In [1]:

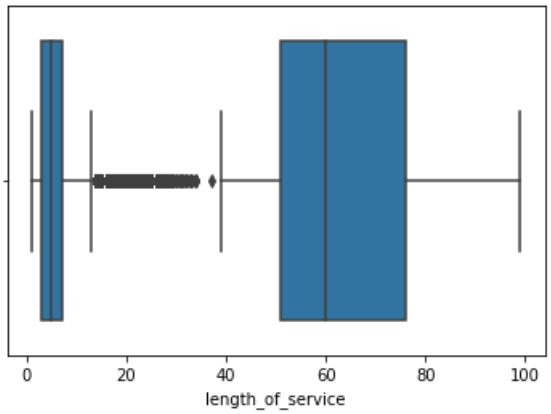
```
import pandas as pd# we import pandas to handle the file and its save our lots of time, it handle large data set efficeintly.
import numpy as np #for creating arrays
import seaborn as sns #for data visualzation
import matplotlib.pyplot as plt # for data visualization

df1 = pd.read_csv(r'C:\Users\PC-chetan\Desktop\train.csv') # trian data

df2 = pd.read_csv(r'C:\Users\PC-chetan\Desktop\test.csv') # test data

df1.education.fillna("Bachelor's", inplace=True)

df2.education.fillna("Bachelor's", inplace=True)

df1.previous_year_rating.fillna('3.0',inplace=True)

df2.previous_year_rating.fillna('3.0',inplace=True)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df1.drop(columns=['employee_id','region','recruitment_channel'], inplace=True)

df2.drop(columns=['employee_id','region','recruitment_channel'], inplace=True)

#lets encode the education in their degree of importance
df1['education'] = df1['education'].replace(("Master's & above", "Bachelor's", "Below Secondary"),
                                            (3, 2, 1))
df2['education'] = df2['education'].replace(("Master's & above", "Bachelor's", "Below Secondary"), (3, 2, 1))

df1.gender = le.fit_transform(df1.gender)

df1.department = le.fit_transform(df1.department)

df2.department = le.transform(df2.department)


df2['gender'] = df2['gender'].replace(("m", "f"),(1,0))




df1.shape

df1.select_dtypes('number').head()

df2.select_dtypes('number').head()

sns.boxplot(data=df1,x=df1['avg_training_score'])

df1.shape

Q1=df1['avg_training_score'].quantile(0.25)
Q3=df1['avg_training_score'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
min_1 = Q1-(1.5)*IQR
max_1 = Q3+(1.5)*IQR
print(min_1)
print( max_1)


df1['avg_training_score'].unique()

df1 = df1[df1['avg_training_score']< max_1]

df1.shape


sns.boxplot(data=df1,x=df1['length_of_service'])

Q2=df1['length_of_service'].quantile(0.25)
Q4=df1['length_of_service'].quantile(0.75)
IQRt=Q4-Q2
print(Q2)
print(Q4)
print(IQRt)
min_2 = Q2-1.5*IQRt
```

```
max_2 = Q4+1.5*IQRt
print(max_2,min_2)

df1['length_of_service'].unique()
df1 = df1[df1['length_of_service'] > 13]


# feature engineering
#it is the most important part of the data preprocessing

df1.shape

df1['sum_metric'] = df1['awards_won?']+ df1['previous_year_rating']


# creating a total score column
df1['total_score'] = df1['avg_training_score'] * df1['no_of_trainings']


pd.set_option('display.max_rows', 5000) # for getting the max veiw of raws
pd.set_option('display.max_column', 5000) # for getting the max veiw of columns

df1[(df1['previous_year_rating'] == 1.0) &
    (df1['awards_won?'] == 0) & (df1['avg_training_score'] < 60) & (df1['is_promoted'] == 1)]

df1 = df1.drop(df1[(df1['previous_year_rating'] == 1.0) &
    (df1['awards_won?'] == 0) & (df1['avg_training_score'] < 60) & (df1['is_promoted'] == 1)].index)

df1.shape
```

```
51.0
76.0
25.0
13.5
113.5
3.0
7.0
4.0
13.0 -3.0
```

Out[1]:

```
(3487, 12)
```



visualizations

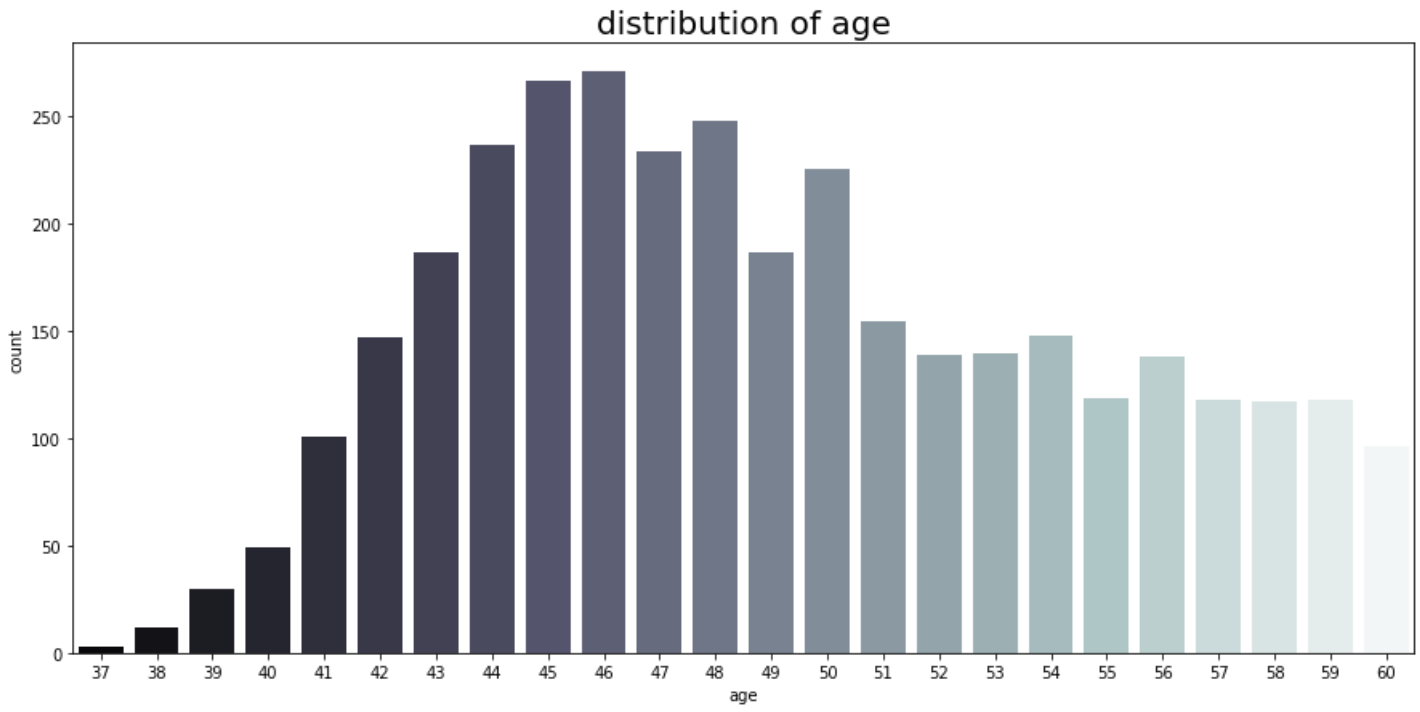
- using
- univariate
- bivariate and
- multivariate analysisvizualisation

1. univariate:

- Univariate analysis is the simplest form of statistical analysis.
- The key fact is that only one variable is involved in Univariate Analysis.
- Univariate analysis can yield misleading results in cases in where multivariate analysis is more appropriate.
- This is an Essential step to understand the variables present in the dataset step by step.
- We can use Distribution plots to check the distribution of the Numerical Columns.
- We can check distribution of Categorical Columns using Pie charts, Count plots etc

In [2]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='age', data = df1, palette= 'bone')
plt.title('distribution of age', fontsize = 20)
plt.show()
```



In [3]:

```
df1.age.unique()
```

Out[3]:

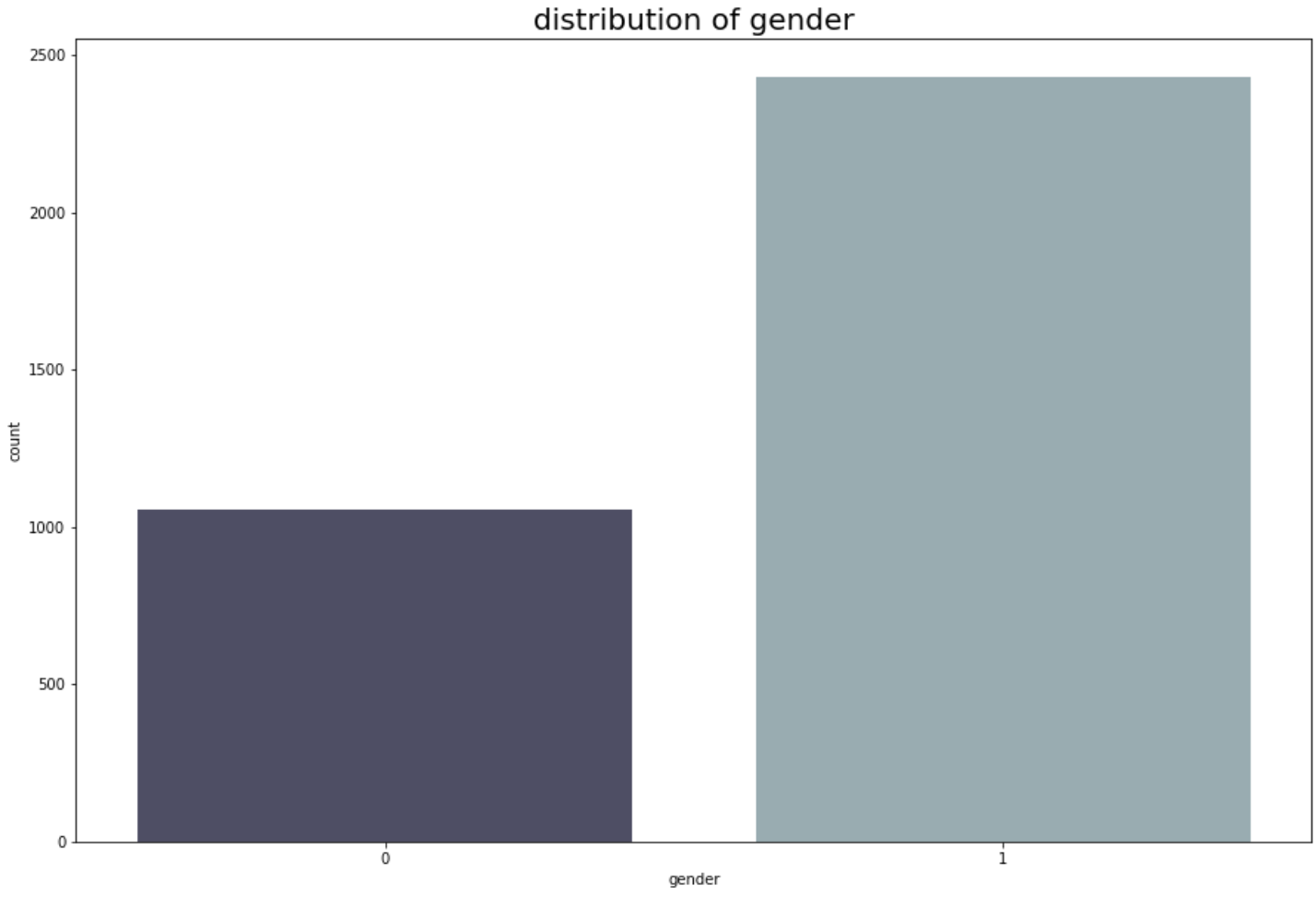
```
array([39, 59, 50, 60, 42, 43, 54, 45, 44, 46, 57, 58, 49, 55, 51, 52, 47,
       48, 53, 40, 56, 41, 38, 37], dtype=int64)
```

In [4]:

```
# min age category = 37
# max age group = 46
```

In [5]:

```
plt.figure(figsize=(15,10))
sns.countplot(x='gender', data = df1, palette= 'bone')
plt.title('distribution of gender', fontsize = 20)
plt.show()
```



In [6]:

```
df1.gender.value_counts()
```

Out[6]:

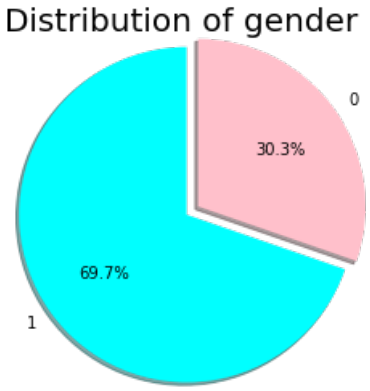
```
1    2431
0    1056
Name: gender, dtype: int64
```

In [7]:

```
q = df1['gender'].value_counts().index
w = df1['gender'].value_counts()
c=['cyan','pink']
e = [0, 0.1]
plt.pie(w,labels= q, colors = c, explode = e ,shadow = True, autopct='%1.1f%%',startangle = 90,  radius = 1.2)
plt.title('Distribution of gender ', fontsize = 20)
```

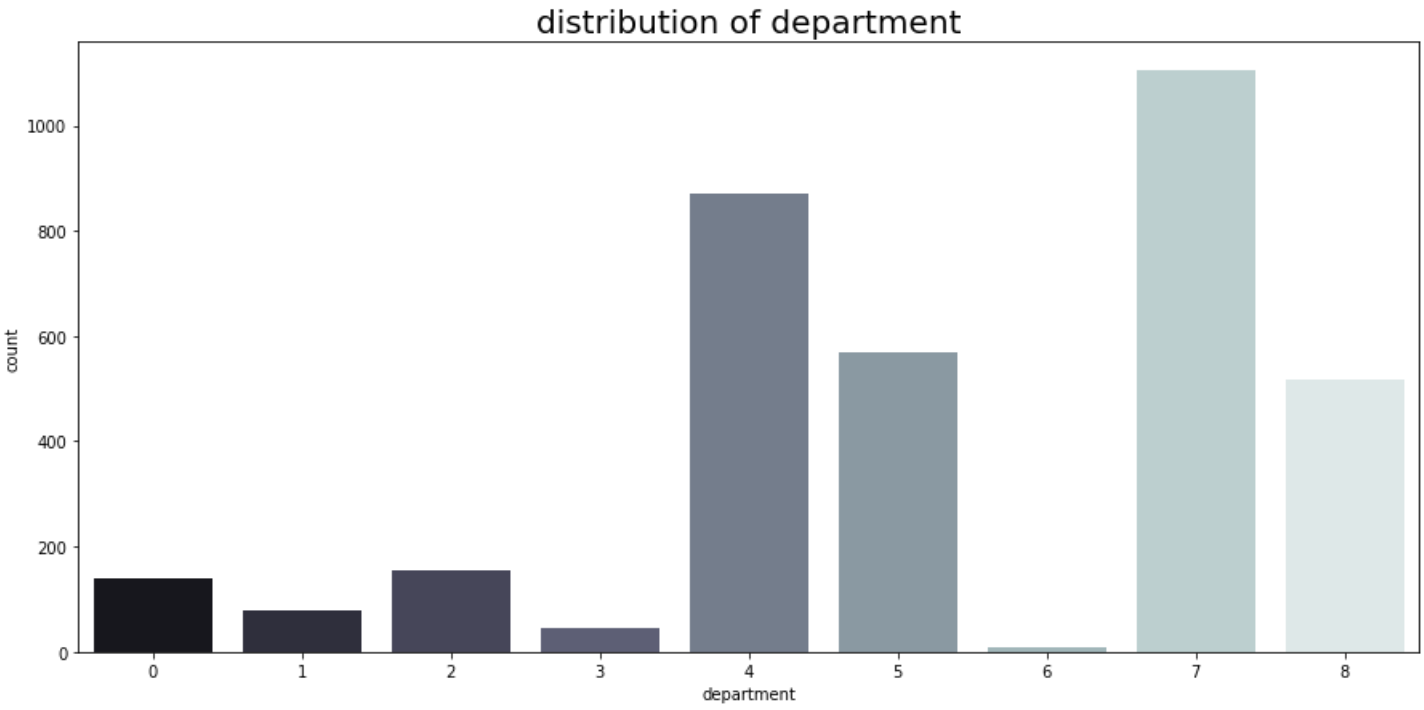
Out[7]:

```
Text(0.5, 1.0, 'Distribution of gender ')
```



In [8]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='department', data = df1, palette= 'bone')
plt.title('distribution of department', fontsize = 20)
plt.show()
```



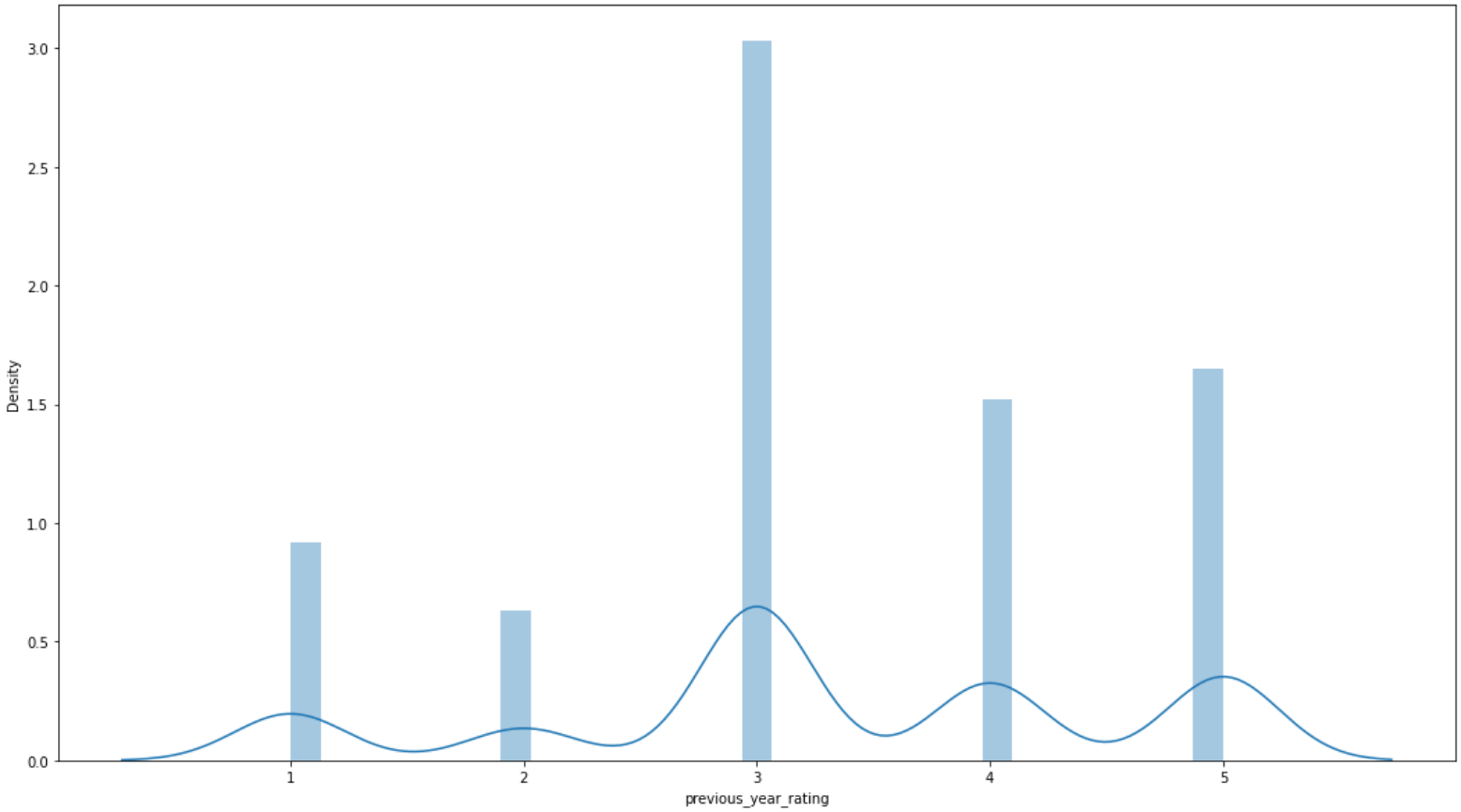
In [9]:

```
# max number of employee in sales and marketing
# min number of employee in R&D department
```

In [10]:

```
x = df1['previous_year_rating']
plt.figure(figsize=(18,10))
sns.distplot(x)
plt.show()
```

```
c:\python\python39\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [11]:

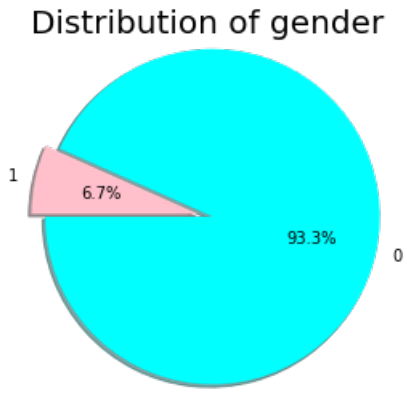
```
# most of the employee have 3 rating last year
```

In [12]:

```
q = df1['is_promoted'].value_counts().index
w = df1['is_promoted'].value_counts()
c=['cyan','pink']
e = [0, 0.1]
plt.pie(w,labels= q, colors = c, explode = e ,shadow = True, autopct='%1.1f%%',startangle = 180,  radius = 1.2)
plt.title('Distribution of gender ', fontsize = 20)
```

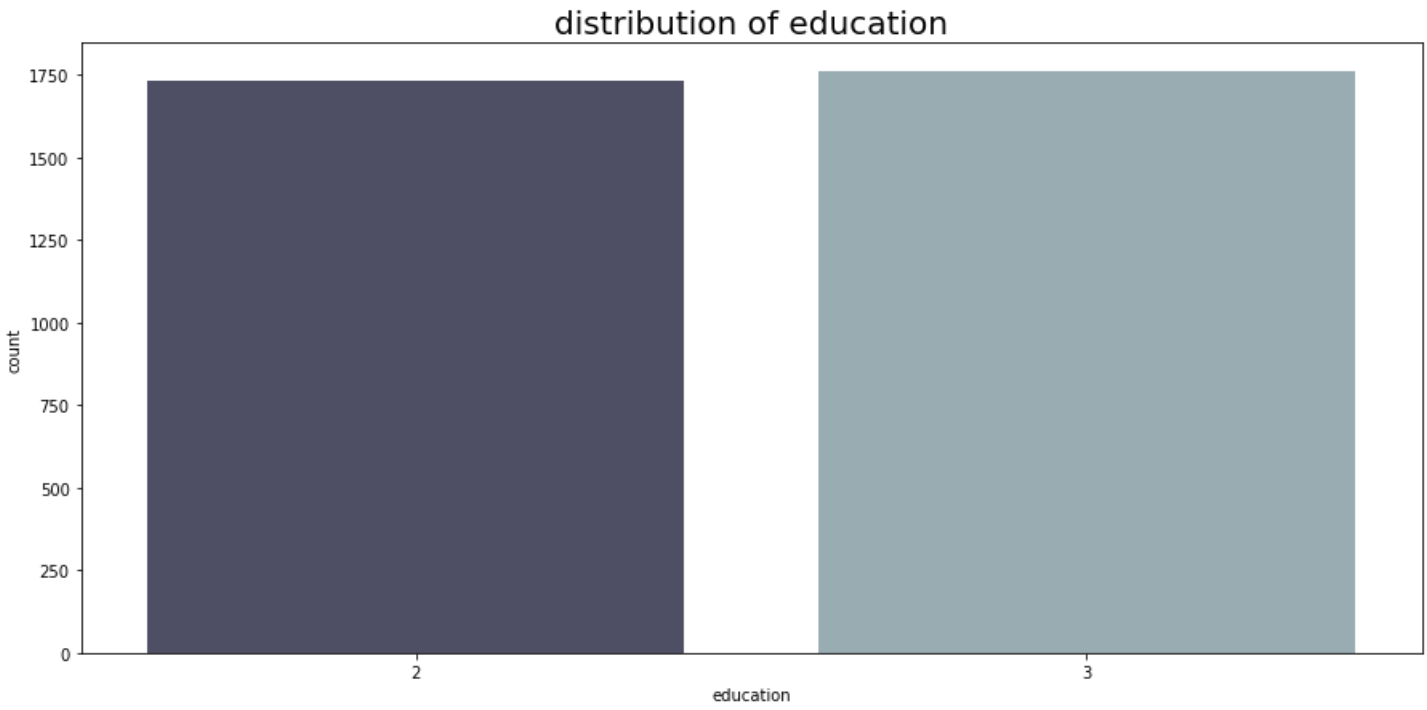
Out[12]:

```
Text(0.5, 1.0, 'Distribution of gender ')
```



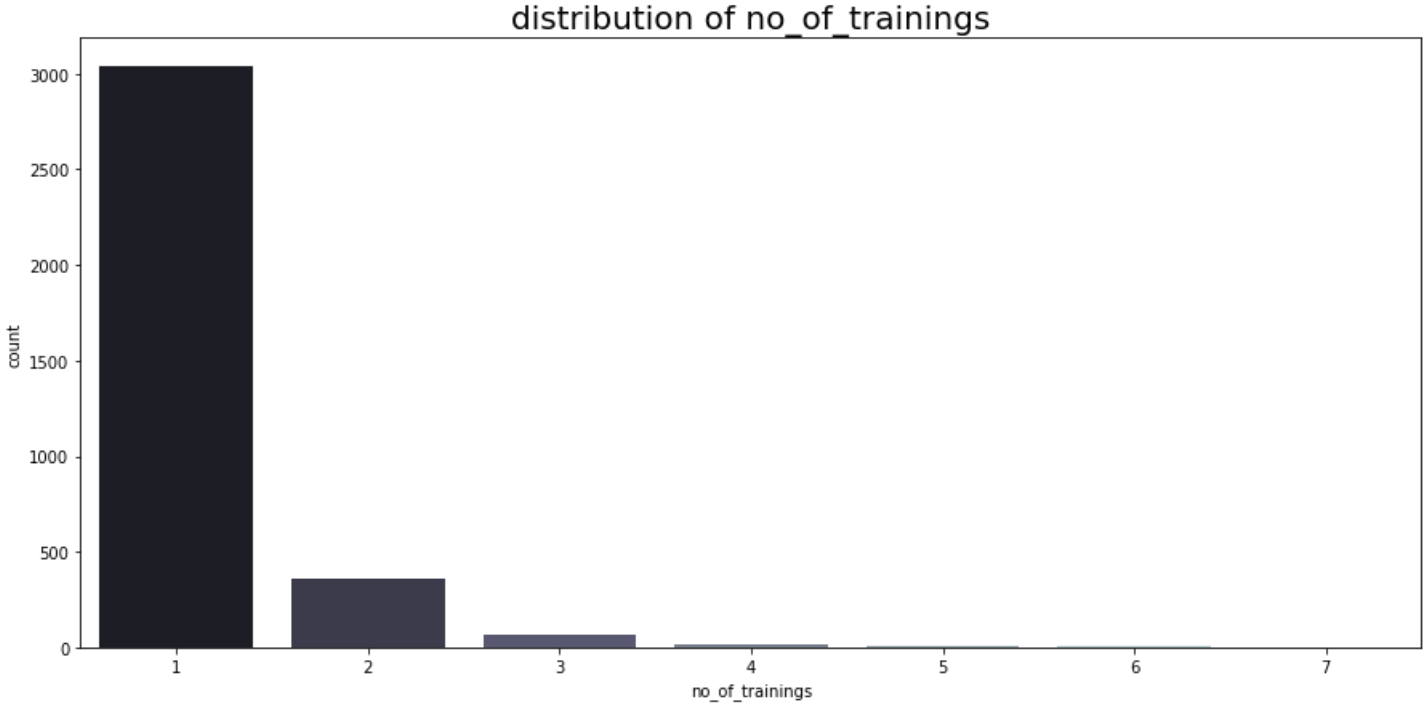
In [13]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='education', data = df1, palette= 'bone')
plt.title('distribution of education', fontsize = 20)
plt.show()
```



In [14]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='no_of_trainings', data = df1, palette= 'bone')
plt.title('distribution of no_of_trainings', fontsize = 20)
plt.show()
```

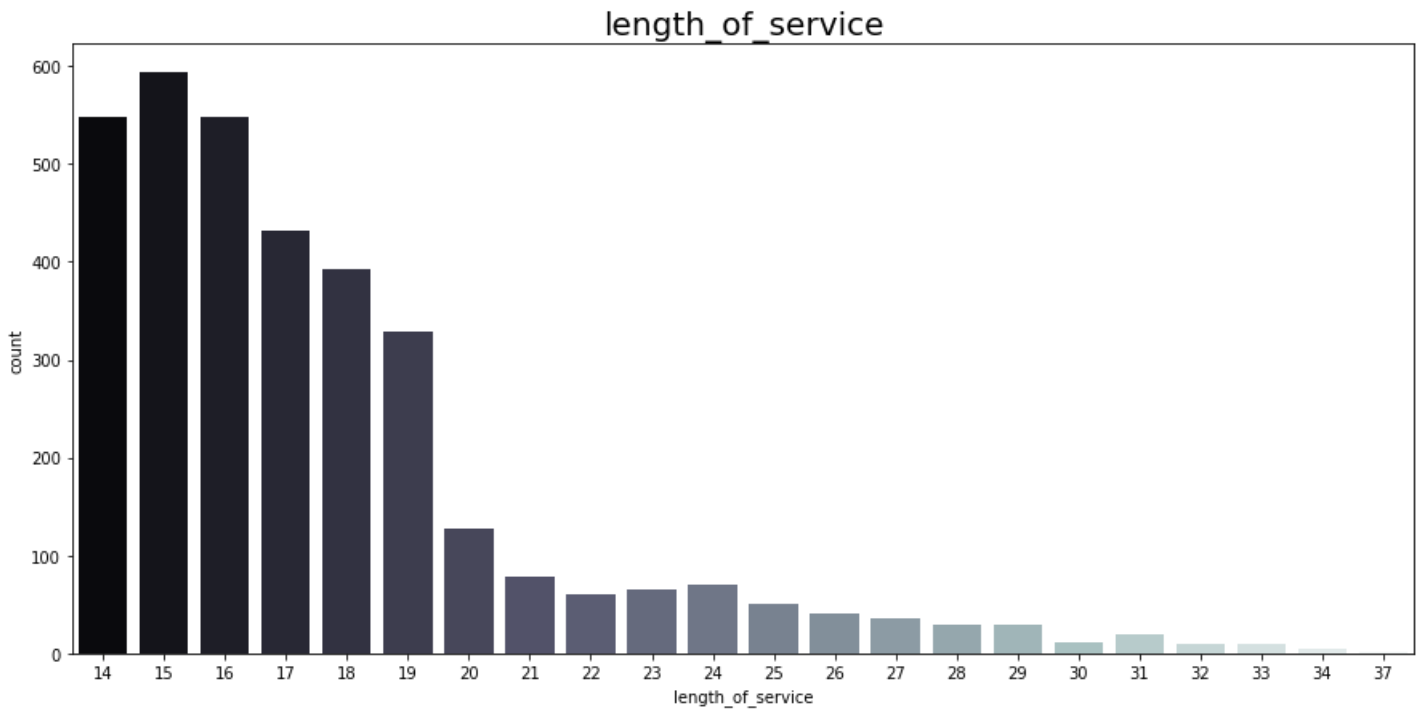


In [15]:

 # most of the employees have only one training

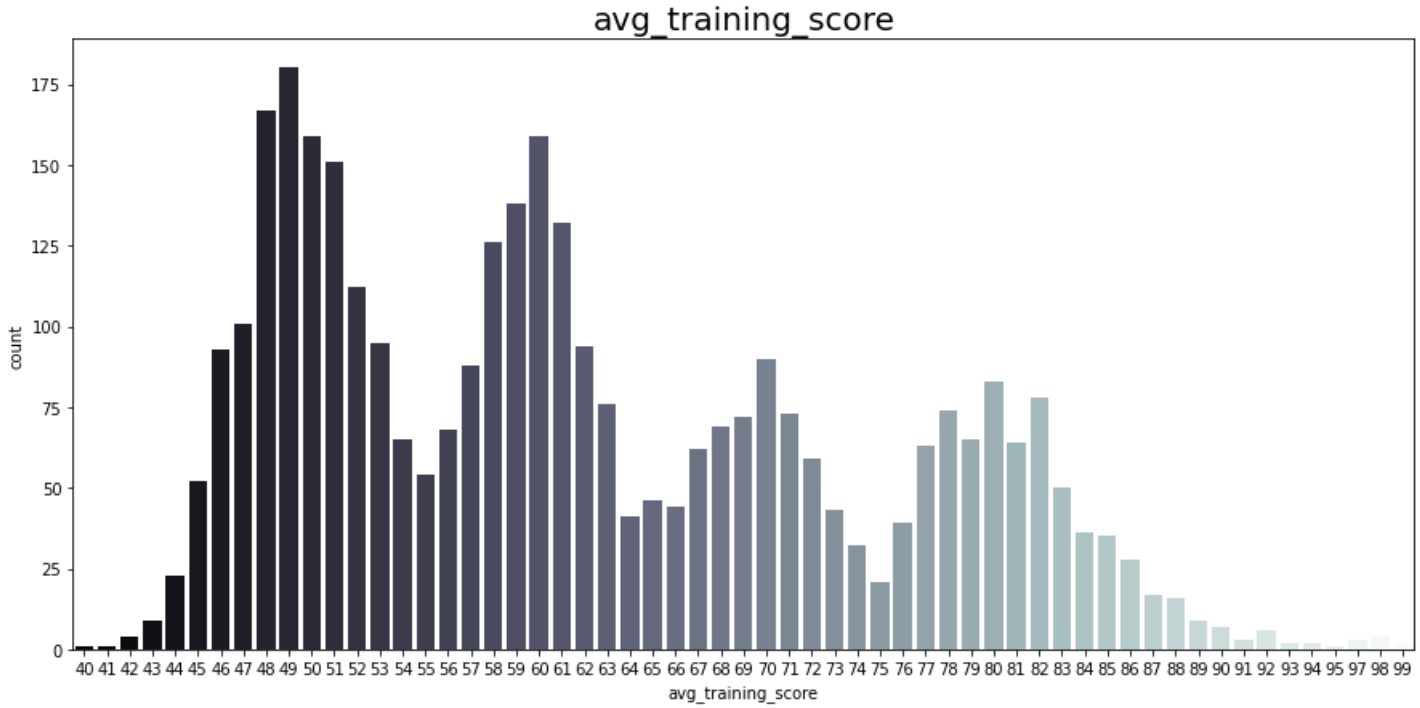
In [16]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='length_of_service', data = df1, palette= 'bone')
plt.title('length_of_service', fontsize = 20)
plt.show()
```



In [17]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='avg_training_score', data = df1, palette= 'bone')
plt.title('avg_training_score', fontsize = 20)
plt.show()
```



In [18]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='award+rating', data = df1, palette= 'bone')
plt.title('award+rating', fontsize = 20)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\PC-CHE~1\AppData\Local\Temp/ipykernel_12960/3406271784.py in <module>
      1 plt.figure(figsize=(15,7))
----> 2 sns.countplot(x='award+rating', data = df1, palette= 'bone')
      3 plt.title('award+rating', fontsize = 20)
      4 plt.show()

c:\python\python39\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

c:\python\python39\lib\site-packages\seaborn\categorical.py in countplot(x, y, hue, data, order, hue_order, orient, color, palette, saturation, dodge,
ax, **kwargs)
   3596         raise ValueError("Cannot pass values for both `x` and `y`")
   3597
-> 3598     plotter = _CountPlotter(
   3599         x, y, hue, data, order, hue_order,
   3600         estimator, ci, n_boot, units, seed,

c:\python\python39\lib\site-packages\seaborn\categorical.py in __init__(self, x, y, hue, data, order, hue_order, estimator, ci, n_boot, units, seed, ori
ent, color, palette, saturation, errcolor, errwidth, capsize, dodge)
   1582                  errwidth, capsize, dodge):
   1583         """Initialize the plotter."""
-> 1584         self.establish_variables(x, y, hue, data, orient,
   1585                                  order, hue_order, units)
   1586         self.establish_colors(color, palette, saturation)

c:\python\python39\lib\site-packages\seaborn\categorical.py in establish_variables(self, x, y, hue, data, orient, order, hue_order, units)
    151                 if isinstance(var, str):
    152                     err = "Could not interpret input '{}'".format(var)
--> 153                     raise ValueError(err)
    154
    155             # Figure out the plotting orientation

ValueError: Could not interpret input 'award+rating'
<Figure size 1080x504 with 0 Axes>
```

In [ ]:

```
plt.figure(figsize=(15,10))
sns.countplot(x='score_total', data = df1, palette= 'bone')
plt.title('score_total', fontsize = 20)
plt.show()
```

2. Bivariate Analysis

Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them.

There are three Types of Bivariate Analysis which can be used to understand the association between two variables in a dataset.

- Categorical vs Categorical
- Categorical vs Numerical
- Numerical vs Numerical

In [ ]:

```
plt.figure(figsize=(12,10))
sns.countplot(df1['age'], hue=df1['gender'], palette = 'husl')
plt.title('age according to the gender ', fontsize = 20)
plt.show()
```

In [ ]:

```
plt.figure(figsize=(12,10))
sns.countplot(df1['education'], hue=df1['gender'], palette = 'husl')
plt.title('education according to the gender ', fontsize = 20)
plt.show()
```

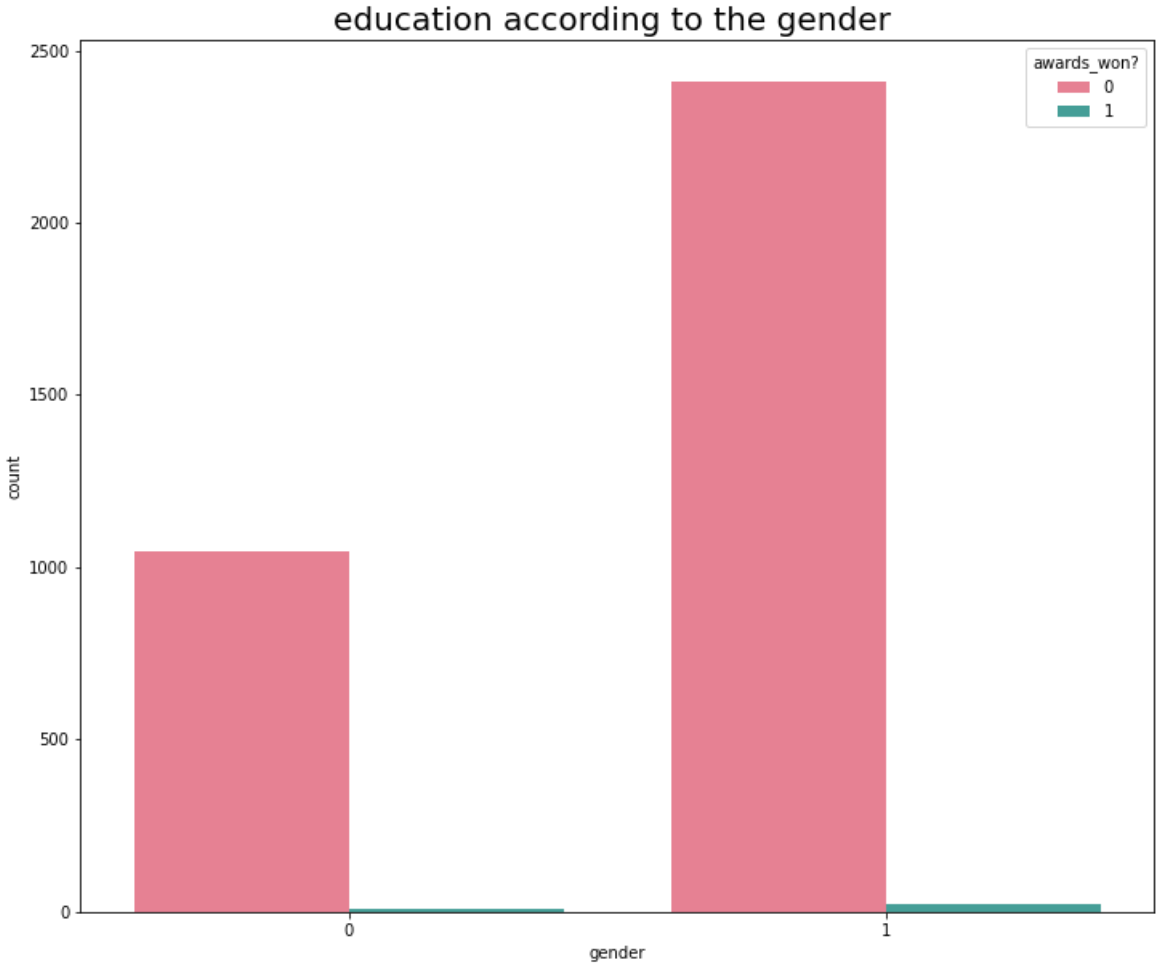
In [ ]:

```
sns.catplot(y="department",
            x="avg_training_score",data=df1,
            palette="colorblind",height=3,kind="bar",aspect=2)
plt.title("Department wise training score")
plt.show()
```

In [19]:

```
plt.figure(figsize=(12,10))
sns.countplot(df1['gender'], hue=df1['awards_won?'], palette = 'husl')
plt.title('education according to the gender ', fontsize = 20)
plt.show()
```

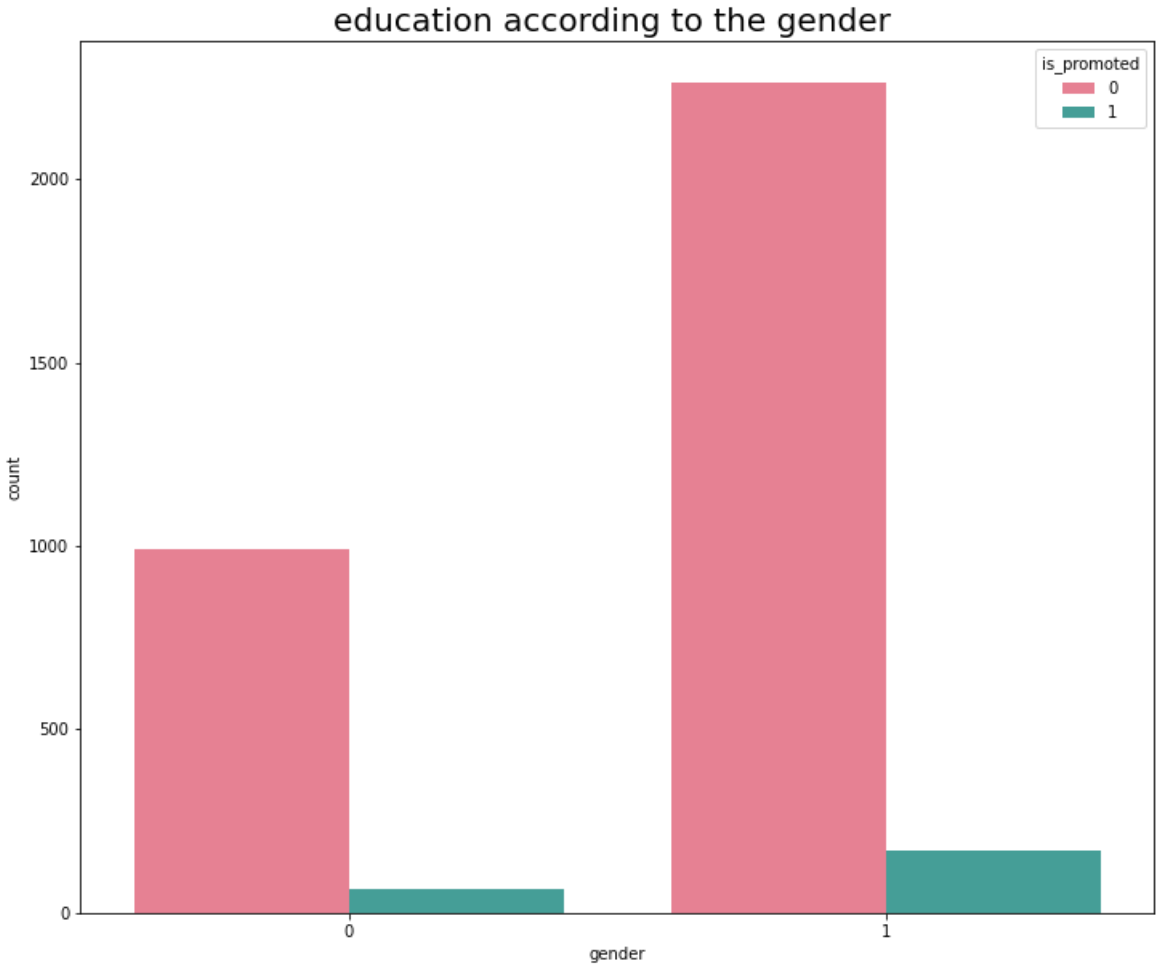
```
c:\python\python39\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [20]:

```
plt.figure(figsize=(12,10))
sns.countplot(df1['gender'], hue=df1['is_promoted'], palette = 'husl')
plt.title('education according to the gender ', fontsize = 20)
plt.show()
```

c:\python\python39\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(



3. Multivariate Analysis

- Multivariate analysis is based on the principles of multivariate statistics, which involves observation and analysis of more than one statistical outcome variable at a time.

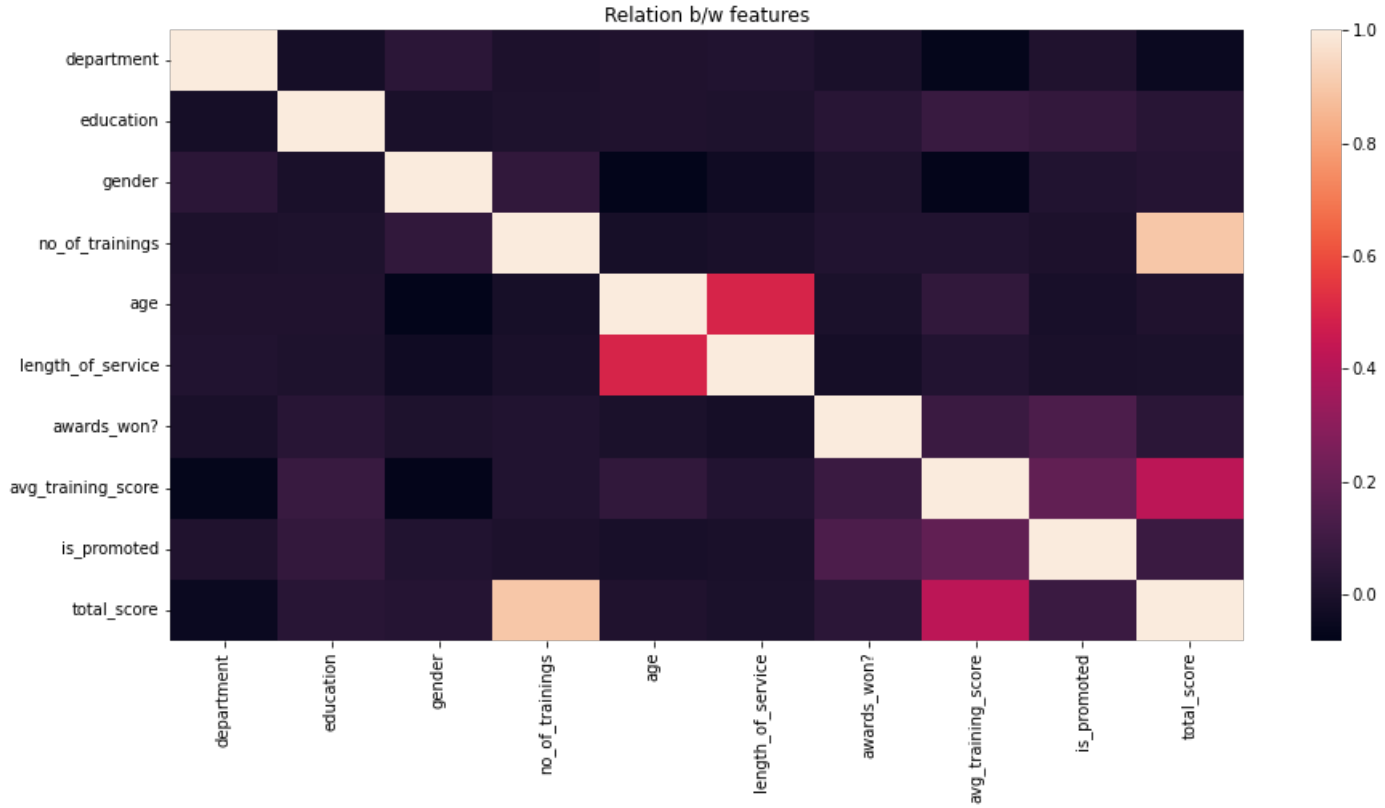
  We will use Heatmaps for finding relation between all the variables in the dataset. A heatmap is a graphical representation of data that uses a system of color-coding to represent different values.

  Before Using Heatmaps, Let's understand how to analyze Correlation.

- If the Correlation Value is +1, it means that the two columns highly similar
- If the Correlation Value is 0, It means that the two columns are having no relation,and
- If the Correlation Value is -1, It means that the two columns are completely Opposite to each other

In [21]:

```
plt.figure(figsize=(15,7))
sns.heatmap(df1.corr())
plt.title('Relation b/w features')
plt.show()
```


Relation b/w features
department
education
gender
no_of_trainings
age
length_of_service
awards_won?
avg_training_score
is_promoted
total_score
1.0
0.8
0.6
0.4
0.2
0.0

In [ ]: