
region_2     12343
region_22     6428
region_7      4843
region_15     2808
region_13     2648
region_26     2260
region_31     1935
region_4      1703
region_27     1659
region_16     1465
region_28     1318
region_11     1315
region_23     1175
region_29      994
region_32      945
region_19      874
region_20      850
region_14      827
region_25      819
region_17      796
region_5       766
region_6       690
region_30      657
region_8       655
region_10      648
region_1       610
region_24      508
region_12      500
region_9       420
region_21      411
region_3       346
region_34      292
region_33      269
region_18       31
Name: region, dtype: int64
```

In [28]:
```
plt.figure(figsize=(15,8))
sns.countplot(train_data['region'])
plt.show()
```



Segregate the Numerical and Categorical Columns

In [29]:
```
[feature for feature in train_data.columns ]
```

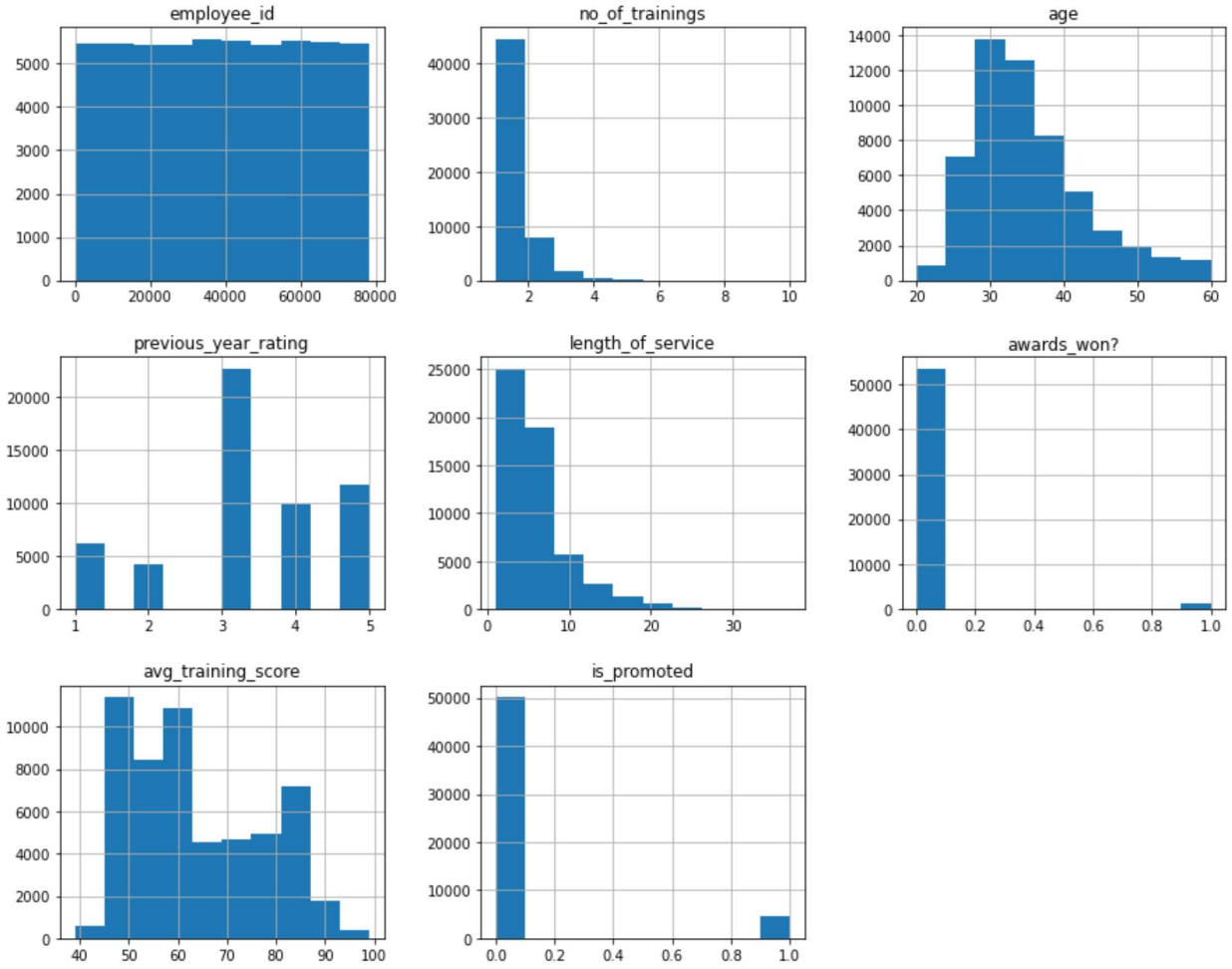
Out[29]:
```
['employee_id',
 'department',
 'region',
 'education',
 'gender',
 'recruitment_channel',
 'no_of_trainings',
 'age',
 'previous_year_rating',
 'length_of_service',
 'awards_won?',
 'avg_training_score',
 'is_promoted']
```

In [30]:
```
# Numerical Columns

num_columns=[feature for feature in train_data.columns if train_data[feature].dtypes!=
num_columns
```

Out[30]:
```
['employee_id',
 'no_of_trainings',
 'age',
 'previous_year_rating',
 'length_of_service',
 'awards_won?',
 'avg_training_score',
 'is_promoted']
```

In [31]:
```
# indiacte the num_columns

train_data.hist(figsize=(15,12))
```

Out[31]:
```
array([[<AxesSubplot:title={'center':'employee_id'}>,
        <AxesSubplot:title={'center':'no_of_trainings'}>,
        <AxesSubplot:title={'center':'age'}>],
       [<AxesSubplot:title={'center':'previous_year_rating'}>,
        <AxesSubplot:title={'center':'length_of_service'}>,
        <AxesSubplot:title={'center':'awards_won?'}>],
       [<AxesSubplot:title={'center':'avg_training_score'}>,
        <AxesSubplot:title={'center':'is_promoted'}>, <AxesSubplot:>]],
      dtype=object)
```



In [32]:
```
# Categorical Columns
cat_columns = [feature for feature in train_data.columns if train_data[feature].dtypes
cat_columns
```

Out[32]:
```
['department', 'region', 'education', 'gender', 'recruitment_channel']
```

In [ ]:
```

```

In [33]:
```
# Getting count of each Category from data
for feature in train_data.columns:
    print(train_data[feature].value_counts)
```

```
<bound method IndexOpsMixin.value_counts of 0        65438
1        65141
2         7513
3         2542
4        48945
         ...
54803     3030
54804    74592
54805    13918
54806    13614
54807    51526
Name: employee_id, Length: 54808, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0        Sales & Marketing
1               Operations
2        Sales & Marketing
3        Sales & Marketing
4               Technology
               ...
54803           Technology
```

In [34]:
```
# Getting count of each Category from  unique values
for feature in train_data.columns:
    print(train_data[feature].unique())
    plt.show()
```

```
[65438 65141  7513 ... 13918 13614 51526]
['Sales & Marketing' 'Operations' 'Technology' 'Analytics' 'R&D'
 'Procurement' 'Finance' 'HR' 'Legal']
['region_7' 'region_22' 'region_19' 'region_23' 'region_26' 'region_2'
 'region_20' 'region_34' 'region_1' 'region_4' 'region_29' 'region_31'
 'region_15' 'region_14' 'region_11' 'region_5' 'region_28' 'region_17'
 'region_13' 'region_16' 'region_25' 'region_10' 'region_27' 'region_30'
 'region_12' 'region_21' 'region_8' 'region_32' 'region_6' 'region_33'
 'region_24' 'region_3' 'region_9' 'region_18']
["Master's & above" "Bachelor's" 'Below Secondary']
['f' 'm']
['sourcing' 'other' 'referred']
[ 1  2  3  4  7  5  6  8 10  9]
[35 30 34 39 45 31 33 28 32 49 37 38 41 27 29 26 24 57 40 42 23 59 44 50
 56 20 25 47 36 46 60 43 22 54 58 48 53 55 51 52 21]
[5. 3. 1. 4. 2.]
[ 8  4  7 10  2  5  6  1  3 16  9 11 26 12 17 14 13 19 15 23 18 20 22 25
 28 24 31 21 29 30 34 27 33 32 37]
[0 1]
[49 60 50 73 85 59 63 83 54 77 80 84 51 46 75 57 70 68 79 44 72 61 48 58
 87 47 52 88 71 65 62 53 78 91 82 69 55 74 86 90 92 67 89 56 76 81 45 64
 39 94 93 66 95 42 96 40 99 43 97 41 98]
[0 1]
```

Descriptive Statistics

In [35]:
```
# indicated the numerical_columns of descriptive statics
train_data.describe()
```

Out[35]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?
count	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000
mean	39195.830627	1.253011	34.803915	3.304481	5.865512	0.023172
std	22586.581449	0.609264	7.660169	1.214770	4.265094	0.150450
min	1.000000	1.000000	20.000000	1.000000	1.000000	0.000000
25%	19669.750000	1.000000	29.000000	3.000000	3.000000	0.000000
50%	39225.500000	1.000000	33.000000	3.000000	5.000000	0.000000
75%	58730.500000	1.000000	39.000000	4.000000	7.000000	0.000000
max	78298.000000	10.000000	60.000000	5.000000	37.000000	1.000000

In [36]:
```
# indiacte the only categorical columns of statics

train_data.describe(include='object')
```

Out[36]:

	department	region	education	gender	recruitment_channel
count	54808	54808	54808	54808	54808
unique	9	34	3	2	3
top	Sales & Marketing	region_2	Bachelor's	m	other
freq	16840	12343	39078	38496	30446

In [37]:
```
# check the transpose statistics

train_data.describe().T
```

Out[37]:

	count	mean	std	min	25%	50%	75%	max
employee_id	54808.0	39195.830627	22586.581449	1.0	19669.75	39225.5	58730.5	78298.0
no_of_trainings	54808.0	1.253011	0.609264	1.0	1.00	1.0	1.0	10.0
age	54808.0	34.803915	7.660169	20.0	29.00	33.0	39.0	60.0
previous_year_rating	54808.0	3.304481	1.214770	1.0	3.00	3.0	4.0	5.0
length_of_service	54808.0	5.865512	4.265094	1.0	3.00	5.0	7.0	37.0
awards_won?	54808.0	0.023172	0.150450	0.0	0.00	0.0	0.0	1.0
avg_training_score	54808.0	63.386750	13.371559	39.0	51.00	60.0	76.0	99.0
is_promoted	54808.0	0.085170	0.279137	0.0	0.00	0.0	0.0	1.0

In [38]:
```
# check the test_data of statistics  statistics
test_data.describe()
```

Out[38]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?
count	23490.000000	23490.000000	23490.000000	21678.000000	23490.000000	23490.000000
mean	39041.399149	1.254236	34.782929	3.339146	5.810387	0.022776
std	22640.809201	0.600910	7.679492	1.263294	4.207917	0.149191
min	3.000000	1.000000	20.000000	1.000000	1.000000	0.000000
25%	19370.250000	1.000000	29.000000	3.000000	3.000000	0.000000
50%	38963.500000	1.000000	33.000000	3.000000	5.000000	0.000000
75%	58690.000000	1.000000	39.000000	4.000000	7.000000	0.000000
max	78295.000000	9.000000	60.000000	5.000000	34.000000	1.000000

In [39]:
```
# check correlation
train_data.corr()
```

Out[39]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	awar
employee_id	1.000000	-0.005121	0.000437	0.004209	0.001274	
no_of_trainings	-0.005121	1.000000	-0.081278	-0.061564	-0.057275	
age	0.000437	-0.081278	1.000000	0.026810	0.657111	
previous_year_rating	0.004209	-0.061564	0.026810	1.000000	0.023504	
length_of_service	0.001274	-0.057275	0.657111	0.023504	1.000000	
awards_won?	0.008420	-0.007628	-0.008169	0.026587	-0.039927	
avg_training_score	-0.000586	0.042517	-0.048380	0.071926	-0.038122	
is_promoted	0.001206	-0.024896	-0.017166	0.153230	-0.010670	

In [40]:
```
# check the covarience
train_data.cov()
```

Out[40]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	aw
employee_id	5.101537e+08	-70.469854	75.643745	115.483177	122.743200	
no_of_trainings	-7.046985e+01	0.371203	-0.379329	-0.045564	-0.148832	
age	7.564375e+01	-0.379329	58.678192	0.249480	21.468711	
previous_year_rating	1.154832e+02	-0.045564	0.249480	1.475666	0.121779	
length_of_service	1.227432e+02	-0.148832	21.468711	0.121779	18.191028	
awards_won?	2.861224e+01	-0.000699	-0.009414	0.004859	-0.025621	
avg_training_score	-1.770908e+02	0.346378	-4.955455	1.168316	-2.174143	
is_promoted	7.602562e+00	-0.004234	-0.036705	0.051958	-0.012703	

In [41]:
```
# check skewness

train_data.skew()
```

Out[41]:
```
employee_id           -0.003128
no_of_trainings        3.445434
age                    1.007432
previous_year_rating  -0.260858
length_of_service      1.738061
awards_won?            6.338914
avg_training_score     0.451908
is_promoted            2.972339
dtype: float64
```

In [42]:
```
# check the Standard Deviation

train_data.std()
```

Out[42]:
```
employee_id             22586.581449
no_of_trainings             0.609264
age                         7.660169
previous_year_rating        1.214770
length_of_service           4.265094
awards_won?                 0.150450
avg_training_score         13.371559
is_promoted                 0.279137
dtype: float64
```

In [43]:
```
# check mean()
train_data.mean()
```

Out[43]:
```
employee_id             39195.830627
no_of_trainings             1.253011
age                        34.803915
previous_year_rating        3.304481
length_of_service           5.865512
awards_won?                 0.023172
avg_training_score         63.386750
is_promoted                 0.085170
dtype: float64
```

In [44]:
```
# check mdeian()
train_data.median()
```

Out[44]:
```
employee_id             39225.5
no_of_trainings             1.0
age                        33.0
previous_year_rating        3.0
length_of_service           5.0
awards_won?                 0.0
avg_training_score         60.0
is_promoted                 0.0
dtype: float64
```

In [45]:
```
# check minimum values
train_data.min()
```

Out[45]:
```
employee_id                   1
department            Analytics
region                 region_1
education           Bachelor's
gender                         f
recruitment_channel        other
no_of_trainings                1
age                           20
previous_year_rating         1.0
length_of_service              1
awards_won?                    0
avg_training_score            39
is_promoted                    0
dtype: object
```

In [46]:
```
# check  maxmimum values
train_data.max()
```

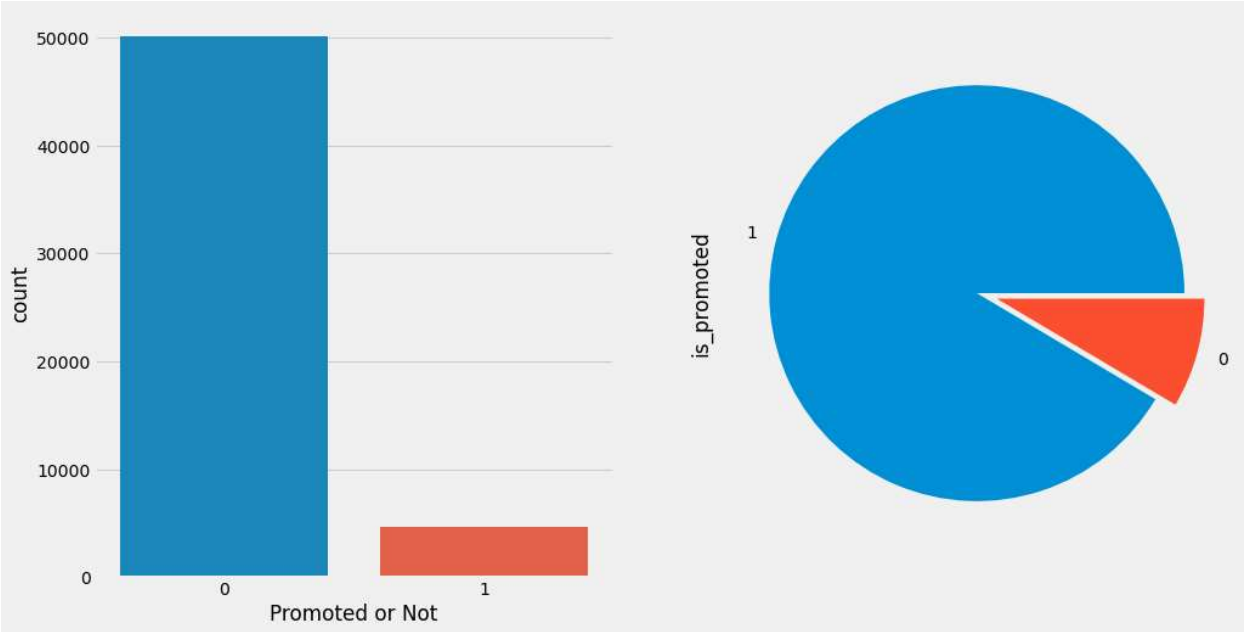
Out[46]:
```
employee_id                   78298
department               Technology
region                     region_9
education           Master's & above
gender                           m
recruitment_channel       sourcing
no_of_trainings                 10
age                             60
previous_year_rating           5.0
length_of_service               37
awards_won?                      1
avg_training_score              99
is_promoted                      1
dtype: object
```

In [47]:
```
# check the quantile values
train_data.quantile()
```

Out[47]:
```
employee_id           39225.5
no_of_trainings           1.0
age                      33.0
previous_year_rating      3.0
length_of_service         5.0
awards_won?               0.0
avg_training_score       60.0
is_promoted               0.0
Name: 0.5, dtype: float64
```

Target Columns

In [48]:
```
plt.rcParams['figure.figsize']=(15,8)
plt.style.use('fivethirtyeight')

plt.subplot(1,2,1)
sns.countplot(train_data['is_promoted'])
plt.xlabel('Promoted or Not')

plt.subplot(1,2,2)
train_data['is_promoted'].value_counts().plot(kind='pie',explode=[0,0.1],labels=['1','(
plt.show()
```



In [49]:
```
train_data['is_promoted'].unique()
```

Out[49]:
```
array([0, 1], dtype=int64)
```

In [50]:
```
train_data['is_promoted'].value_counts()
```

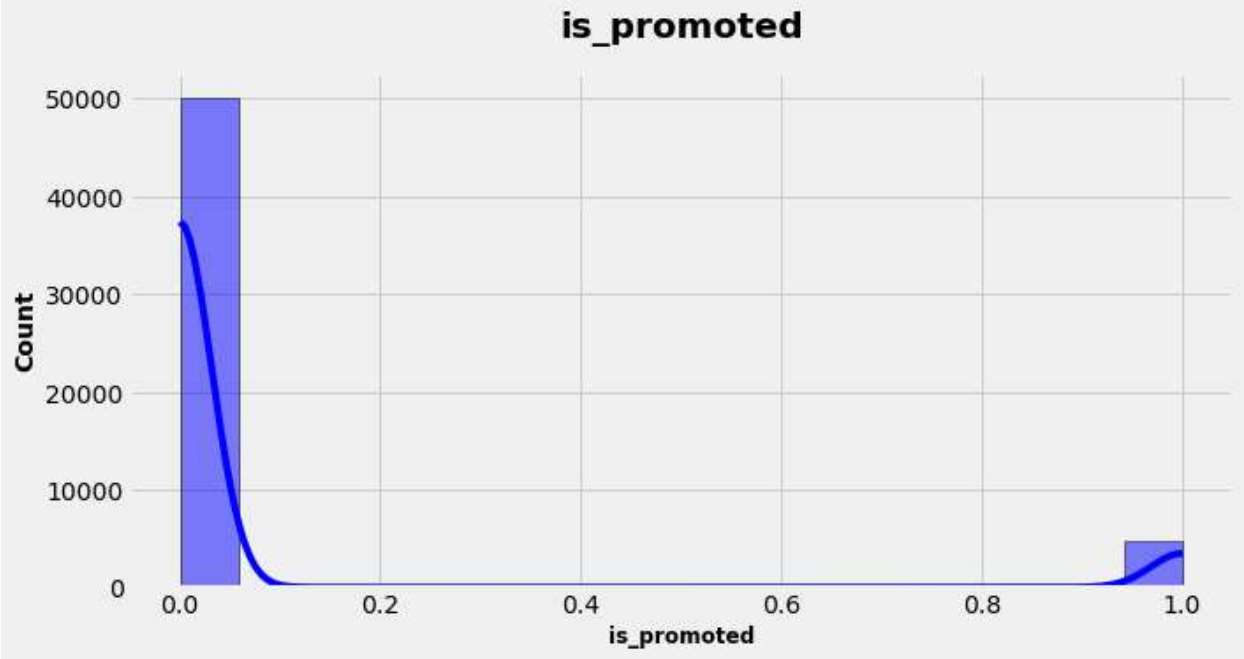
Out[50]:
```
0    50140
1     4668
Name: is_promoted, dtype: int64
```

"is_promoted" data is represent the imblance data

we need to handle imblance data

Visualization of Traget Variable

In [51]:
```
plt.subplots(figsize=(10,5))
sns.histplot(train_data['is_promoted'],ec='Black',color='blue',kde=True)
plt.title('is_promoted', weight='bold',fontsize=20,pad=20)
plt.ylabel('Count',weight='bold',fontsize=14)
plt.xlabel('is_promoted',weight='bold',fontsize=12)
plt.show()
```



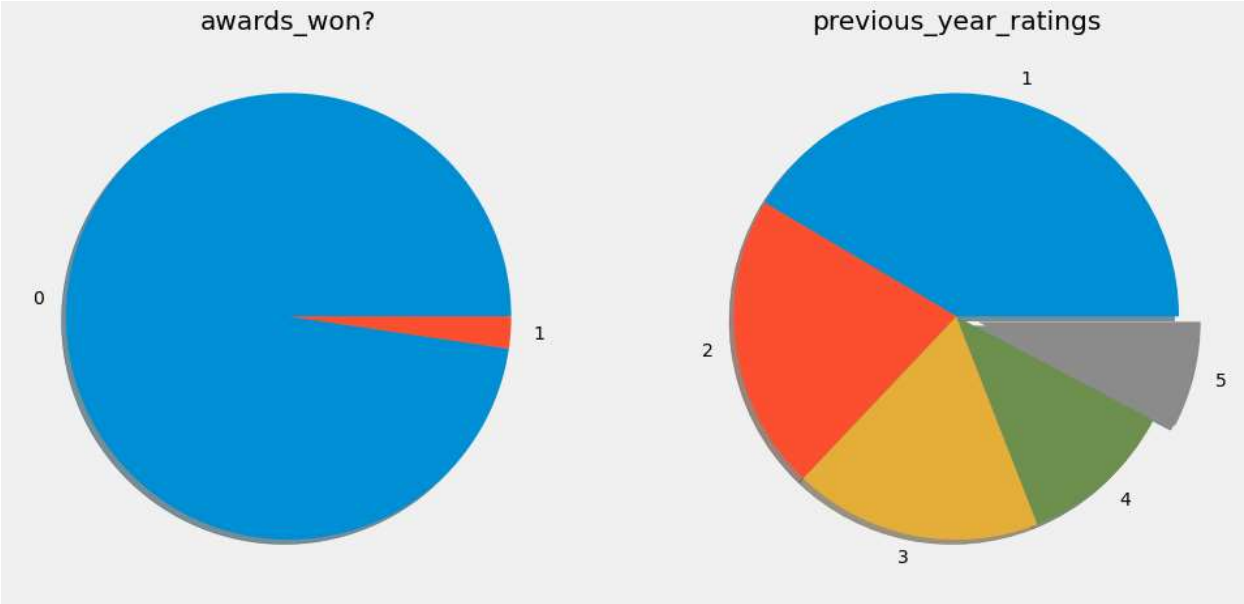
In [ ]:
```
```

- Univariate Analysis

- Bivariate Analysis
- Multivariate Analysis
- Feature Engineering
- Module Building and Prediction

In [52]:
```
# award_won? , previous_year_rating

# frist_plot :--> award_won?
plt.subplot(1,2,1)
labels=['0','1']
explode=[0,0]
size=train_data['awards_won?'].value_counts()
plt.pie(size,labels=labels , explode=explode,shadow=True)
plt.title('awards_won?')

# second_plot :---> previous_year_reating

plt.subplot(1,2,2)
labels=['1','2','3','4','5']

explode = [0,0,0,0,0.1]
size=train_data['previous_year_rating'].value_counts()
plt.pie(size,labels=labels ,explode=explode,shadow=True)
plt.title('previous_year_ratings')
```

Out[52]:
```
Text(0.5, 1.0, 'previous_year_ratings')
```



In [53]:
```
train_data.columns
```

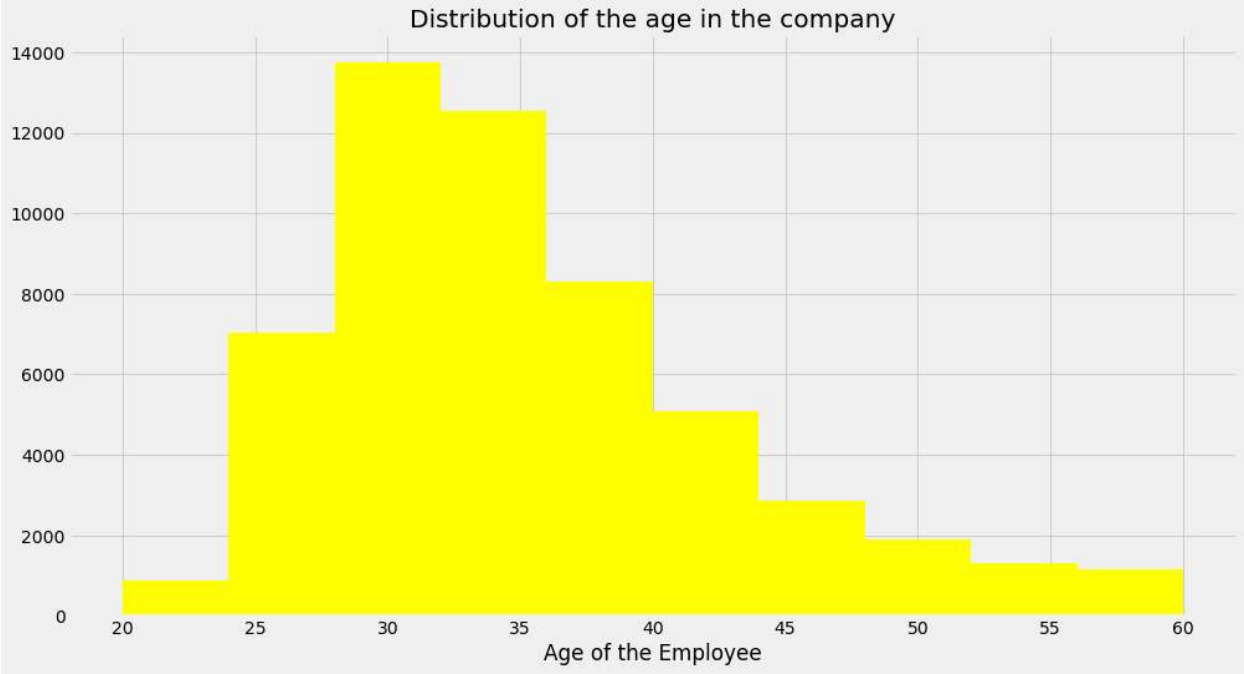
Out[53]:
```
Index(['employee_id', 'department', 'region', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'awards_won?', 'avg_training_score',
       'is_promoted'],
      dtype='object')
```

In [54]:
```
# indicated the no_of_training in countplot


sns.countplot(train_data['no_of_trainings'])
plt.xlabel('counts')
plt.title('Distribution ot the training data taken buy our employee')
plt.show()
```

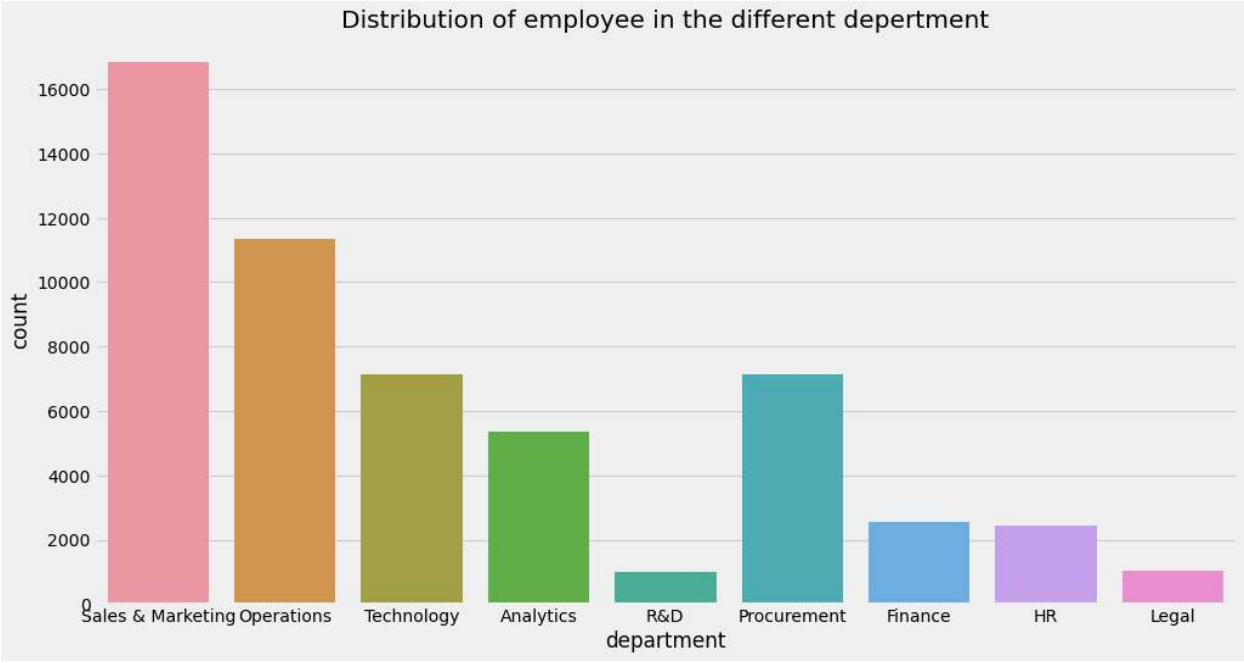
In [55]:
```
train_data['no_of_trainings'].value_counts()
```

Out[55]:
```
1     44378
2      7987
3      1776
4       468
5       128
6        44
7        12
8         5
10        5
9         5
Name: no_of_trainings, dtype: int64
```

In [56]:
```
# histogram of age data
plt.hist(train_data['age'],color='yellow')
plt.title('Distribution of the age in the company')
plt.xlabel('Age of the Employee')
plt.show()
```

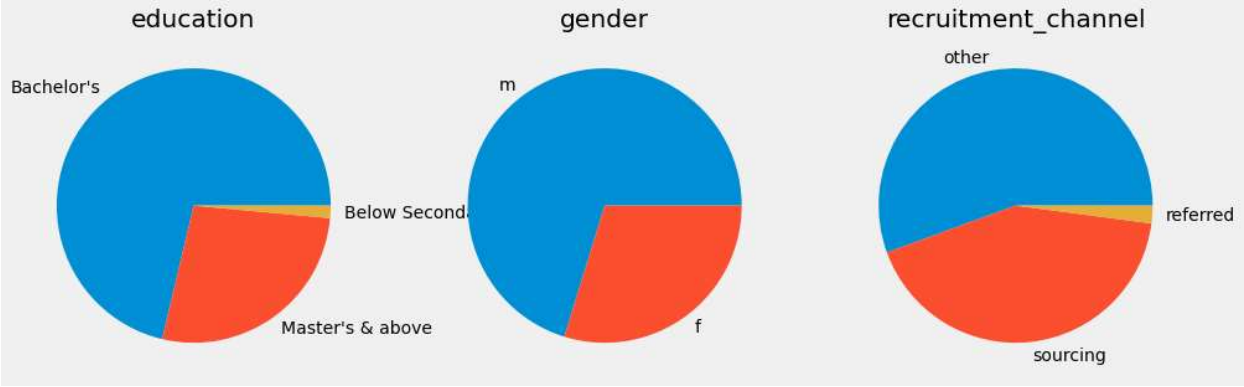


In [58]:
```
sns.countplot(train_data['department'])
plt.title('Distribution of employee in the different depertment')
plt.show()
```



In [63]:
```
# education , gender , requirements_channel

# education
plt.subplot(1,3,1)
labels = train_data['education'].value_counts().index  # index is used for categorical_
size=train_data['education'].value_counts()
explode=None
plt.pie(size,labels=labels,explode=explode)
plt.title('education')

# gender
plt.subplot(1,3,2)
labels=train_data['gender'].value_counts().index
size=train_data['gender'].value_counts()
explode=None
plt.pie(size,labels=labels,explode=explode)
plt.title('gender')

#requirements_channel
plt.subplot(1,3,3)
labels=train_data['recruitment_channel'].value_counts().index
size=train_data['recruitment_channel'].value_counts()
explode=None
plt.pie(size,labels=labels,explode=explode)
plt.title('recruitment_channel')
```

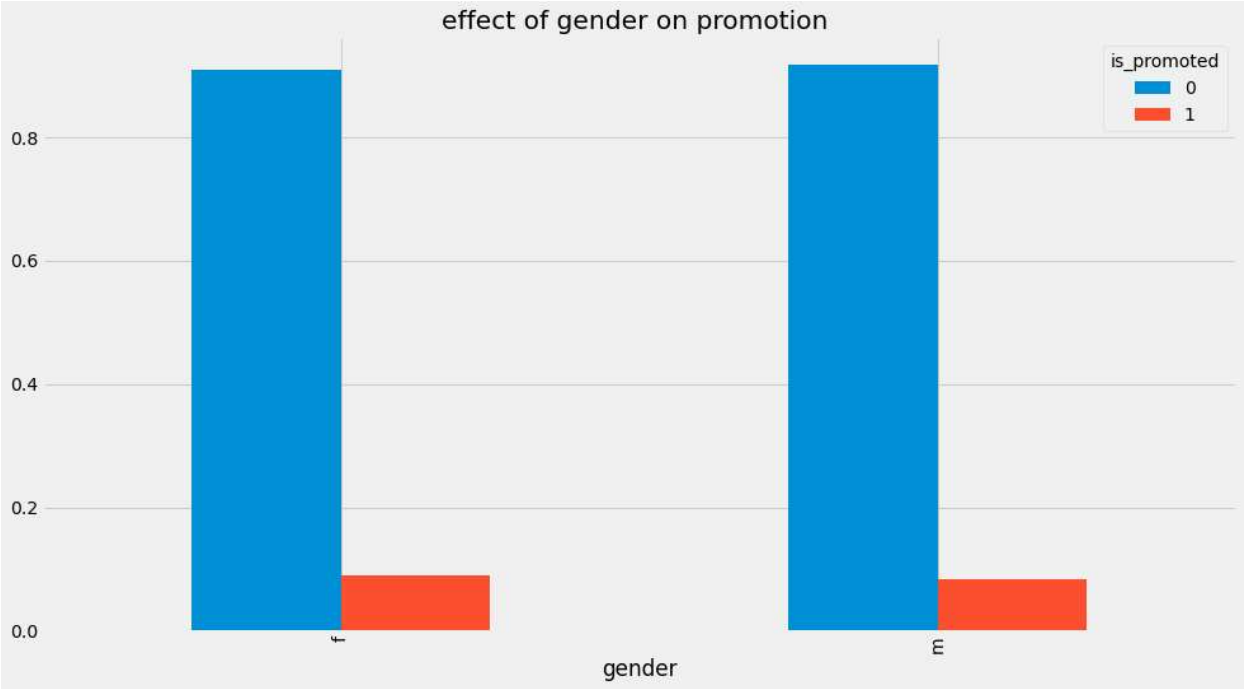
Out[63]:
```
Text(0.5, 1.0, 'recruitment_channel')
```



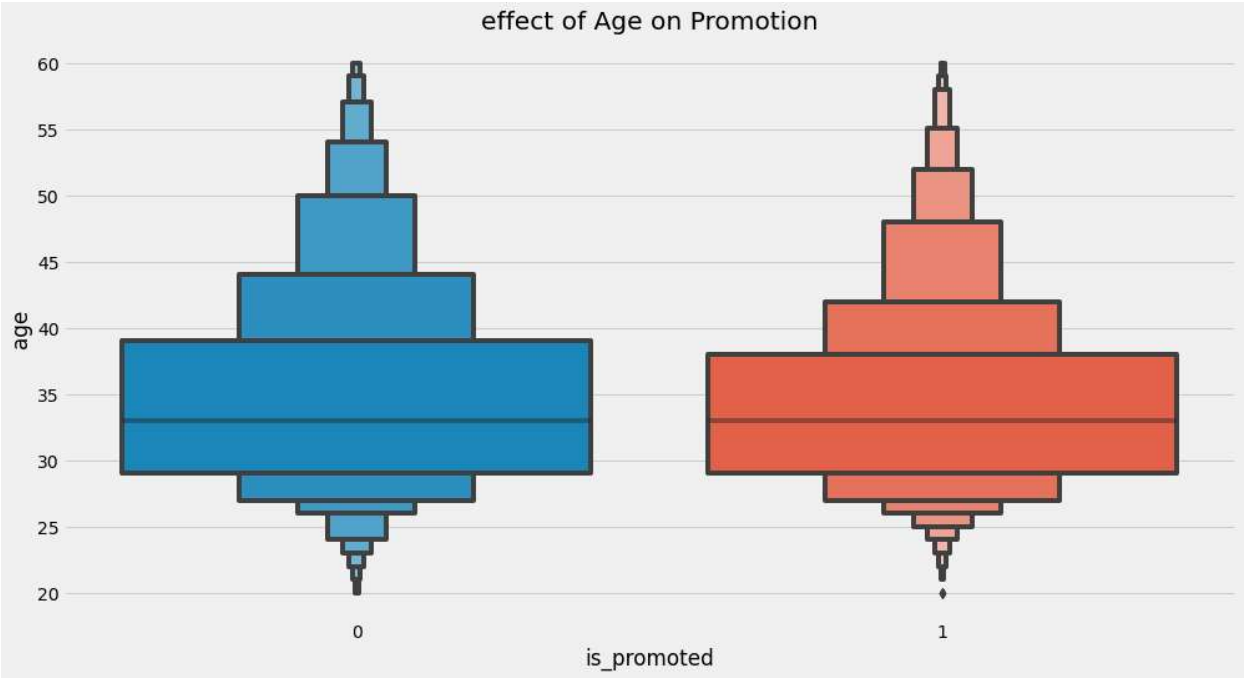
Bivariate Analysis

- C to C
- C to N
- N to N

In [69]:
```
# Gender and is_promtion

x=pd.crosstab(train_data['gender'],train_data['is_promoted'])
x.div(x.sum(1).astype(float),axis=0).plot(kind='bar',stacked=False)
plt.title('effect of gender on promotion ')

plt.show()
```

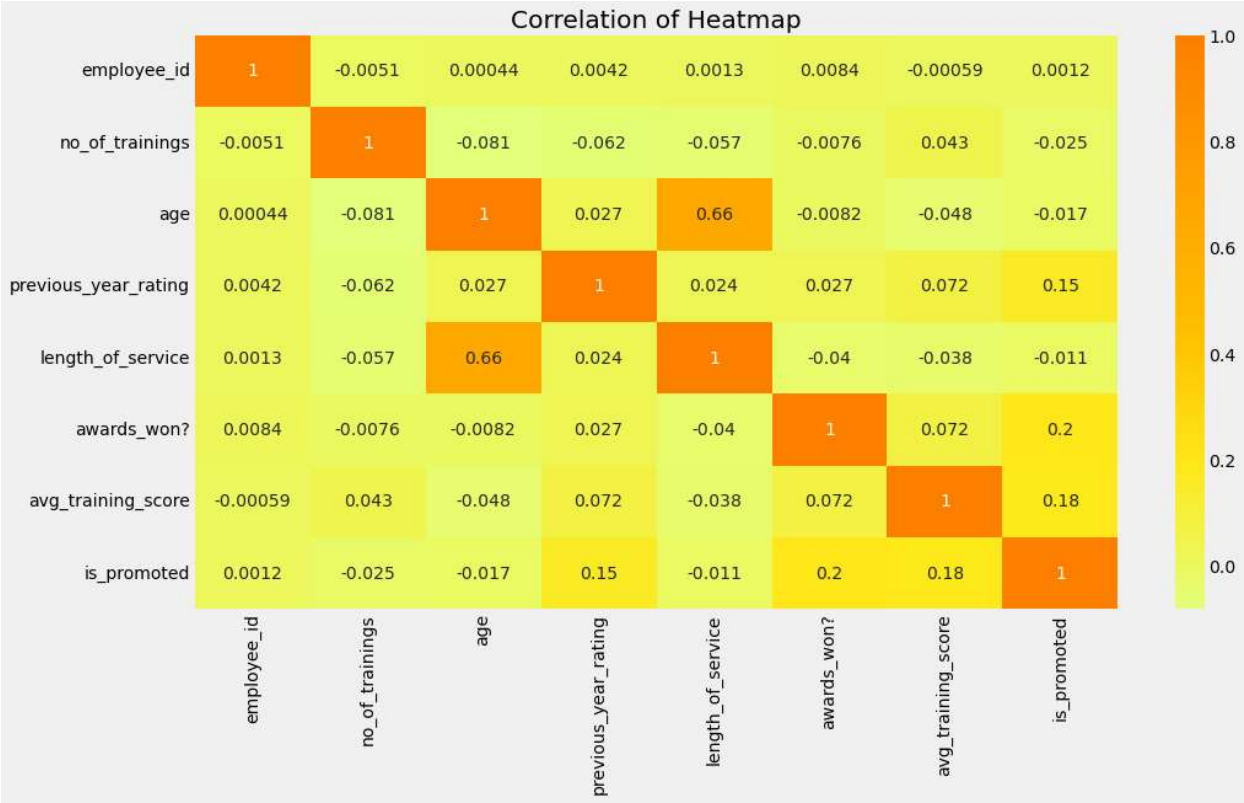


In [66]:
```
# Age and Promotion

sns.boxenplot(train_data['is_promoted'],train_data['age'])
plt.title('effect of Age on Promotion')
plt.show()
```



Mutlivariate Analysis

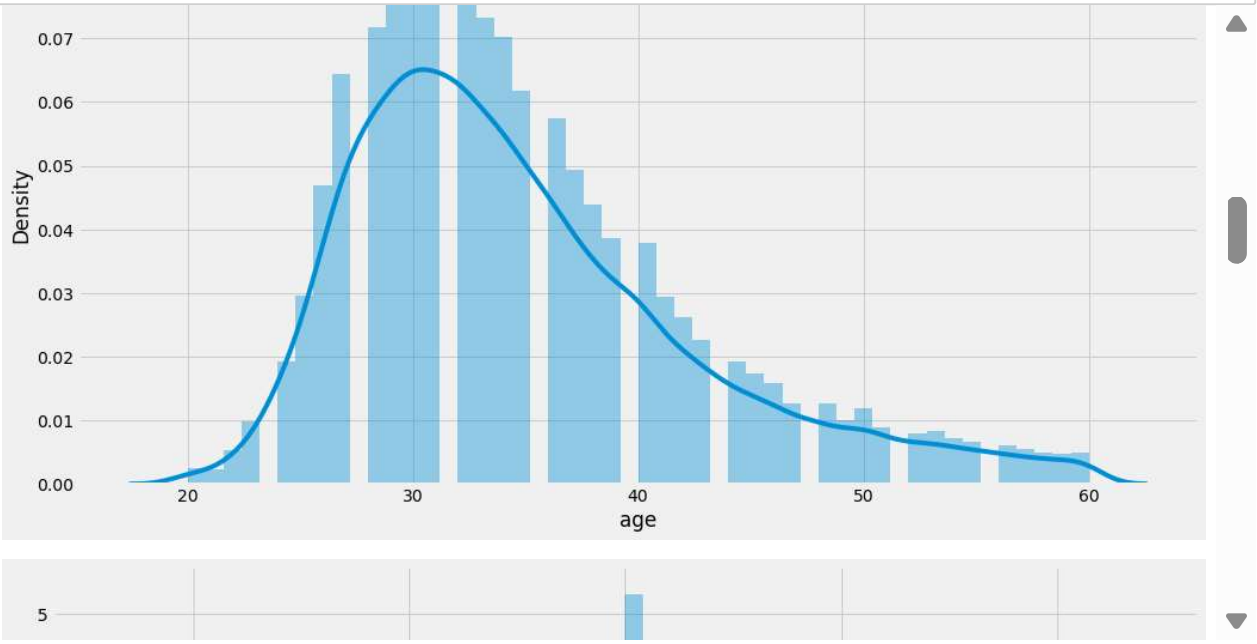
In [70]:
```
sns.heatmap(train_data.corr(),annot=True,cmap='Wistia')
plt.title('Correlation of Heatmap')
plt.show()
```



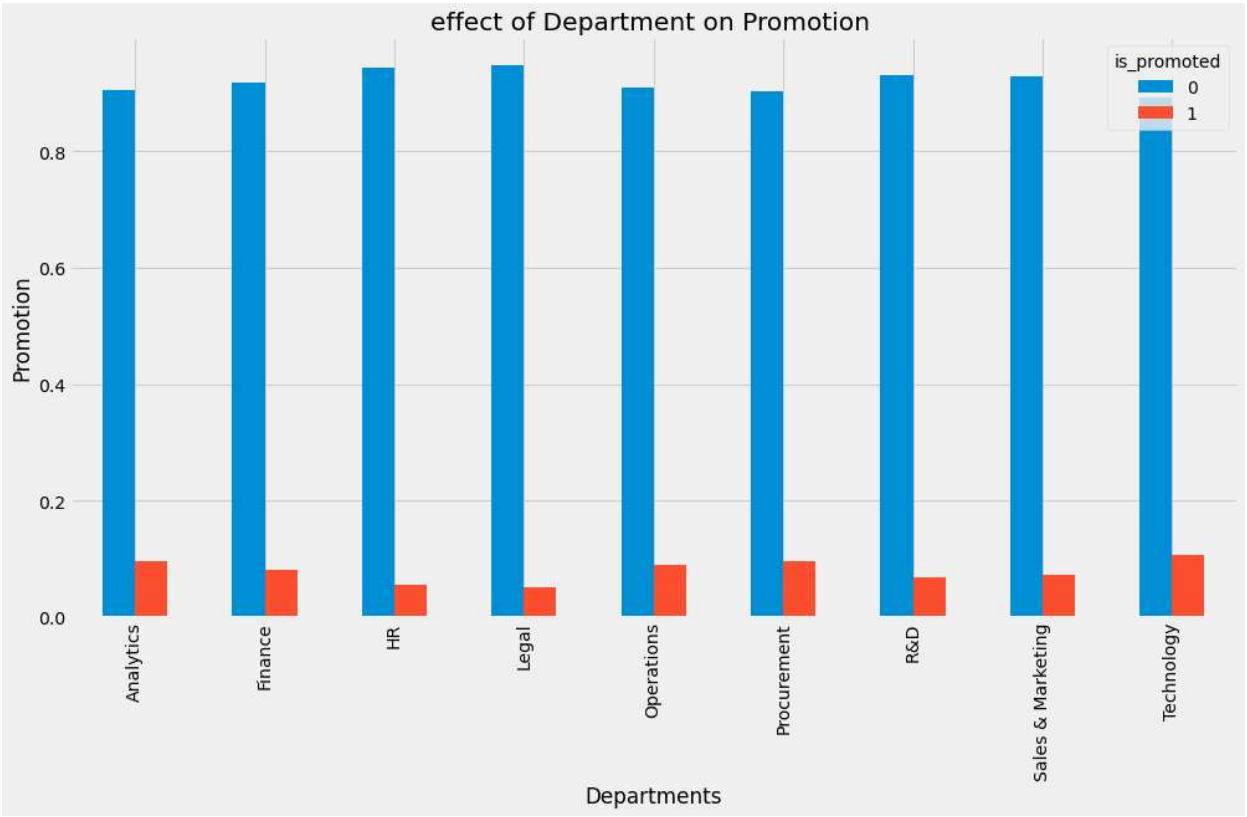
In [73]:
```
# check the skewness
train_data.skew()
```

Out[73]:
```
employee_id           -0.003128
no_of_trainings        3.445434
age                    1.007432
previous_year_rating  -0.260858
length_of_service      1.738061
awards_won?            6.338914
avg_training_score     0.451908
is_promoted            2.972339
dtype: float64
```

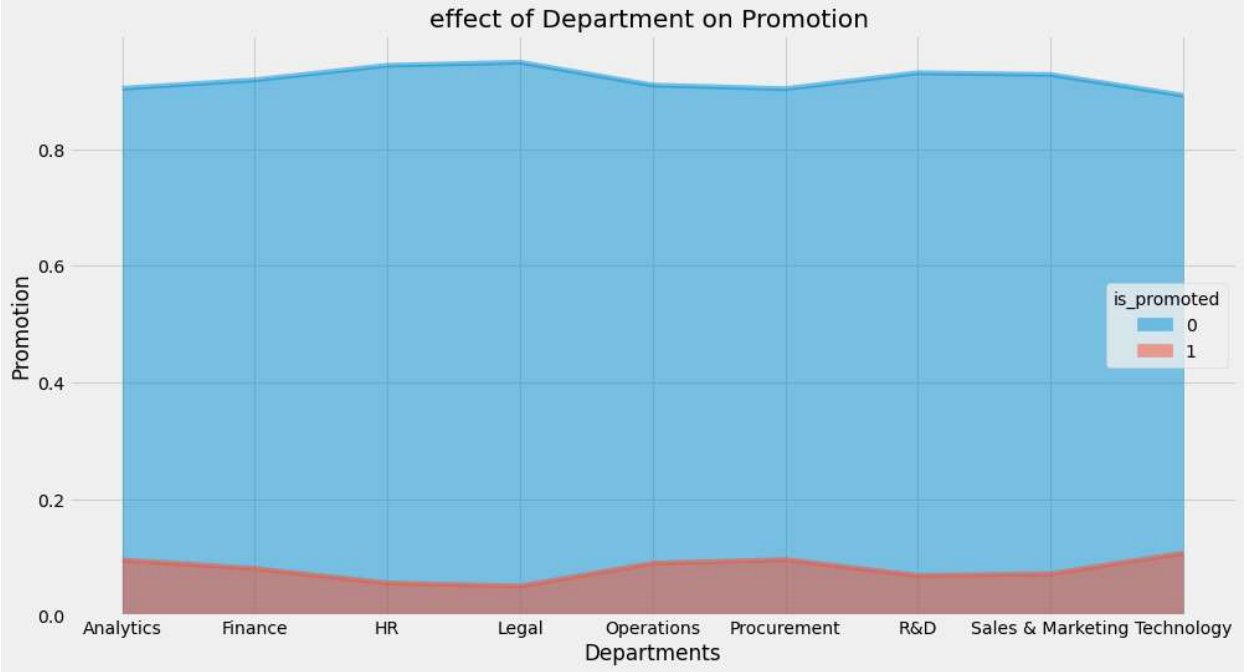
In [76]:
```
for i in num_columns:
    sns.distplot(train_data[i])
    plt.show()
```



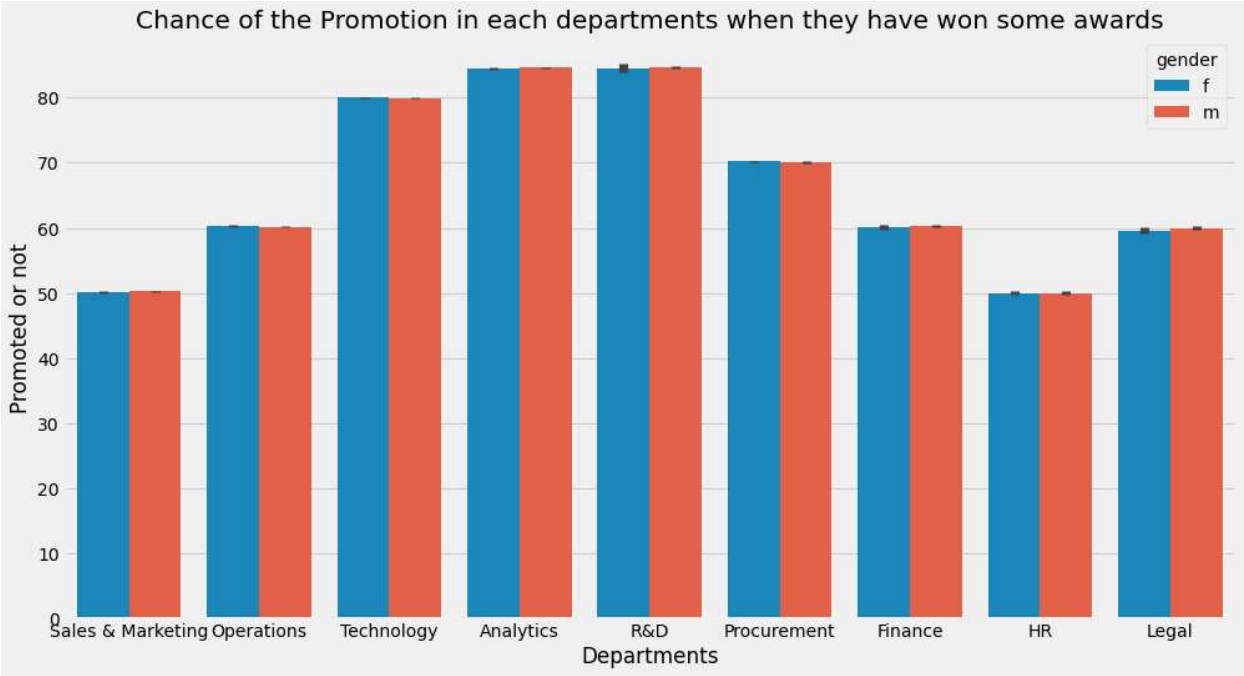
In [82]:
```
# Department and Promotion
x=pd.crosstab(train_data['department'],train_data['is_promoted'])
x.div(x.sum(1).astype(float),axis=0).plot(kind='bar',stacked=False)
plt.title('effect of Department on Promotion ')
plt.xlabel('Departments')
plt.ylabel('Promotion')
plt.show()
```



In [83]:
```
# Department and Promotion in Area chart

x=pd.crosstab(train_data['department'],train_data['is_promoted'])
x.div(x.sum(1).astype(float),axis=0).plot(kind='area',stacked=False)
plt.title('effect of Department on Promotion ')
plt.xlabel('Departments')
plt.ylabel('Promotion')
plt.show()
```



In [84]:
```
# department and award_winning

sns.barplot(train_data['department'],train_data['avg_training_score'],hue = train_data
plt.title('Chance of the Promotion in each departments when they have won some awards'
plt.xlabel('Departments')
plt.ylabel('Promoted or not')
plt.show()
```



In [88]:
```
train_data.columns
```

Out[88]:
```
Index(['employee_id', 'department', 'region', 'education', 'gender',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_year_rating',
       'length_of_service', 'awards_won?', 'avg_training_score', 'is_promoted',
       'sum_metric'],
      dtype='object')
```

Feature Engineeing

In [89]:
```
# create a metric of sum
train_data['sum_metric'] = train_data['awards_won?'] + train_data['previous_year_rating
test_data['sum_metric'] = train_data['awards_won?'] + train_data['previous_year_rating

# create a total columns

train_data['total_score'] = train_data['avg_training_score'] * train_data['no_of_train
test_data['total_score'] = test_data['avg_training_score'] * test_data['no_of_training
```

In [91]:
```
train_data.head()
```

Out[91]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previ
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	
1	65141	Operations	region_22	Bachelor's	m	other	1	30	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	
4	48945	Technology	region_26	Bachelor's	m	other	1	45	

drop columns

In [92]:
```
train_data = train_data.drop(['employee_id','recruitment_channel','region'],axis=1)
test_data = test_data.drop(['employee_id','recruitment_channel','region'],axis=1)

train_data.columns
test_data.columns
```

Out[92]:
```
Index(['department', 'education', 'gender', 'no_of_trainings', 'age',
       'previous_year_rating', 'length_of_service', 'awards_won?',
       'avg_training_score', 'sum_metric', 'total_score'],
      dtype='object')
```

In [93]:
```
test_data.columns
```

Out[93]:
```
Index(['department', 'education', 'gender', 'no_of_trainings', 'age',
       'previous_year_rating', 'length_of_service', 'awards_won?',
       'avg_training_score', 'sum_metric', 'total_score'],
      dtype='object')
```

In [98]:
```
# Checking the Categorical columns convert to numerical columns

train_data.select_dtypes('object')
```

Out[98]:

	department	education	gender
0	Sales & Marketing	Master's & above	f
1	Operations	Bachelor's	m
2	Sales & Marketing	Bachelor's	m
3	Sales & Marketing	Bachelor's	m
4	Technology	Bachelor's	m
...	...	...	...
54803	Technology	Bachelor's	m
54804	Operations	Master's & above	f
54805	Analytics	Bachelor's	m
54806	Sales & Marketing	Bachelor's	m
54807	HR	Bachelor's	m

54808 rows × 3 columns

In [101]:
```
train_data['education'] = train_data['education'].replace(("Master's & above","Bachelo

test_data['education'] = test_data['education'].replace(("Master's & above","Bachelor'
```

In [106]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [108]:
```
train_data['department'] = le.fit_transform(train_data['department'])
test_data['department'] = le.fit_transform(test_data['department'])

train_data['gender'] = le.fit_transform(train_data['gender'])
test_data['gender'] = le.fit_transform(test_data['gender'])
```

In [110]:
```
train_data.head()
```

Out[110]:

	department	education	gender	no_of_trainings	age	previous_year_rating	length_of_service	awards_w
0	7	3	0	1	35	5.0	8	
1	4	2	1	1	30	5.0	4	
2	7	2	1	1	34	3.0	7	
3	7	2	1	2	39	1.0	10	
4	8	2	1	1	45	3.0	2	

In [111]:
```
test_data.head()
```

Out[111]:

	department	education	gender	no_of_trainings	age	previous_year_rating	length_of_service	awards_w
0	8	2.0	1	1	24	NaN	1	
1	2	2.0	0	1	31	3.0	5	
2	7	2.0	1	1	31	1.0	4	
3	5	2.0	0	3	31	2.0	9	
4	1	2.0	1	1	30	4.0	7	

In [114]:
```
# Splitting the data

x = train_data.drop(['is_promoted'],axis=1)
y = train_data['is_promoted']


x_test = test_data
```

In [115]:
```
x.shape , y.shape
```

Out[115]:
```
((54808, 11), (54808,))
```

In [117]:
```
x_test.shape
```

Out[117]:
```
(23490, 11)
```

In [118]:
```
# Handling the Imblance data

from imblearn.over_sampling import SMOTE

oversample = SMOTE()
x,y = oversample.fit_resample(x,y)
```

In [121]:
```
from sklearn.model_selection import train_test_split

x_train,y_train, x_test, y_test = train_test_split(x,y,test_size=0.20,random_state=35)
```

In [122]:
```
x_train.shape , y_train.shape
```

Out[122]:
```
((80224, 11), (20056, 11))
```

In [124]:
```
x_test.shape , y_test.shape
```

Out[124]:
```
((80224,), (20056,))
```

In [ ]:
```

```