In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
test=pd.read_csv("C:/Users/Shrinidhi/Downloads/employee+promotion/employee+promot
```

In [3]:
```
train=pd.read_csv("C:/Users/Shrinidhi/Downloads/employee+promotion/employee+promo
```

In [4]:
```
test.head()
```

Out[4]:

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

In [5]:
```
train.head()
```

Out[5]:

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

In [6]:
```
len(test)
```

Out[6]:
```
23490
```

In [7]:
```
len(train)
```

Out[7]:
```
54808
```

EDA

In [8]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 14 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  KPIs_met >80%         54808 non-null  int64  
 11  awards_won?           54808 non-null  int64  
 12  avg_training_score    54808 non-null  int64  
 13  is_promoted           54808 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.9+ MB
```

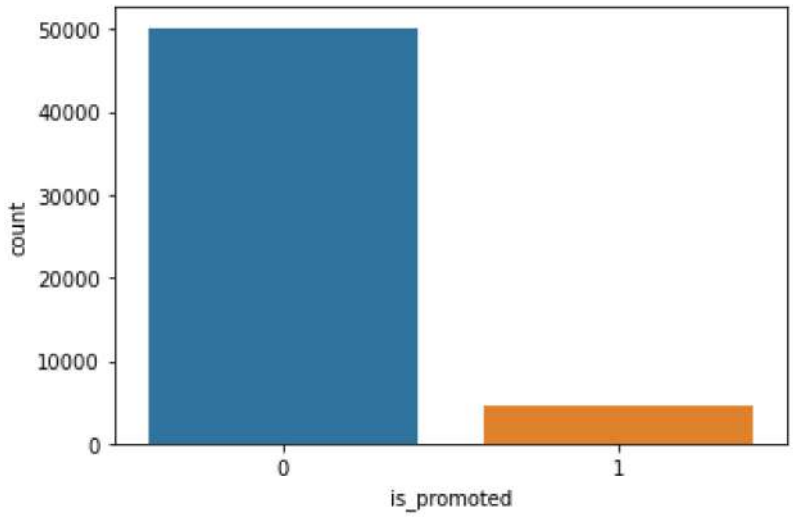
In [9]:
```
test.shape
```

Out[9]:
```
(23490, 13)
```

In [10]:
```
sns.countplot(train.is_promoted)
```

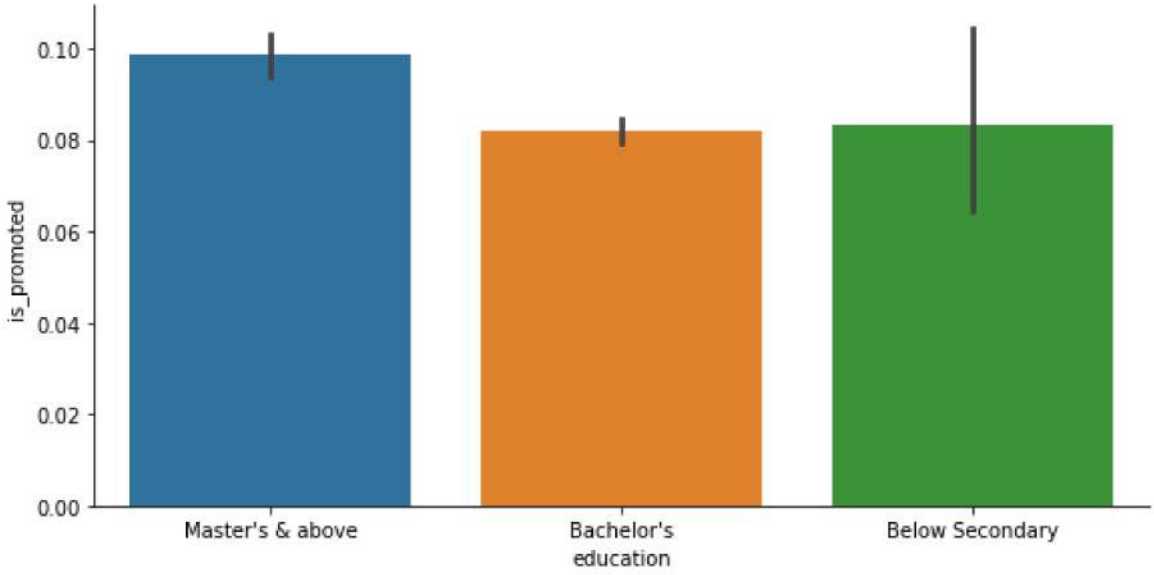
```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[10]: <AxesSubplot:xlabel='is_promoted', ylabel='count'>



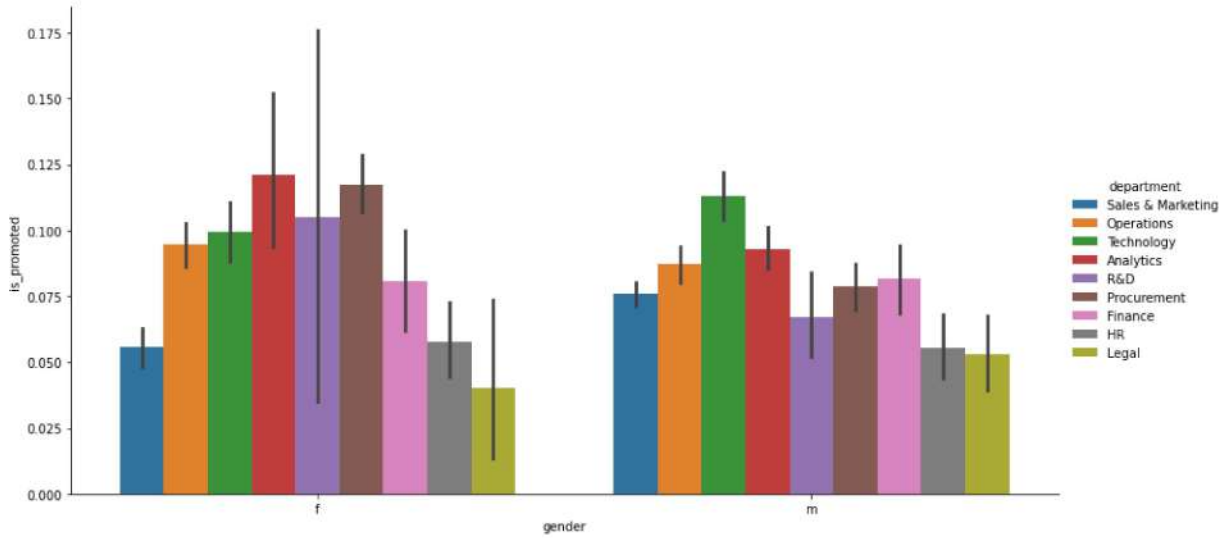
In [11]:
```
sns.catplot(x="education", y="is_promoted", kind="bar", data=train, height=4, asp
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x15a80489400>
```



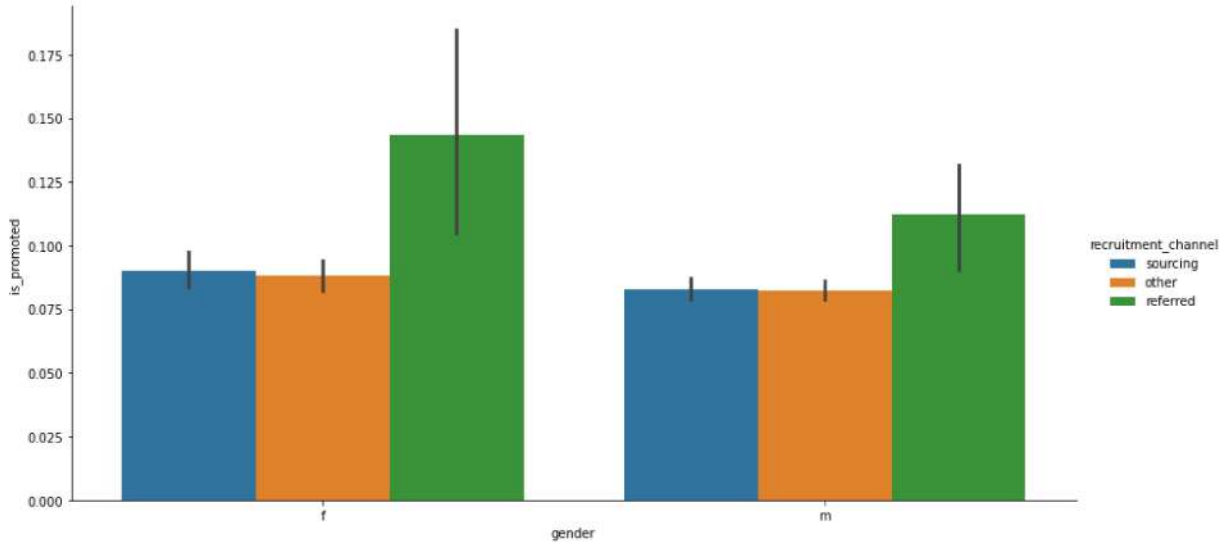
In [12]:
```
sns.catplot(x="gender", y="is_promoted", hue="department", kind="bar", data=train
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x15afaa531f0>



In [13]:
```
sns.catplot(x="gender", y="is_promoted", hue="recruitment_channel", kind="bar", d
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x15a804e2820>



In [14]:
```
# Lets check the Target Class Balance

plt.rcParams['figure.figsize'] = (15, 5)
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
sns.countplot(train['is_promoted'],)

plt.xlabel('Promoted or Not?', fontsize = 10)

plt.subplot(1, 2, 2)
train['is_promoted'].value_counts().plot(kind = 'pie', explode = [0, 0.1], autopc
                                         labels = ['1','0'], shadow = True, pctdist
plt.axis('off')

plt.suptitle('Target Class Balance', fontsize = 15)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

In [15]:
```
# missing values in training data set
#total missing values
train_total = train.isnull().sum()
# percentage of missing values in the dataset
train_percent = ((train.isnull().sum()/train.shape[0])*100).round(2)

#  total missing values in the dataset
test_total = test.isnull().sum()

# percentage of missing values in the dataset
test_percent = ((test.isnull().sum()/test.shape[0])*100).round(2)

train_missing_data = pd.concat([train_total, train_percent, test_total, test_perc
                                axis=1,
                                keys=['Train_Total', 'Train_Percent %','Test_Tota
                                sort = True)

train_missing_data.style.bar(color = ['gold'])
```

Out[15]:

	Train_Total	Train_Percent %	Test_Total	Test_Percent %
KPIs_met >80%	0	0.000000	0.000000	0.000000
age	0	0.000000	0.000000	0.000000
avg_training_score	0	0.000000	0.000000	0.000000
awards_won?	0	0.000000	0.000000	0.000000
department	0	0.000000	0.000000	0.000000
education	2409	4.400000	1034.000000	4.400000
employee_id	0	0.000000	0.000000	0.000000
gender	0	0.000000	0.000000	0.000000
is_promoted	0	0.000000	nan	nan
length_of_service	0	0.000000	0.000000	0.000000
no_of_trainings	0	0.000000	0.000000	0.000000
previous_year_rating	4124	7.520000	1812.000000	7.710000
recruitment_channel	0	0.000000	0.000000	0.000000
region	0	0.000000	0.000000	0.000000

In [16]:
```
# filling missing values of train data
train['education'] = train['education'].fillna(train['education'].mode()[0])
train['previous_year_rating'] = train['previous_year_rating'].fillna(train['previ
# null values after filling
print("Number of Missing Values Left in the Training Data :", train.isnull().sum(
```

```
Number of Missing Values Left in the Training Data : 0
```

In [17]:
```
# filling missing values of test data
test['education'] = test['education'].fillna(test['education'].mode()[0])
test['previous_year_rating'] = test['previous_year_rating'].fillna(test['previous
# null values after filling
print("Number of Missing Values Left in the Testing Data :", test.isnull().sum().
```

```
Number of Missing Values Left in the Testing Data : 0
```

OUTLIERS

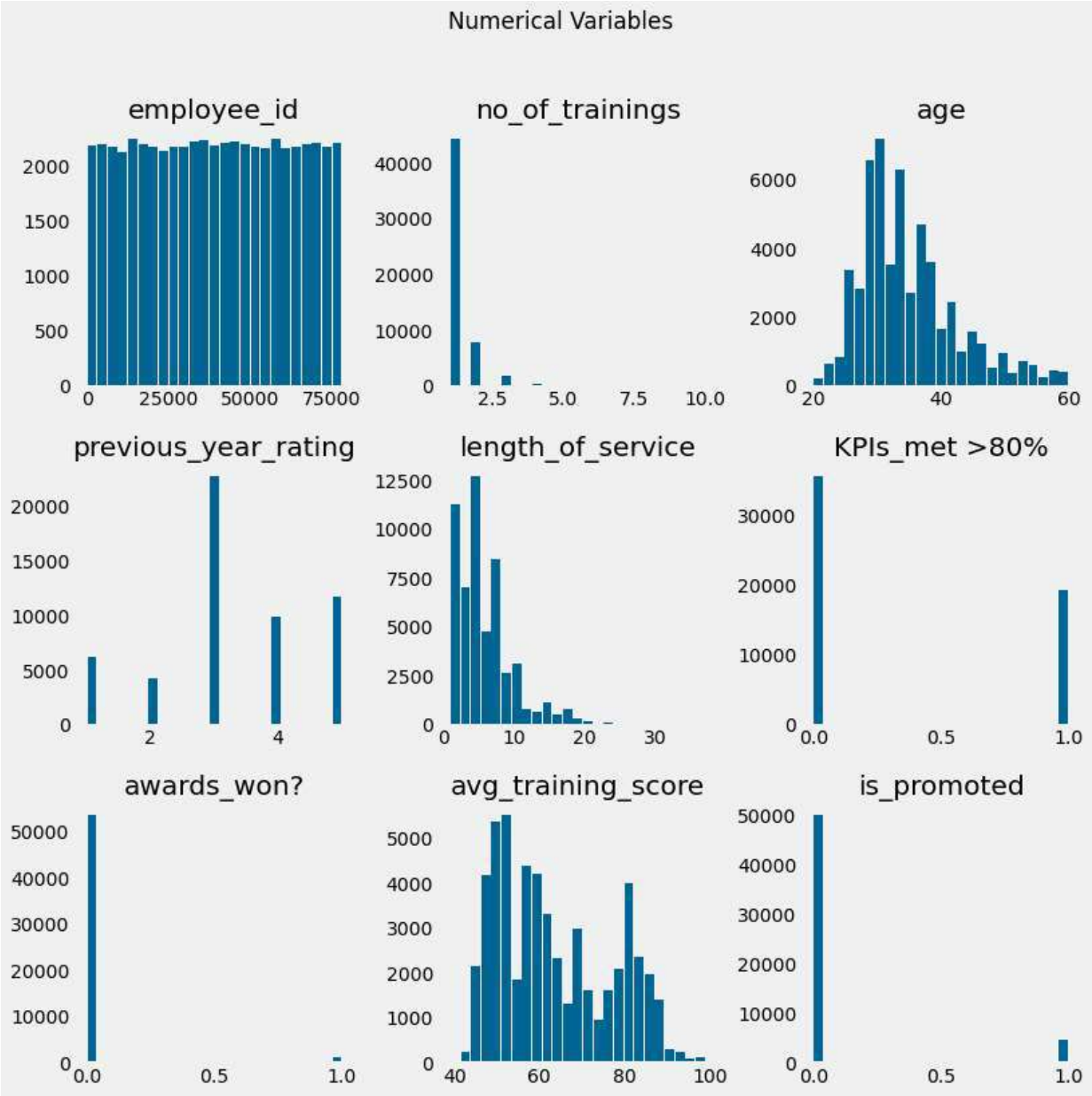
In [18]:
```
train.select_dtypes('number').head()
```

Out[18]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_w
0	65438	1	35	5.0	8	1	
1	65141	1	30	5.0	4	0	
2	7513	1	34	3.0	7	0	
3	2542	2	39	1.0	10	0	
4	48945	1	45	3.0	2	0	

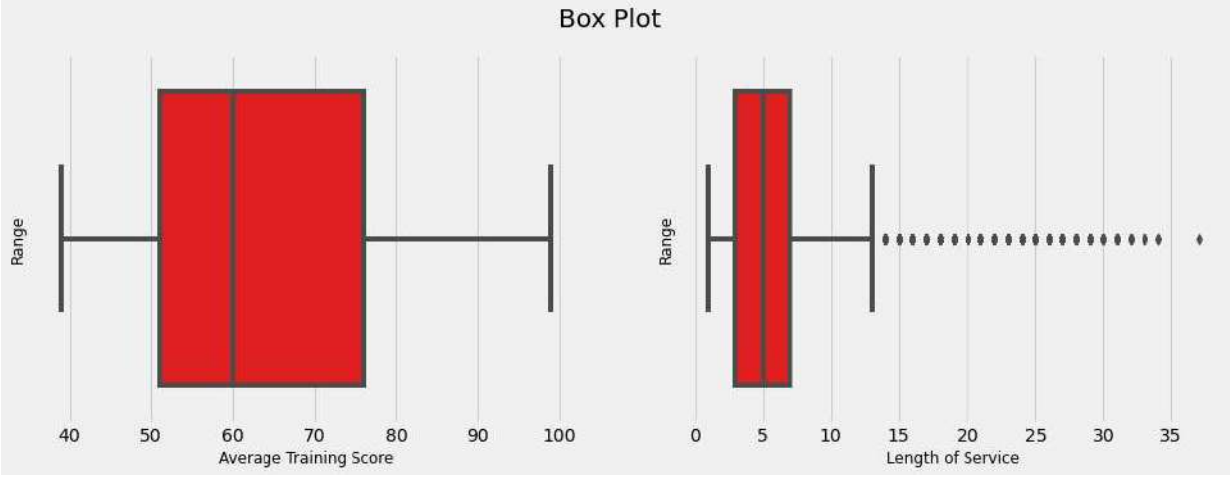
In [19]:
```
ax = train.hist(bins=25, grid=False, figsize=(12,12),color='#006494',zorder=2, rw
plt.suptitle("Numerical Variables")
```

Out[19]:
```
Text(0.5, 0.98, 'Numerical Variables')
```



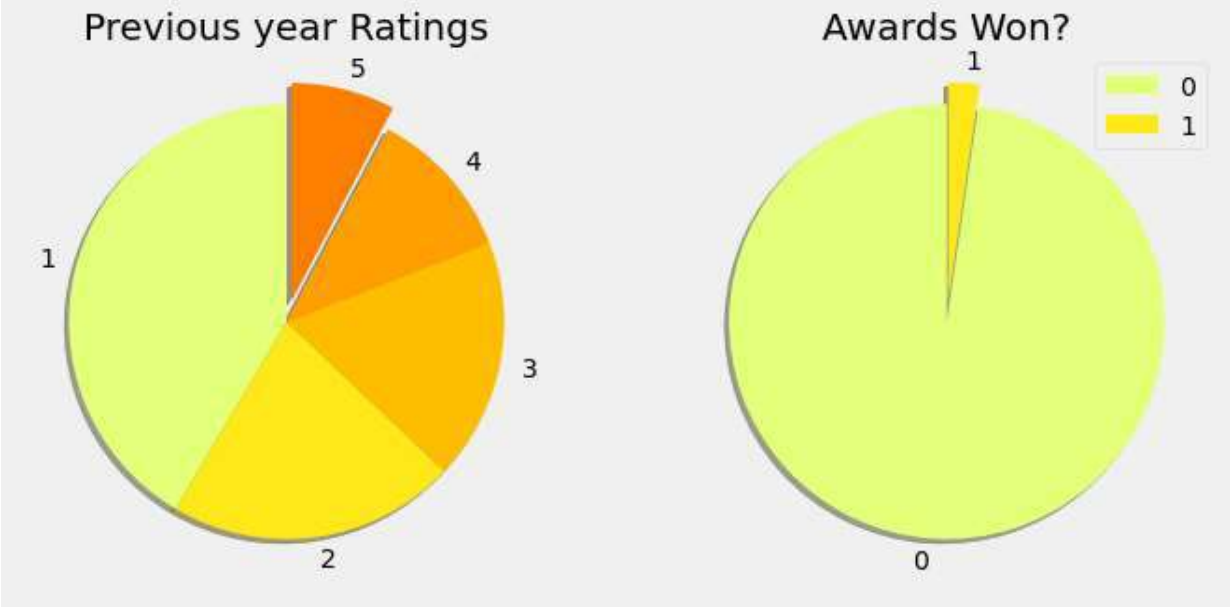
In [20]:
```
# boxplots for the columns where we suspect for outliers
plt.rcParams['figure.figsize'] = (15, 5)
plt.style.use('fivethirtyeight')

# Box plot for average training score
plt.subplot(1, 2, 1)
sns.boxplot(train['avg_training_score'], color = 'red')
plt.xlabel('Average Training Score', fontsize = 12)
plt.ylabel('Range', fontsize = 12)

# Box plot for length of service
plt.subplot(1, 2, 2)
sns.boxplot(train['length_of_service'], color = 'red')
plt.xlabel('Length of Service', fontsize = 12)
plt.ylabel('Range', fontsize = 12)

plt.suptitle('Box Plot', fontsize = 20)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



UNIVARIATE ANALYSIS

In [21]:
```
plt.rcParams['figure.figsize'] = (16,5)
plt.style.use('fivethirtyeight')
# plotting a pie chart to represent share of Previous year Rating of the Employee
plt.subplot(1, 3, 2)
labels = ['1', '2', '3', '4', '5']
sizes = train['previous_year_rating'].value_counts()
colors = plt.cm.Wistia(np.linspace(0, 1, 5))
explode = [0, 0, 0, 0, 0.1]

plt.pie(sizes, labels = labels, colors = colors, explode = explode, shadow = True
plt.title('Previous year Ratings', fontsize = 20)

# plotting a pie chart to represent share of Previous year Rating of the Employee
plt.subplot(1, 3, 3)
labels = ['0', '1']
sizes = train['awards_won?'].value_counts()
colors = plt.cm.Wistia(np.linspace(0, 1, 5))
explode = [0,0.1]
plt.pie(sizes, labels = labels, colors = colors, explode = explode, shadow = True
plt.title('Awards Won?', fontsize = 20)


plt.legend()
plt.show()
```

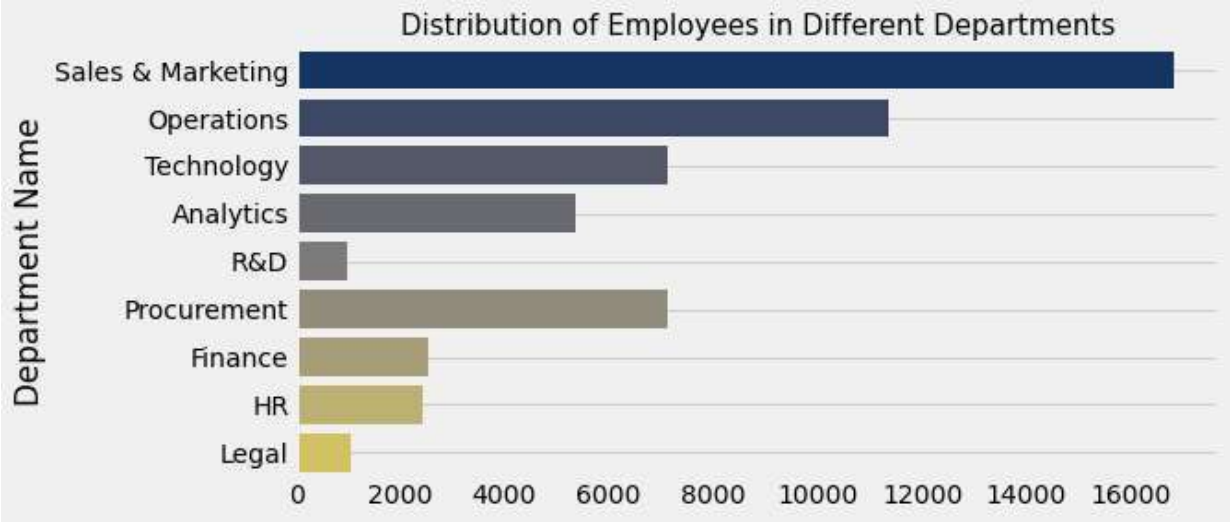


In [22]:
```
plt.rcParams['figure.figsize'] = (17, 4)
sns.countplot(train['no_of_trainings'], palette = 'spring')
plt.xlabel(' ', fontsize = 14)
plt.title('Distribution of Trainings undertaken by the Employees')
plt.show()
```

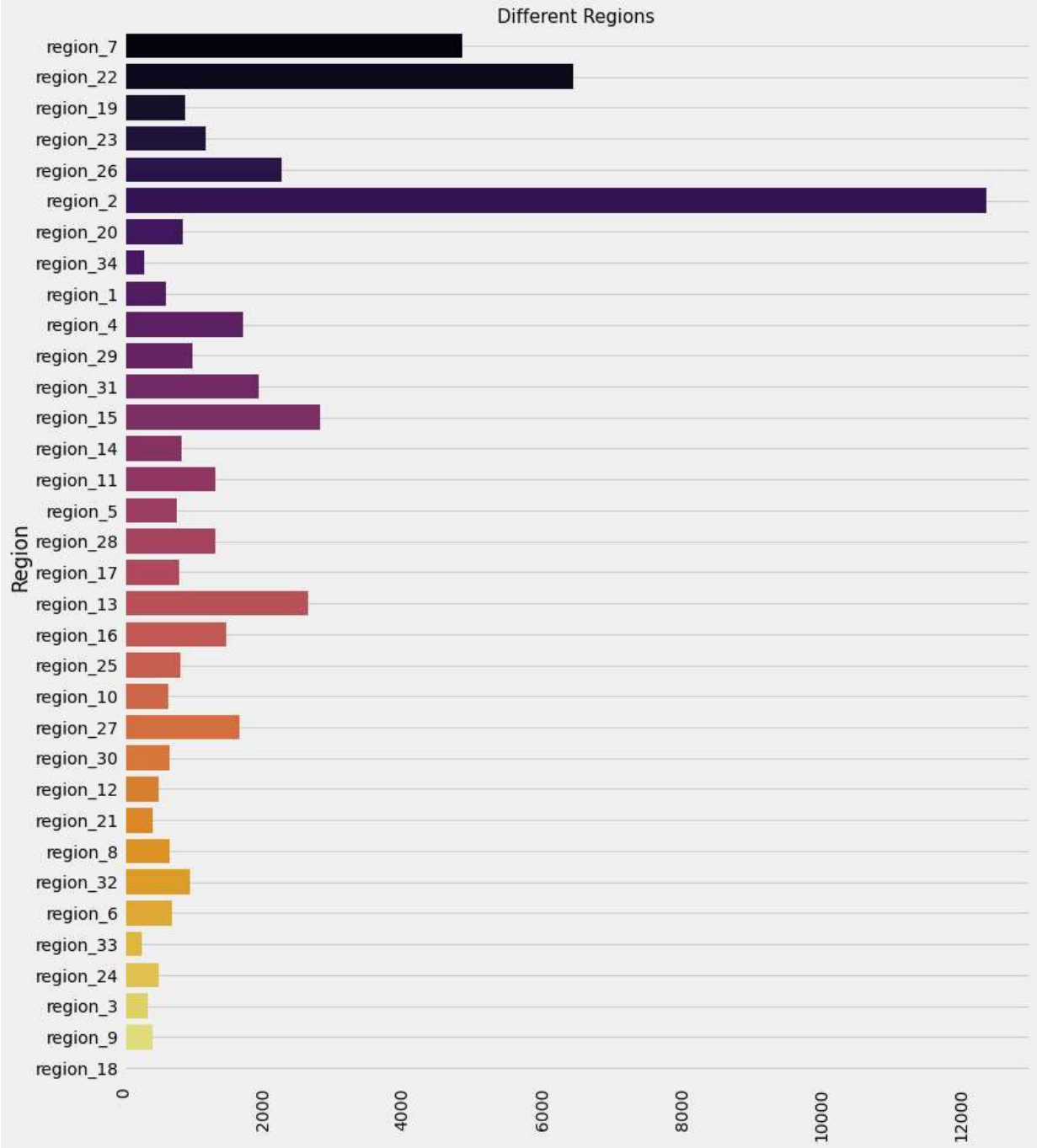
```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments w
ithout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

In [23]:
```
# Age of the Employees
plt.rcParams['figure.figsize'] = (8,4)
plt.hist(train['age'], color = 'black')
plt.title('Distribution of Age among the Employees', fontsize = 15)
plt.xlabel('Age of the Employees')
plt.grid()
plt.show()
```

In [24]:
```
plt.rcParams['figure.figsize'] = (8,4)
sns.countplot(y = train['department'], palette = 'cividis', orient = 'v')
plt.xlabel('')
plt.ylabel('Department Name')
plt.title('Distribution of Employees in Different Departments', fontsize = 15)
plt.grid()

plt.show()
```



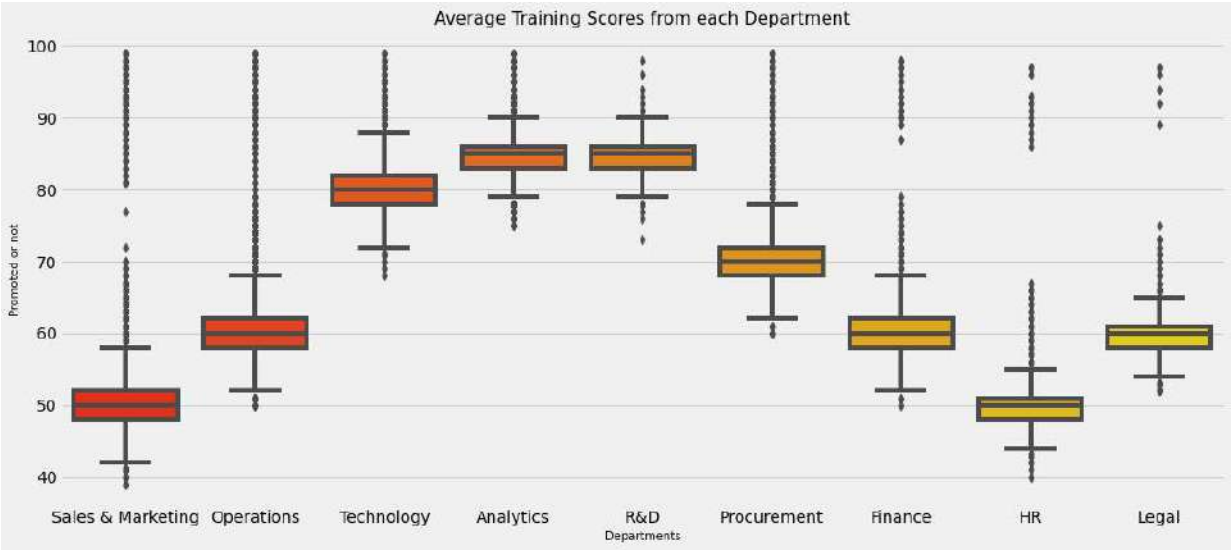
In [25]:
```
# distribution of different Regions
plt.rcParams['figure.figsize'] = (12,15)
plt.style.use('fivethirtyeight')
sns.countplot(y = train['region'], palette = 'inferno', orient = 'v')
plt.xlabel('')
plt.ylabel('Region')
plt.title('Different Regions', fontsize = 15)
plt.xticks(rotation = 90)
plt.grid()
plt.show()
```



BIVARIATE ANALYSIS

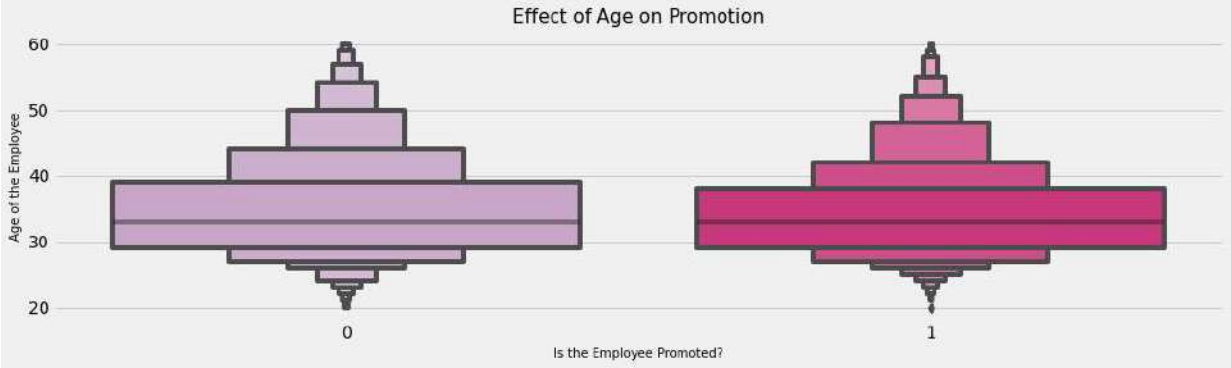
In [26]:
```
plt.rcParams['figure.figsize'] = (16, 7)
sns.boxplot(train['department'], train['avg_training_score'], palette = 'autumn')
plt.title('Average Training Scores from each Department', fontsize = 15)
plt.ylabel('Promoted or not', fontsize = 10)
plt.xlabel('Departments', fontsize = 10)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

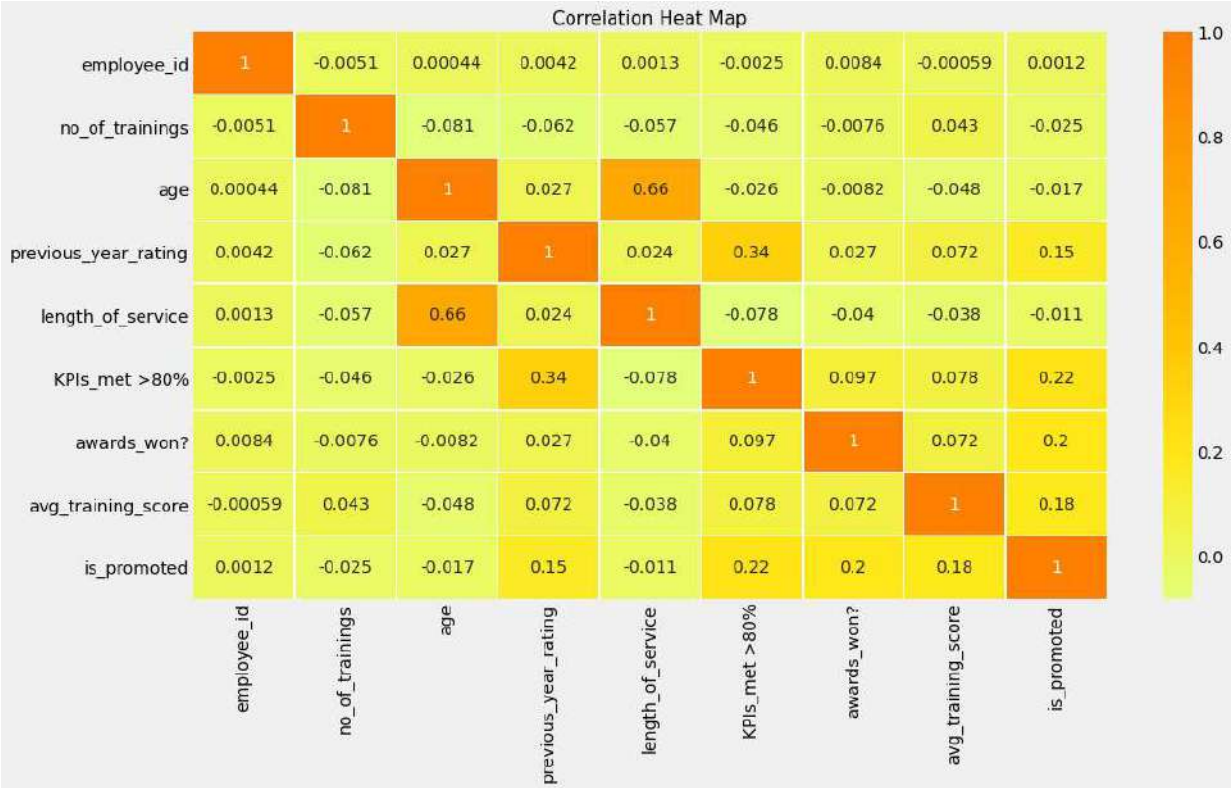


In [27]:
```
# Effect of Age on the Promotion
plt.rcParams['figure.figsize'] = (15,4)
sns.boxenplot(train['is_promoted'], train['age'], palette = 'PuRd')
plt.title('Effect of Age on Promotion', fontsize = 15)
plt.xlabel('Is the Employee Promoted?', fontsize = 10)
plt.ylabel('Age of the Employee', fontsize = 10)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

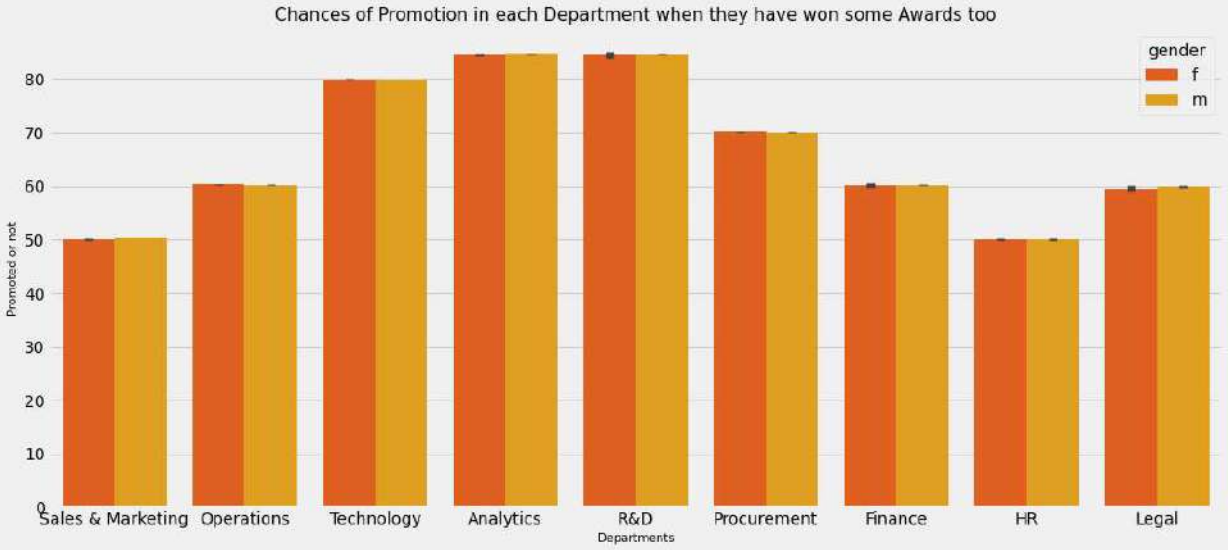


In [29]:
```
#Heat Map for the Data with respect to correlation.
plt.rcParams['figure.figsize'] = (15, 8)
sns.heatmap(train.corr(), annot = True, linewidth = 0.5, cmap = 'Wistia')
plt.title('Correlation Heat Map', fontsize = 15)
plt.show()
```



In [30]:
```
# relation of Departments and Promotions when they won awards
plt.rcParams['figure.figsize'] = (16, 7)
sns.barplot(train['department'], train['avg_training_score'], hue = train['gender
plt.title('Chances of Promotion in each Department when they have won some Awards
plt.ylabel('Promoted or not', fontsize = 10)
plt.xlabel('Departments', fontsize = 10)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [31]:
```
'''
lets check the no. of employee who did not get an award, did not acheive 80+ KPI,
and avg_training score is less than 40
but, still got promotion.
'''

train[(train['KPIs_met >80%'] == 0) & (train['previous_year_rating'] == 1.0) &
      (train['awards_won?'] == 0) & (train['avg_training_score'] < 60) & (train['
```

Out[31]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings
31860	29663	Sales & Marketing	region_22	Bachelor's	m	referred	
51374	28327	Sales & Marketing	region_2	Bachelor's	m	sourcing	

In [32]:
```
print("Before Deleting the above two rows :", train.shape)

train = train.drop(train[(train['KPIs_met >80%'] == 0) & (train['previous_year_ra
      (train['awards_won?'] == 0) & (train['avg_training_score'] < 60) & (train['

# lets check the shape of the train data after deleting the two rows
print("After Deletion of the above two rows :", train.shape)
```

```
Before Deleting the above two rows : (54808, 14)
After Deletion of the above two rows : (54806, 14)
```

In [33]:
```
# relation between the length of service and the average training score

sns.stripplot(train['avg_training_score'], train['length_of_service'], palette =
plt.title('Average Training Score vs Length of Service', fontsize = 15)
plt.xlabel('Average Training Score', fontsize = 10)
plt.ylabel('Length of Service', fontsize = 10)
plt.show()
```

```
C:\Users\Shrinidhi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

DEALING WITH CATEGORICAL COLUMNS

In [34]:
```
## Lets check the categorical columns present in the data
train.select_dtypes('object').head()
```

Out[34]:

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

In [35]:
```
# lets check the value counts for the education column
train['education'].value_counts()
```

Out[35]:
```
Bachelor's          39076
Master's & above    14925
Below Secondary       805
Name: education, dtype: int64
```

In [36]:
```
# ENCODING CATEGORICAL COLUMNS AND CONVERTING TO NUMERICAL ONES
# encoding the education in their degree of importance
train['education'] = train['education'].replace(("Master's & above", "Bachelor's'
                                                 (3, 2, 1))
test['education'] = test['education'].replace(("Master's & above", "Bachelor's",
                                                 (3, 2, 1))

# lets use Label Encoding for Gender and Department to convert them into Numerica
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
train['department'] = le.fit_transform(train['department'])
test['department'] = le.fit_transform(test['department'])
train['gender'] = le.fit_transform(train['gender'])
test['gender'] = le.fit_transform(test['gender'])

# lets check whether we still have any categorical columns left after encoding
print(train.select_dtypes('object').columns)
print(test.select_dtypes('object').columns)
```

```
Index(['region', 'recruitment_channel'], dtype='object')
Index(['region', 'recruitment_channel'], dtype='object')
```

SPLITTING THE DATA

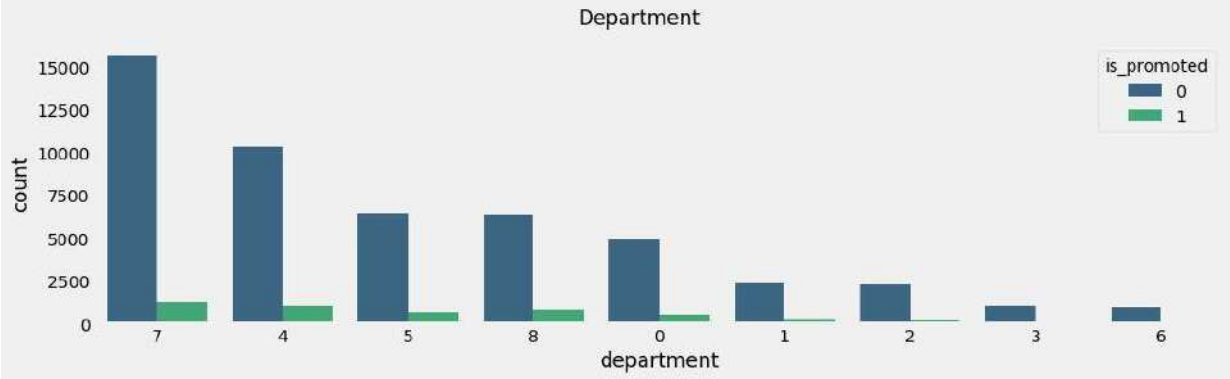
In [37]:
```
# splitting the target data from the train data
y = train['is_promoted']
x = train.drop(['is_promoted'], axis = 1)
x_test = test

# printing the shapes of these newly formed data sets
print("Shape of the x :", x.shape)
print("Shape of the y :", y.shape)
print("Shape of the x Test :", x_test.shape)
```

```
Shape of the x : (54806, 13)
Shape of the y : (54806,)
Shape of the x Test : (23490, 13)
```

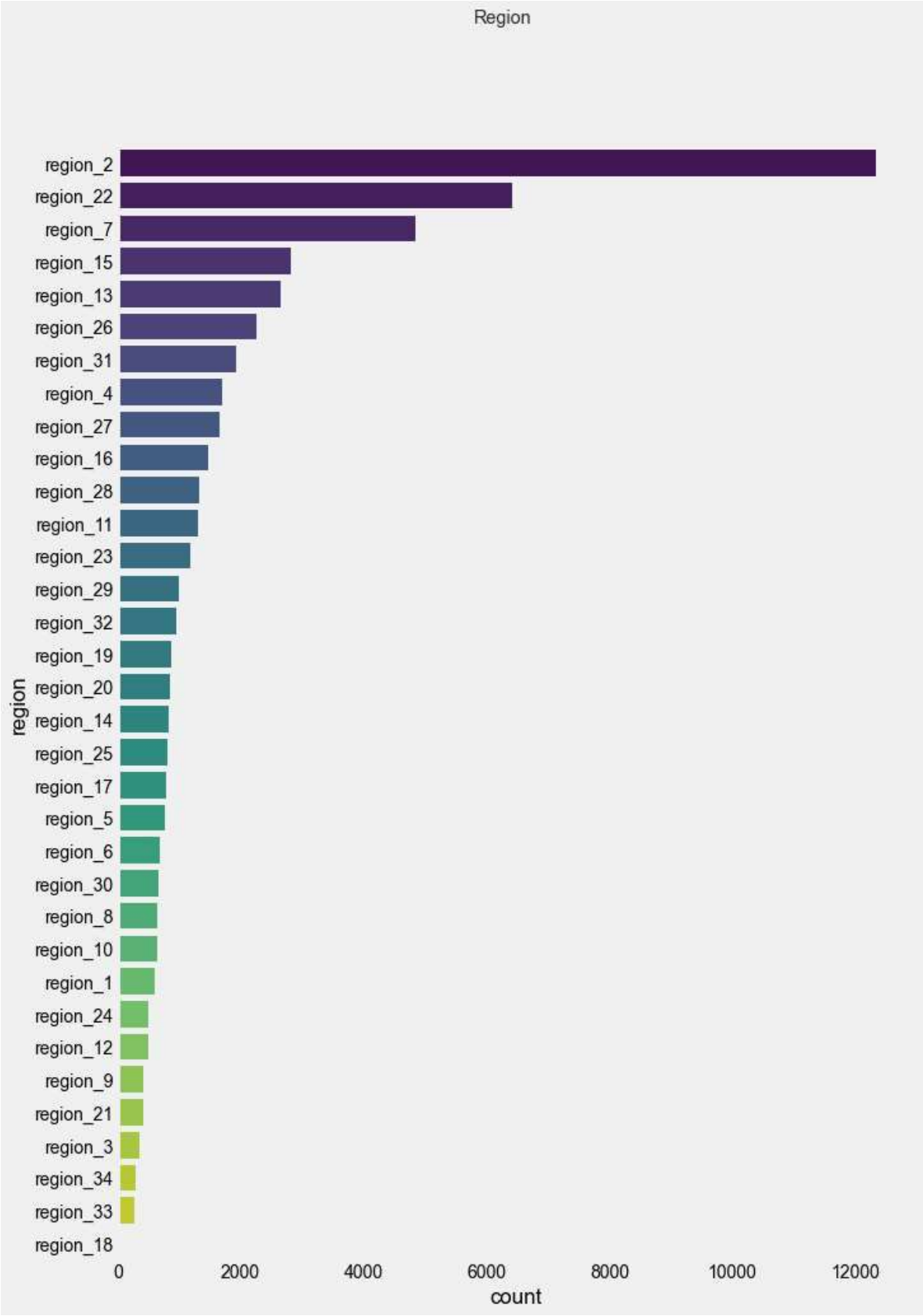
In [38]:
```
# min and max age of promotion
max_age = test['age'].max()
min_age = test['age'].min()
print('Minimum age is {} and Maximum age is  {}'.format(min_age, max_age))
```

```
Minimum age is 20 and Maximum age is  60
```

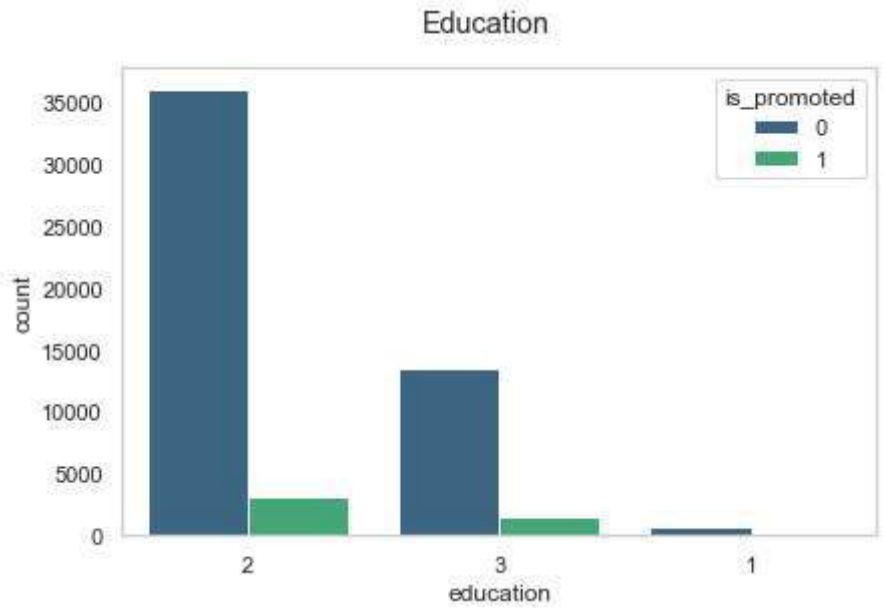
In [39]:
```
# categorical features - department
plt.figure(figsize=(15,4))
ax = sns.countplot(x="department",data=train, palette="viridis",hue="is_promoted"
ax.grid(False)
plt.suptitle("Department")
plt.show()
```



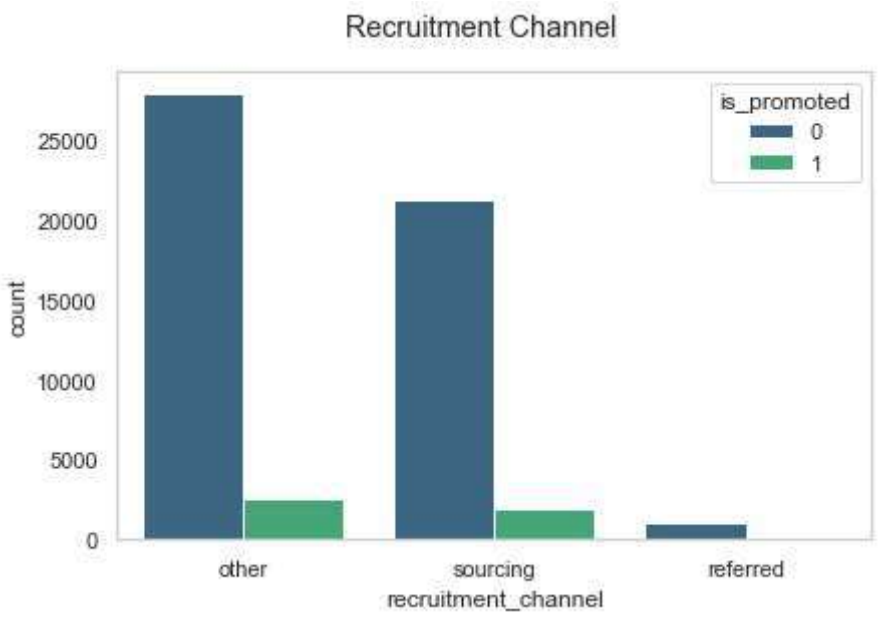
In [40]:
```
plt.figure(figsize=(10,15))
ax = sns.countplot(y="region",data=train,palette="viridis", order = train['regior
ax.grid(False)
sns.set(style="whitegrid")
plt.suptitle("Region")
plt.show()
```



In [41]:
```
plt.figure(figsize=(6,4))
ax = sns.countplot(x="education",data=train, palette="viridis",hue="is_promoted",
ax.grid(False)
plt.suptitle("Education")
plt.show()
```



In [42]:
```
plt.figure(figsize=(6,4))
ax = sns.countplot(x="recruitment_channel",data=train, palette="viridis",hue="is_
ax.grid(False)
sns.set(style="whitegrid")
plt.suptitle("Recruitment Channel")
plt.show()
```



In [43]:
```
train.is_promoted.value_counts(normalize=True)
```

Out[43]:
```
0    0.914863
1    0.085137
Name: is_promoted, dtype: float64
```

In [44]:
```
train.is_promoted.value_counts()
```

Out[44]:
```
0    50140
1     4666
Name: is_promoted, dtype: int64
```

PREPROCESSING

In [45]:
```
df = train
df = df.append(test)
df.shape
```

Out[45]: (78296, 14)

In [46]:
```
df.head(10)
```

Out[46]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	ag
0	65438	7	region_7	3	0	sourcing	1	3
1	65141	4	region_22	2	1	other	1	3
2	7513	7	region_19	2	1	sourcing	1	3
3	2542	7	region_23	2	1	other	2	3
4	48945	8	region_26	2	1	other	1	4
5	58896	0	region_2	2	1	sourcing	2	3
6	20379	4	region_20	2	0	other	1	3
7	16290	4	region_34	3	1	sourcing	1	3
8	73202	0	region_20	2	1	other	1	2
9	28911	7	region_1	3	1	sourcing	1	3

In [47]:
```
print("Education NA: ",(df.education.isna().sum()/len(df))*100)
print("Previous Year Rating NA: ",(df.previous_year_rating.isna().sum()/len(df))*
```

```
Education NA:  0.0
Previous Year Rating NA:  0.0
```

In [48]:
```
df.education.value_counts(normalize=True)
```

Out[48]:
```
2    0.711250
3    0.273692
1    0.015058
Name: education, dtype: float64
```

In [49]:
```
df['education'] = df['education'].fillna('Missing')
df.education.value_counts(normalize=True,dropna=False)
```

Out[49]:
```
2    0.711250
3    0.273692
1    0.015058
Name: education, dtype: float64
```

In [50]:
```
df.previous_year_rating.value_counts(normalize=True)
```

Out[50]:
```
3.0    0.414772
5.0    0.215056
4.0    0.180418
1.0    0.113684
2.0    0.076070
Name: previous_year_rating, dtype: float64
```

In [51]:
```
df.previous_year_rating.median()
```

Out[51]:
```
3.0
```

In [52]:
```
df['previous_year_rating'] = df['previous_year_rating'].fillna(3.0)
df.previous_year_rating.value_counts(normalize=True,dropna=False)
```

Out[52]:
```
3.0    0.414772
5.0    0.215056
4.0    0.180418
1.0    0.113684
2.0    0.076070
Name: previous_year_rating, dtype: float64
```

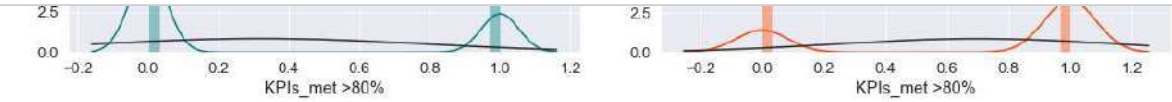
In [53]:
```
cats = [c for c in train.columns if train[c].dtypes=='object']
nums = [c for c in train.columns if c not in cats]
print(cats)
print(nums)
```

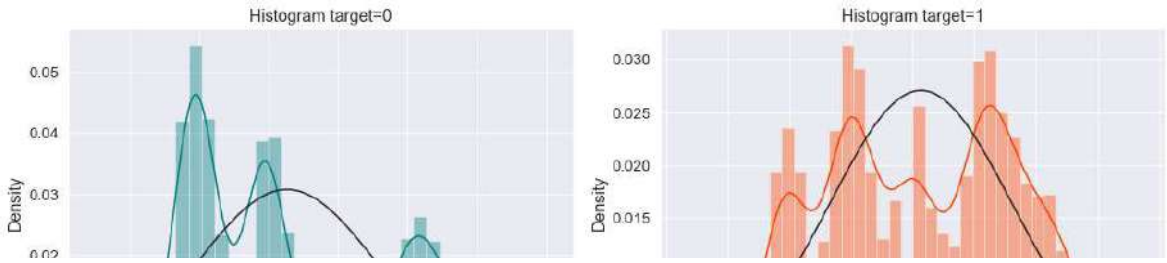
```
['region', 'recruitment_channel']
['employee_id', 'department', 'education', 'gender', 'no_of_trainings', 'age',
'previous_year_rating', 'length_of_service', 'KPIs_met >80%', 'awards_won?', 'a
vg_training_score', 'is_promoted']
```

In [54]:
```
true_cats = ['department', 'region', 'education', 'gender', 'recruitment_channel'
             'previous_year_rating','length_of_service', 'no_of_trainings']
```

In [55]:
```
true_nums = [c for c in train.columns if c not in true_cats]
true_nums.remove('is_promoted')
```

In [56]:
```
from scipy.stats import norm

sns.set()
sns.set_style("darkgrid")
sns.set_context("paper")
sns.set_context("paper", font_scale=1.5, rc={"lines.linewidth": 1.5})
for c in true_nums:

    fig = plt.figure(constrained_layout=True, figsize=(14,5))
    grid = gridspec.GridSpec(ncols=2, nrows=1, figure=fig)
    ax1 = fig.add_subplot(grid[0, 0])
    ax1.set_title('Histogram target=0')
    sns.distplot(train[train.is_promoted==0].loc[:,c].dropna(),bins=30, fit=norm,
    ax2 = fig.add_subplot(grid[0, 1])
    ax2.set_title('Histogram target=1')
    sns.distplot(train[train.is_promoted==1].loc[:,c].dropna(), bins=30, fit=norm
```



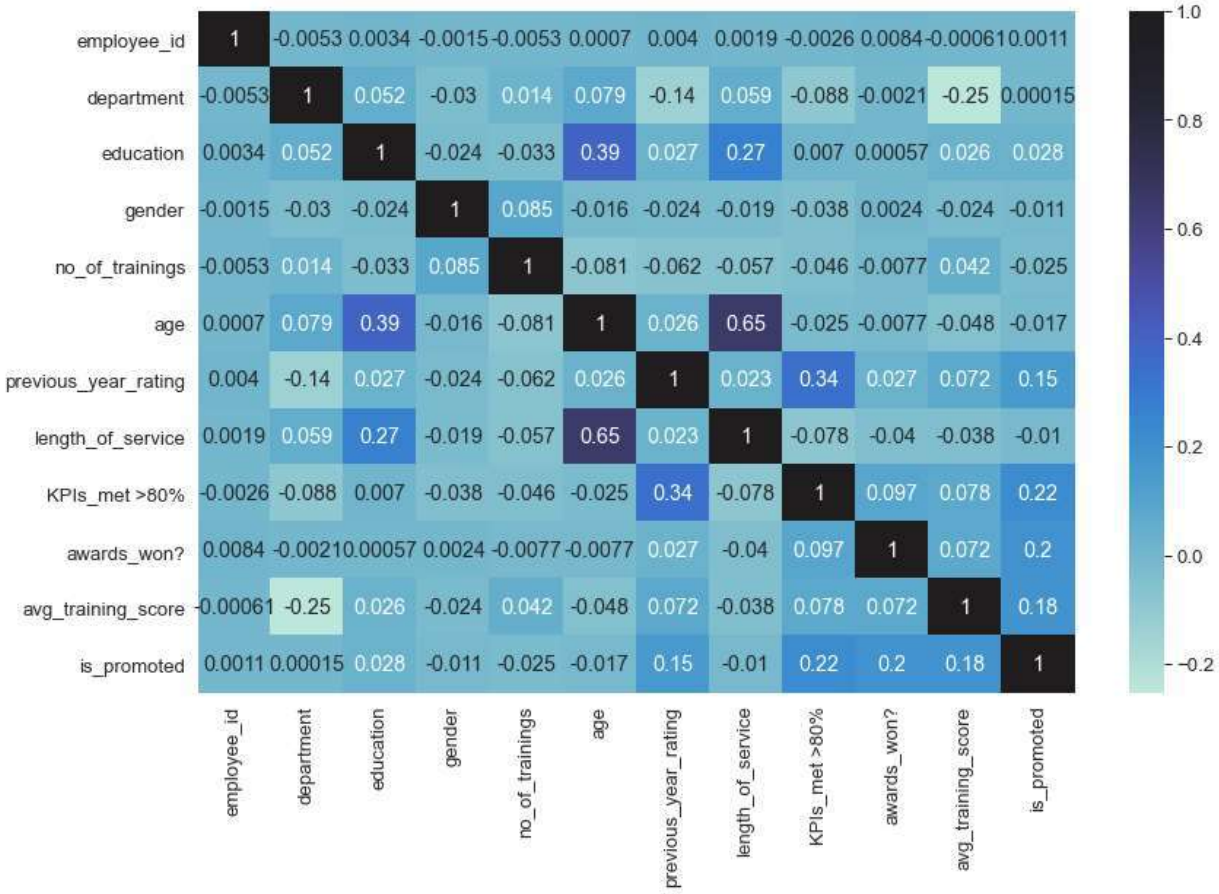


OUTLIERS

In [57]:
```
train = train[train.length_of_service<30]
```

In [58]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(pd.DataFrame(train, columns=train.columns).corr(), annot=True, center
```

Out[58]: <AxesSubplot:>



In [59]:
```
# IMPUTATION

from sklearn.impute import SimpleImputer
target = train.pop('is_promoted')

data = pd.concat([train, test], axis=0)
si = SimpleImputer(strategy= 'most_frequent')
data['education'] = si.fit_transform(data.education.values.reshape(-1, 1))
si = SimpleImputer(strategy='mean')
data['previous_year_rating'] = si.fit_transform(data.previous_year_rating.values.
data.isnull().sum()
```

Out[59]:
```
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
KPIs_met >80%         0
awards_won?           0
avg_training_score    0
dtype: int64
```

LABEL ENCODER

In [60]:
```
from sklearn.preprocessing import LabelEncoder
datal = data.copy()
for c in cats:
    le = LabelEncoder()
    datal[c] = le.fit_transform(datal[c])

train_le = datal.iloc[:len(train)]
test_le = datal.iloc[len(train):]
train_le.shape, test_le.shape
```

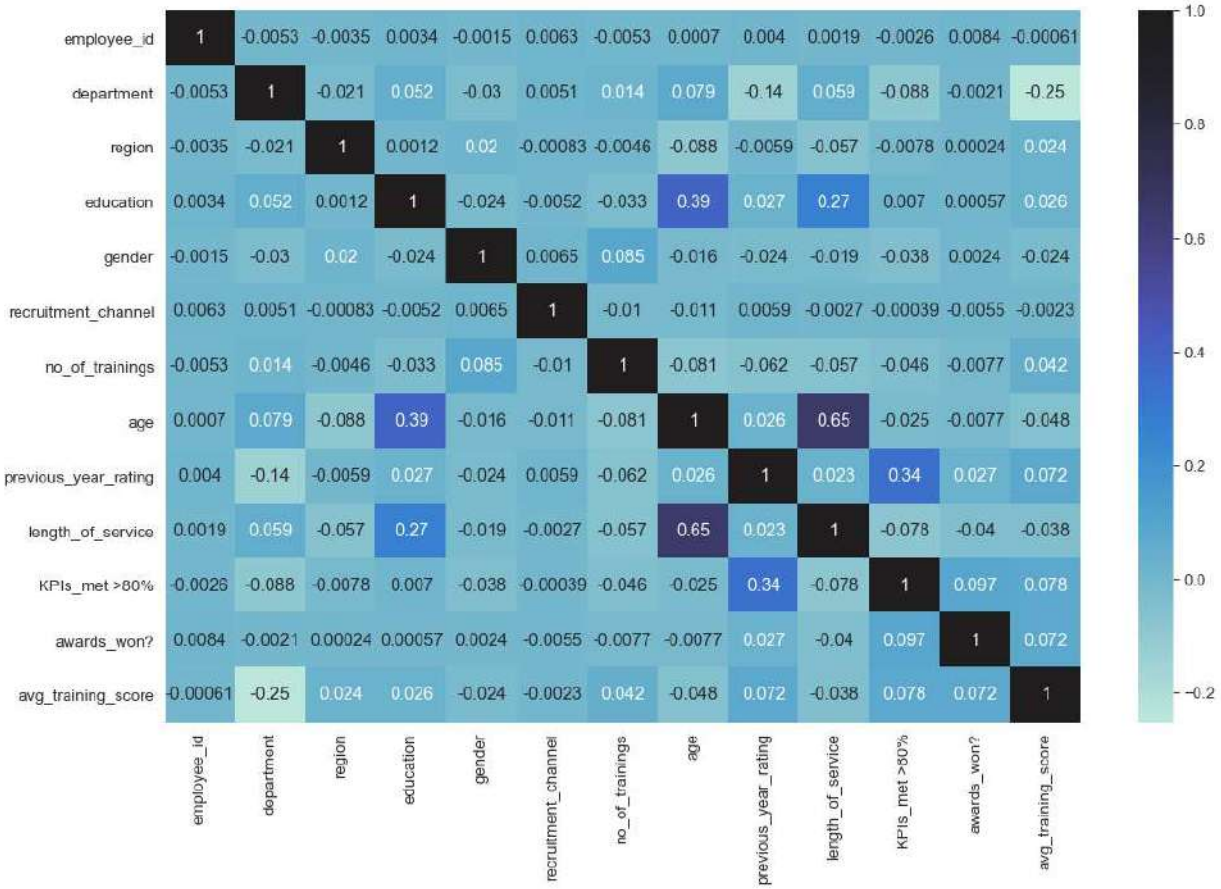
Out[60]:
```
((54750, 13), (23490, 13))
```

In [61]:
```
data_oh = pd.get_dummies(data)
train_oh = data_oh.iloc[:len(train)]
test_oh = data_oh.iloc[len(train):]
train_oh.shape, test_oh.shape
```

Out[61]:
```
((54750, 48), (23490, 48))
```

In [62]:
```
plt.figure(figsize=(15, 10))
sns.heatmap(train_le.corr(), annot=True, center=True)
```

Out[62]: <AxesSubplot:>



In [63]:
```
from sklearn.preprocessing import StandardScaler, RobustScaler
ss = StandardScaler()
train_le = ss.fit_transform(train_le)
test_le = ss.fit_transform(test_le)

train_oh = ss.fit_transform(train_oh)
test_oh = ss.fit_transform(test_oh)
```

In [64]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import roc_auc_score
scores = []
oof = np.zeros(len(train))
y_le = target.values
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for fold_, (train_ind, val_ind) in enumerate(folds.split(train_le, y_le)):
    print('fold:', fold_)
    X_tr, X_test = train_le[train_ind], train_le[val_ind]
    y_tr, y_test = y_le[train_ind], y_le[val_ind]
    clf = LogisticRegression(max_iter=200, random_state=2020)
    clf.fit(X_tr, y_tr)
    oof[val_ind]= clf.predict_proba(X_test)[:, 1]
    y = clf.predict_proba(X_tr)[:,1]
    print('train:',roc_auc_score(y_tr, y),'val :' , roc_auc_score(y_test, (oof[va
    print(20 * '-')

    scores.append(roc_auc_score(y_test, oof[val_ind]))



print('log reg  roc_auc=  ', np.mean(scores))
np.save('oof_rf', oof)
```

```
fold: 0
train: 0.8057681509848473 val : 0.8123475383794148
--------------------
fold: 1
train: 0.8075738140236636 val : 0.8036875898061601
--------------------
fold: 2
train: 0.8059223035291457 val : 0.8113847402609562
--------------------
fold: 3
train: 0.809550217414431 val : 0.7965170892227051
--------------------
fold: 4
train: 0.8061284039793633 val : 0.8068476515967871
--------------------
log reg  roc_auc=   0.8061569218532046
```

In [65]:
```
from sklearn.metrics import *
oof_rnd = np.where(oof > 0.5, 1, 0)

f1_score(target, oof_rnd)
```

Out[65]: 0.12675785012521673

In [66]:
```
recall_score(target, oof_rnd)
```

Out[66]: 0.07055543641432555

ONE HOT ENCODER

In [67]:
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(categories='auto')
ohe_arr = ohe.fit_transform(df[['department','region','education','gender','recru
ohe_labels = ohe.get_feature_names(['department','region','education','gender','r
ohe_df = pd.DataFrame(ohe_arr, columns= ohe_labels)
ohe_df.head()
```

Out[67]:

	department_0	department_1	department_2	department_3	department_4	department_5	departm
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	1.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 51 columns

LOGISTIC REGRESSION WITH ONE-HOT ENCODED DATA

In [68]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import roc_auc_score
scores = []
oof = np.zeros(len(train_oh))
y_le = target.values
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
for fold_, (train_ind, val_ind) in enumerate(folds.split(train_oh, y_le)):
    print('fold:', fold_)
    X_tr, X_test = train_oh[train_ind], train_oh[val_ind]
    y_tr, y_test = y_le[train_ind], y_le[val_ind]
    clf = LogisticRegression(max_iter=200, random_state=2020)
    clf.fit(X_tr, y_tr)
    oof[val_ind]= clf.predict_proba(X_test)[:, 1]
    y = clf.predict_proba(X_tr)[:,1]
    print('train:',roc_auc_score(y_tr, y),'val :' , roc_auc_score(y_test, (oof[va
    print(20 * '-')

    scores.append(roc_auc_score(y_test, oof[val_ind]))

    print('log reg  roc_auc=  ', np.mean(scores))
np.save('oof_rf', oof)
```

```
fold: 0
train: 0.8142757281614703 val : 0.8174469431418298
--------------------
log reg  roc_auc=   0.8174469431418298
fold: 1
train: 0.8153128228504302 val : 0.8110749363591886
--------------------
log reg  roc_auc=   0.8142609397505092
fold: 2
train: 0.8139275637607377 val : 0.8169959942695488
--------------------
log reg  roc_auc=   0.815172624590189
fold: 3
train: 0.8172908201532039 val : 0.8043991880469867
--------------------
log reg  roc_auc=   0.8124792654543884
fold: 4
train: 0.8147134672500069 val : 0.8120546624864204
--------------------
log reg  roc_auc=   0.8123943448607948
```

In [69]:
```
oof_rnd = np.where(oof > 0.5, 1, 0)
f1_score(target, oof_rnd)
```

Out[69]: 0.13690815744438106

In [70]:
```
recall_score(target, oof_rnd)
```

Out[70]: 0.07720351704911002

In [71]:
```
class_names=np.array(['0','1'])

# Function to plot the confusion Matrix
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):

            plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

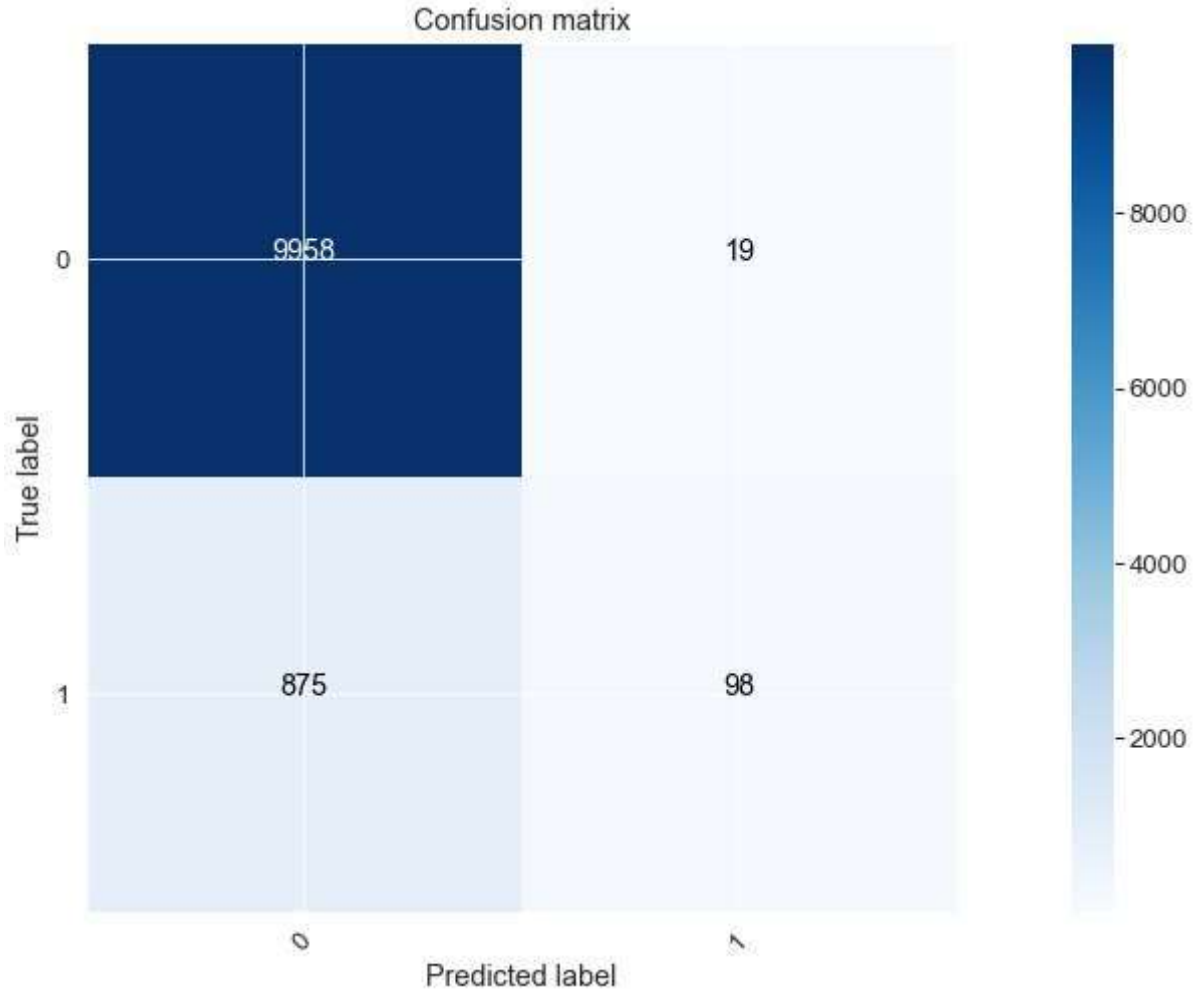
In [72]:
```
y = target
x = train_le


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,random_st

from sklearn.svm import SVC
svm=SVC(random_state=1)
svm.fit(x_train,y_train)
print("train accuracy:",svm.score(x_train,y_train))
print("test accuracy:",svm.score(x_test,y_test))
```

```
train accuracy: 0.9242922374429223
test accuracy: 0.9183561643835616
```

In [73]:
```
from sklearn.metrics import classification_report
prediction_SVM_all = svm.predict(x_test)
print(classification_report(y_test, prediction_SVM_all, target_names=class_names))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      9977
           1       0.84      0.10      0.18       973

    accuracy                           0.92     10950
   macro avg       0.88      0.55      0.57     10950
weighted avg       0.91      0.92      0.89     10950
```

In [74]:
```
cm = confusion_matrix(y_test, prediction_SVM_all)
plot_confusion_matrix(cm,class_names)
```



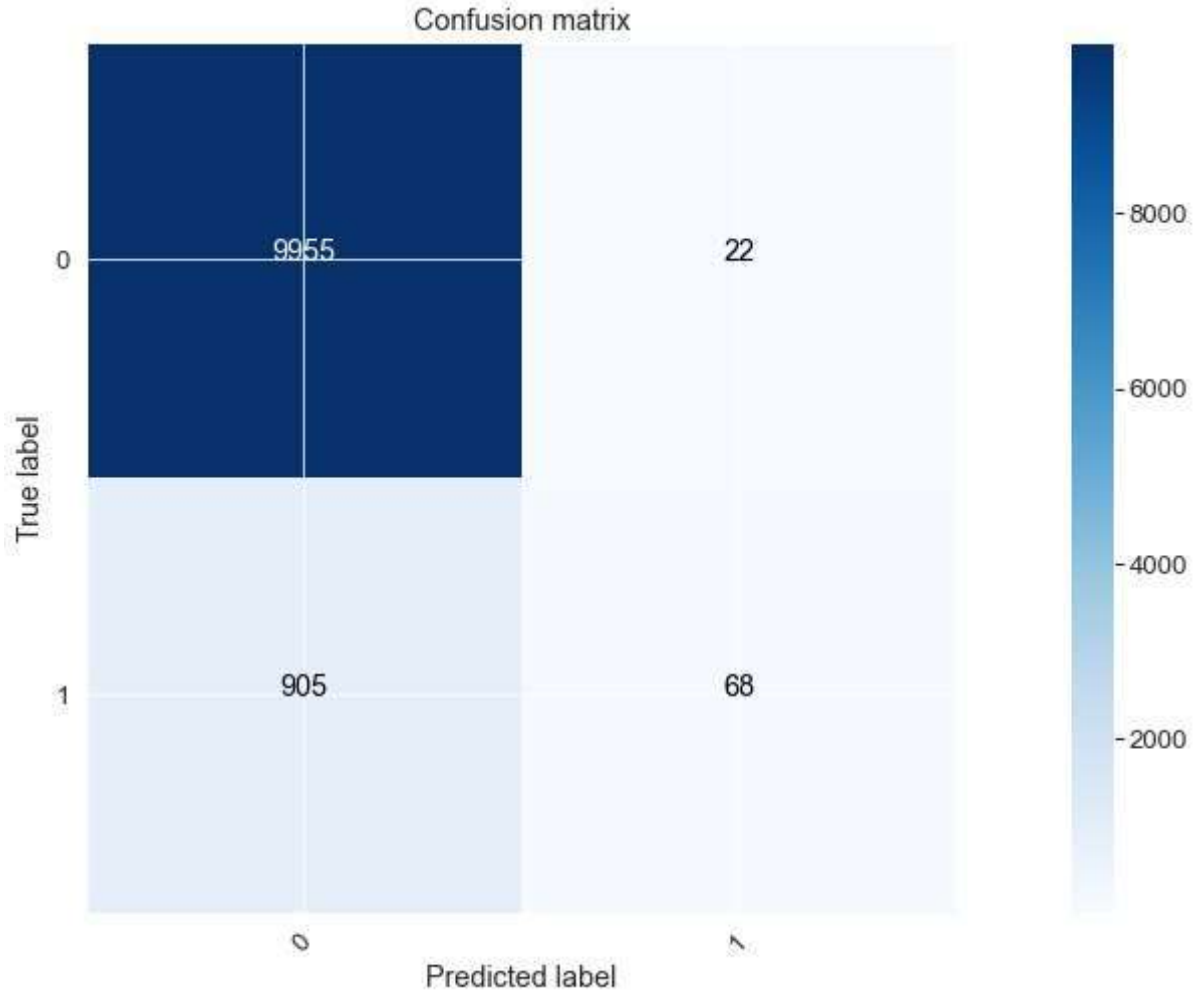
In [75]:
```
y = target
x = train_oh


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,random_st

from sklearn.svm import SVC
svm=SVC(random_state=1)
svm.fit(x_train,y_train)
print("train accuracy:",svm.score(x_train,y_train))
print("test accuracy:",svm.score(x_test,y_test))
```

```
train accuracy: 0.9214383561643835
test accuracy: 0.9153424657534247
```

In [76]:
```
prediction_SVM_all = svm.predict(x_test)
print(classification_report(y_test, prediction_SVM_all, target_names=class_names)
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      9977
           1       0.76      0.07      0.13       973

    accuracy                           0.92     10950
   macro avg       0.84      0.53      0.54     10950
weighted avg       0.90      0.92      0.88     10950
```

In [77]:
```
prediction_SVM_all = svm.predict(x_test)
cm = confusion_matrix(y_test, prediction_SVM_all)
plot_confusion_matrix(cm,class_names)
```



RANDOM FOREST CLASSIFIER

In [78]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(criterion='gini',
                            n_estimators=700,
                            min_samples_split=10,
                            min_samples_leaf=1,
                            max_features='auto',
                            oob_score=True,
                            random_state=1,
                            n_jobs=-1)
rf.fit(x_train,y_train)
print("%.4f" % rf.oob_score_)
```

```
0.9289
```

In [79]:
```
rf.feature_importances_
```

Out[79]:
```
array([0.12362784, 0.08263987, 0.01213178, 0.01483096, 0.01791137,
       0.07807639, 0.05487927, 0.06566871, 0.07517078, 0.05232381,
       0.27187593, 0.00320524, 0.0033694 , 0.00374214, 0.00163108,
       0.00631354, 0.00356816, 0.00630844, 0.00398663, 0.00501952,
       0.00036816, 0.00318262, 0.00833055, 0.00287873, 0.00153861,
       0.00740033, 0.00551289, 0.00181316, 0.00452875, 0.00505859,
       0.00483568, 0.00487281, 0.00291645, 0.00233377, 0.00371599,
       0.0041359 , 0.00262568, 0.00142875, 0.00074798, 0.00619189,
       0.00214385, 0.0021457 , 0.00739431, 0.00309306, 0.00098306,
       0.00950868, 0.00435639, 0.00967682])
```

In [80]:
```
pd.concat((pd.DataFrame(train.iloc[:, 1:].columns, columns = ['variable']),
           pd.DataFrame(rf.feature_importances_, columns = ['importance'])),
          axis = 1).sort_values(by='importance', ascending = False)[:5]
```

Out[80]:

	variable	importance
10	awards_won?	0.271876
0	department	0.123628
1	region	0.082640
5	no_of_trainings	0.078076
8	length_of_service	0.075171

NORMALIZATION

In [81]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['no_of_trainings','age','previous_year_rating','length_of_service','awards_wo
df.head()
```

Out[81]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	
0	65438	7	region_7	3	0	sourcing	0.000000	0.
1	65141	4	region_22	2	1	other	0.000000	0.
2	7513	7	region_19	2	1	sourcing	0.000000	0.
3	2542	7	region_23	2	1	other	0.111111	0.
4	48945	8	region_26	2	1	other	0.000000	0.

SEED

In [82]:
```
seed_value = 12321
import os
os.environ['PYTHONHASHSEED']=str(seed_value)
import random
random.seed(seed_value)
np.random.seed(seed_value)
```

VALIDATION DATA

In [83]:
```
train, test = df[~df['is_promoted'].isnull()], df[df['is_promoted'].isnull()]
print("Shape of Train Dataset: ",train.shape)
print("Shape of Test Dataset: ",test.shape)
```

```
Shape of Train Dataset:  (54806, 14)
Shape of Test Dataset:  (23490, 14)
```

In [84]:
```
train.drop(columns=['employee_id'],inplace=True)
print("Shape of Train Dataset: ",train.shape)
```

```
Shape of Train Dataset:  (54806, 13)

C:\Users\Shrinidhi\anaconda3\lib\site-packages\pandas\core\frame.py:4308: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  return super().drop(
```

In [85]:
```
# train valid split
from sklearn.model_selection import train_test_split
X_train, X_valid, y_train, y_valid = train_test_split(train.drop(columns=['is_pro
print("Shape of X Train Dataset: ",X_train.shape)
print("Shape of Y Train Dataset: ", y_train.shape)
print("Shape of X Valid Dataset: ",X_valid.shape)
print("Shape of Y Valid Dataset: ",y_valid.shape)
```

```
Shape of X Train Dataset:  (38364, 12)
Shape of Y Train Dataset:  (38364,)
Shape of X Valid Dataset:  (16442, 12)
Shape of Y Valid Dataset:  (16442,)
```

STARTUP CASE STUDY

In [86]:
```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\shrinidhi\anaconda3\lib\si
te-packages (1.8.1)
Requirement already satisfied: matplotlib in c:\users\shrinidhi\anaconda3\lib\s
ite-packages (from wordcloud) (3.3.4)
Requirement already satisfied: pillow in c:\users\shrinidhi\anaconda3\lib\site-
packages (from wordcloud) (8.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\shrinidhi\anaconda3\lib
\site-packages (from wordcloud) (1.20.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\shrinidhi\anaco
nda3\lib\site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\u
sers\shrinidhi\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\shrinidhi\anaconda
3\lib\site-packages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\shrinidhi\anaconda3\lib
\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: six in c:\users\shrinidhi\anaconda3\lib\site-pac
kages (from cycler>=0.10->matplotlib->wordcloud) (1.15.0)
```

In [87]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import os
import re
from wordcloud import WordCloud
import numpy as np
```

In [88]:
```
data=pd.read_csv('C:/Users/Shrinidhi/Downloads/Startups-Case-study (1)/EDA on Fir
```

In [89]:
```
data.head(10)
```

Out[89]:

	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name
0	1	09/01/2020	BYJU'S	E-Tech	E-learning	Bengaluru	Tiger Globa Managemen
1	2	13/01/2020	Shuttl	Transportation	App based shuttle service	Gurgaon	Susquehanna Growth Equit
2	3	09/01/2020	Mamaearth	E-commerce	Retailer of baby and toddler products	Bengaluru	Sequoia Capital India
3	4	02/01/2020	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatuma
4	5	02/01/2020	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprou Ventur Partner
5	6	13/01/2020	Pando	Logistics	Open-market, freight management platform	Chennai	Chirata Venture
6	7	10/01/2020	Zomato	Hospitality	Online Food Delivery Platform	Gurgaon	Ant Financia
7	8	12/12/2019	Ecozen	Technology	Agritech	Pune	Sathgur Catalyze Advisor
8	9	06/12/2019	CarDekho	E-Commerce	Automobile	Gurgaon	Ping A Globa Voyager Fun
9	10	03/12/2019	Dhruva Space	Aerospace	Satellite Communication	Bengaluru	Mumba Angels Ravikant Redd

In [90]:
```
data.shape
```

Out[90]:
```
(3044, 10)
```

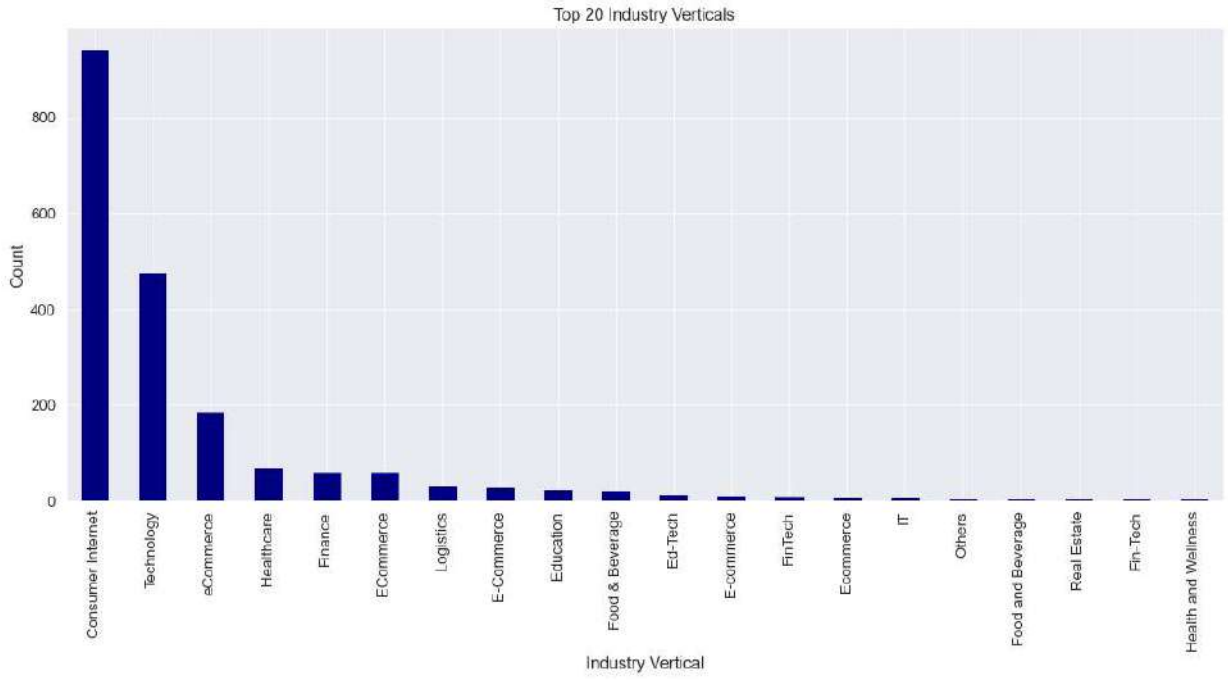
In [91]:
```
data.columns
```

Out[91]:
```
Index(['Sr No', 'Date dd/mm/yyyy', 'Startup Name', 'Industry Vertical',
       'SubVertical', 'City  Location', 'Investors Name', 'InvestmentnType',
       'Amount in USD', 'Remarks'],
      dtype='object')
```

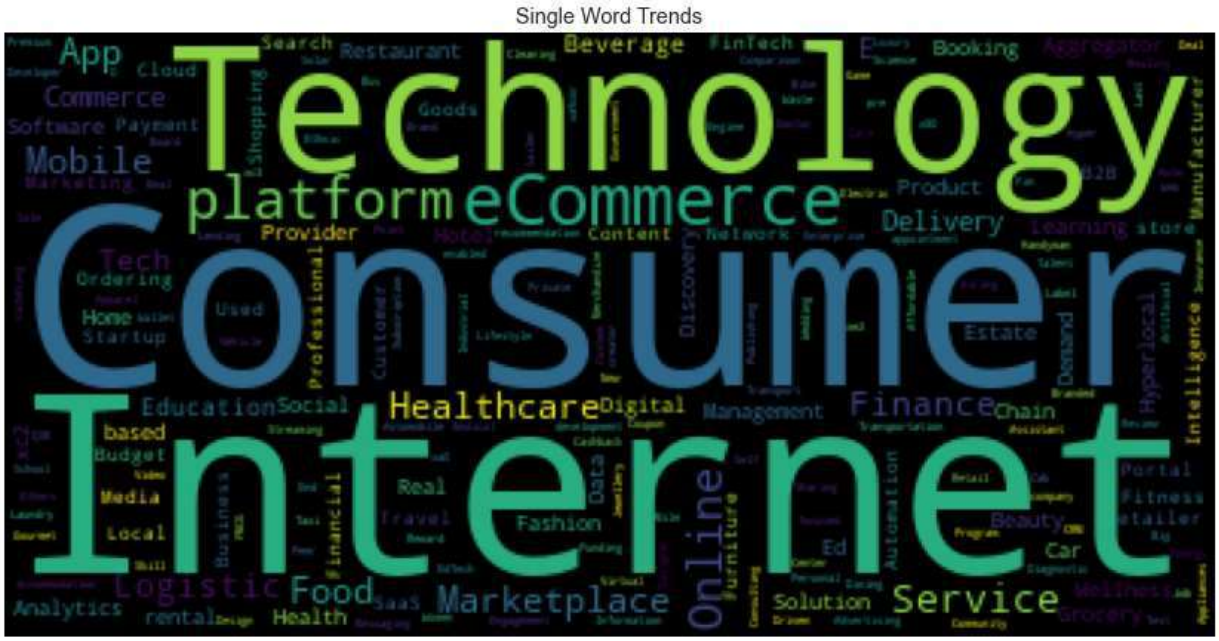
In [92]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3044 entries, 0 to 3043
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Sr No              3044 non-null   int64 
 1   Date dd/mm/yyyy    3044 non-null   object
 2   Startup Name       3044 non-null   object
 3   Industry Vertical  2873 non-null   object
 4   SubVertical        2108 non-null   object
 5   City  Location     2864 non-null   object
 6   Investors Name     3020 non-null   object
 7   InvestmentnType    3040 non-null   object
 8   Amount in USD      2084 non-null   object
 9   Remarks            419 non-null    object
dtypes: int64(1), object(9)
memory usage: 237.9+ KB
```

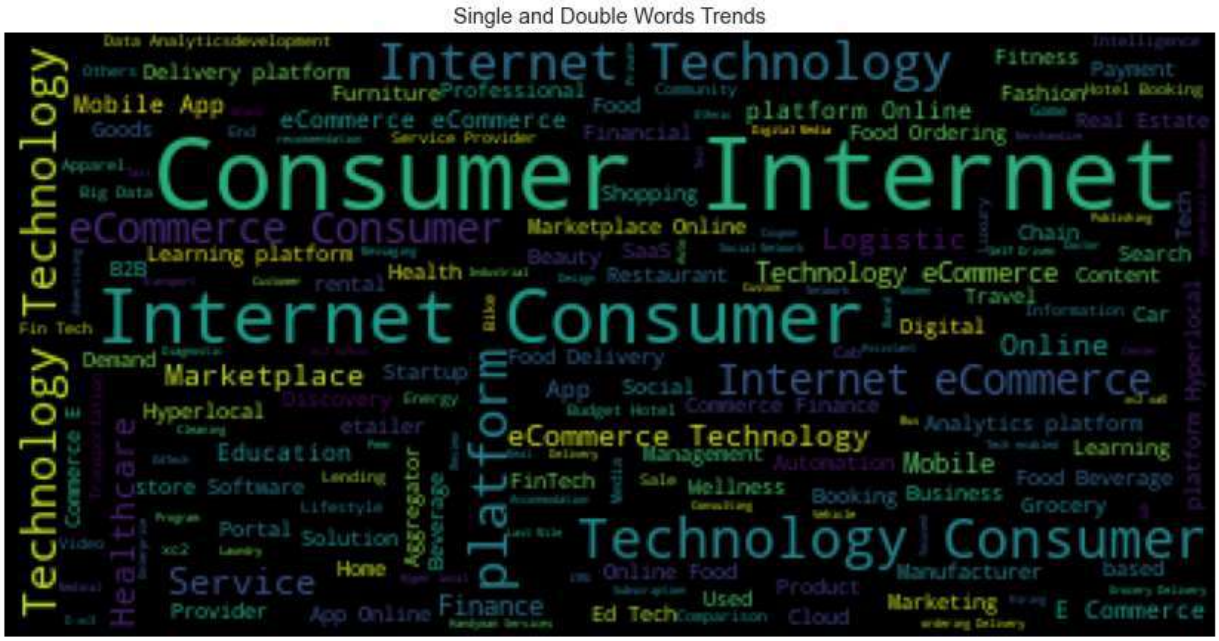
In [93]:
```
data['Industry Vertical'].value_counts()[:20].plot.bar(color='navy')
plt.title('Top 20 Industry Verticals')
plt.xlabel('Industry Vertical')
plt.ylabel('Count')
plt.show()
```



In [94]:
```
wordcloud = WordCloud(collocations=False) # Disable collocations (bigrams) of two
wordcloud.generate(data['Industry Vertical'].str.cat(sep=' '))
plt.imshow(wordcloud) # 'Plot' wordcloud
plt.axis('off') # Hide axes
plt.title('Single Word Trends')
plt.show()
```



In [95]:
```
wordcloud = WordCloud()
wordcloud.generate(data['Industry Vertical'].str.cat(sep=' '))
plt.imshow(wordcloud) # 'Plot' wordcloud
plt.axis('off') # Hide axes
plt.title('Single and Double Words Trends')
plt.show()
```



In [96]:
```
data['City  Location'].nunique()
```

Out[96]: 112

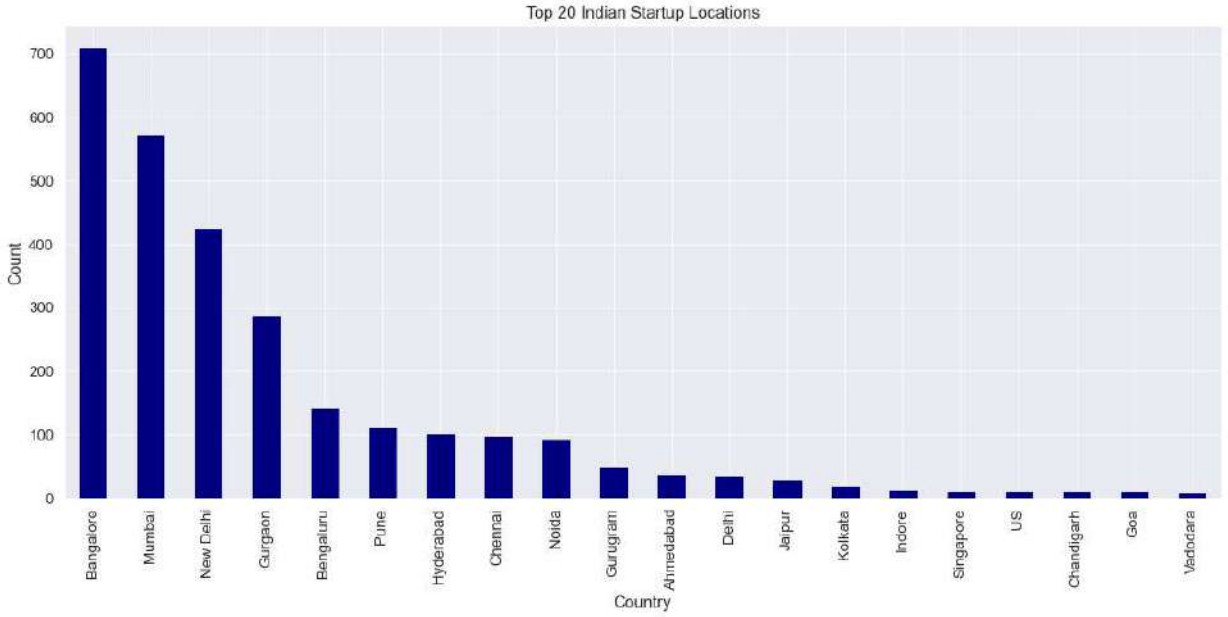
In [97]:
```
data['City  Location'].value_counts().head(20)
```

Out[97]:
```
Bangalore     700
Mumbai        567
New Delhi     421
Gurgaon       287
Bengaluru     141
Pune          105
Hyderabad      99
Chennai        97
Noida          92
Gurugram       50
Ahmedabad      38
Delhi          34
Jaipur         30
Kolkata        21
Indore         13
Chandigarh     11
Vadodara       10
Goa            10
Singapore       8
Coimbatore      5
Name: City  Location, dtype: int64
```

In [98]:
```
# Separate locations
# Drop nulls, split location and flatten 2D array to 1D array
city_locations = data['City  Location'].dropna() \
                                       .map(lambda location:
                                            [l.strip() for l in location.split('/')]) \
                                       .sum()
pd.Series(city_locations).value_counts()[:20]
```

Out[98]:
```
Bangalore     709
Mumbai        572
New Delhi     425
Gurgaon       288
Bengaluru     143
Pune          112
Hyderabad     102
Chennai        98
Noida          93
Gurugram       50
Ahmedabad      38
Delhi          34
Jaipur         30
Kolkata        21
Indore         13
Singapore      12
US             11
Chandigarh     11
Goa            11
Vadodara       10
dtype: int64
```

In [99]:
```
pd.Series(city_locations).value_counts()[:20].plot.bar(color='navy')
plt.title('Top 20 Indian Startup Locations')
plt.xlabel('Country')
plt.ylabel('Count')
plt.show()
```



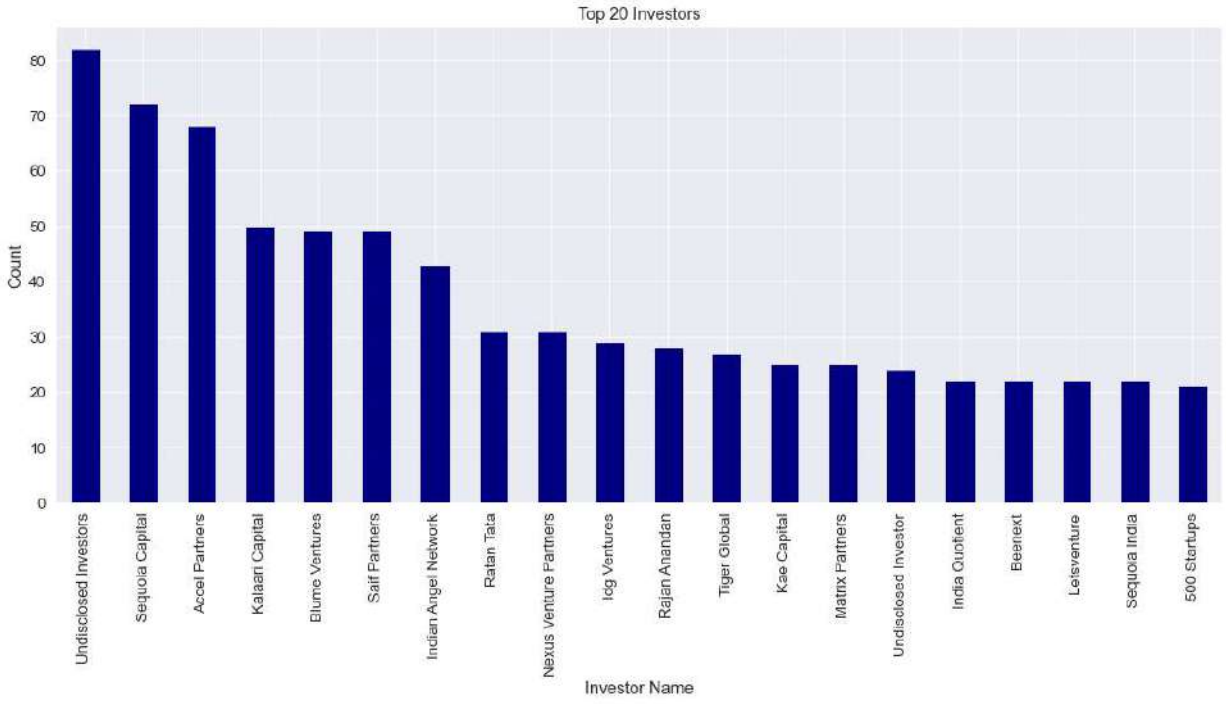
In [100]:
```
data['Investors Name'].unique()[:20]
```

Out[100]:
```
array(['Tiger Global Management', 'Susquehanna Growth Equity',
       'Sequoia Capital India', 'Vinod Khatumal',
       'Sprout Venture Partners', 'Chiratae Ventures', 'Ant Financial',
       'Sathguru Catalyzer Advisors', 'Ping An Global Voyager Fund',
       'Mumbai Angels, Ravikanth Reddy',
       'SAIF Partners, Spring Canter Investment Ltd.',
       'Paytm, NPTK, Sabre Partners and Neoplux', 'Vertex Growth Fund',
       nan, 'Ruizheng Investment',
       'Manipal Education and Medical Group (MEMG)',
       'SoftBank Vision Fund', 'Sequoia, CapitalG, Accel',
       'Sauce.vc, Rainforest Ventures',
       'Prime Venture Partners, LetsVenture, PS1 Venture and GlobalLogic co-fou
nder Rajul Garg'],
      dtype=object)
```

In [101]:
```
# Separate investor names
# Drop nulls, split location and flatten 2D array to 1D array
investors_names = data['Investors Name'].dropna() \
                                        .map(lambda investors:
                                             [i.strip().title() for i in investors.sp
                                        .sum()

# Remove empty names
pd.Series(investors_names) \
    .replace('', np.nan) \
    .dropna() \
    .value_counts()[:20]
```

Out[101]:
```
Undisclosed Investors     82
Sequoia Capital           72
Accel Partners            68
Kalaari Capital           50
Blume Ventures            49
Saif Partners             49
Indian Angel Network      43
Ratan Tata                31
Nexus Venture Partners    31
Idg Ventures              29
Rajan Anandan             28
Tiger Global              27
Kae Capital               25
Matrix Partners           25
Undisclosed Investor      24
India Quotient            22
Beenext                   22
Letsventure               22
Sequoia India             22
500 Startups              21
dtype: int64
```

In [102]:
```
# top 20 investors
pd.Series(investors_names) \
    .replace('', np.nan) \
    .dropna() \
    .value_counts()[:20] \
    .plot.bar(color='navy')

plt.title('Top 20 Investors')
plt.xlabel('Investor Name')
plt.ylabel('Count')
plt.show()
```



In [103]:
```
data['InvestmentnType'].unique()
```

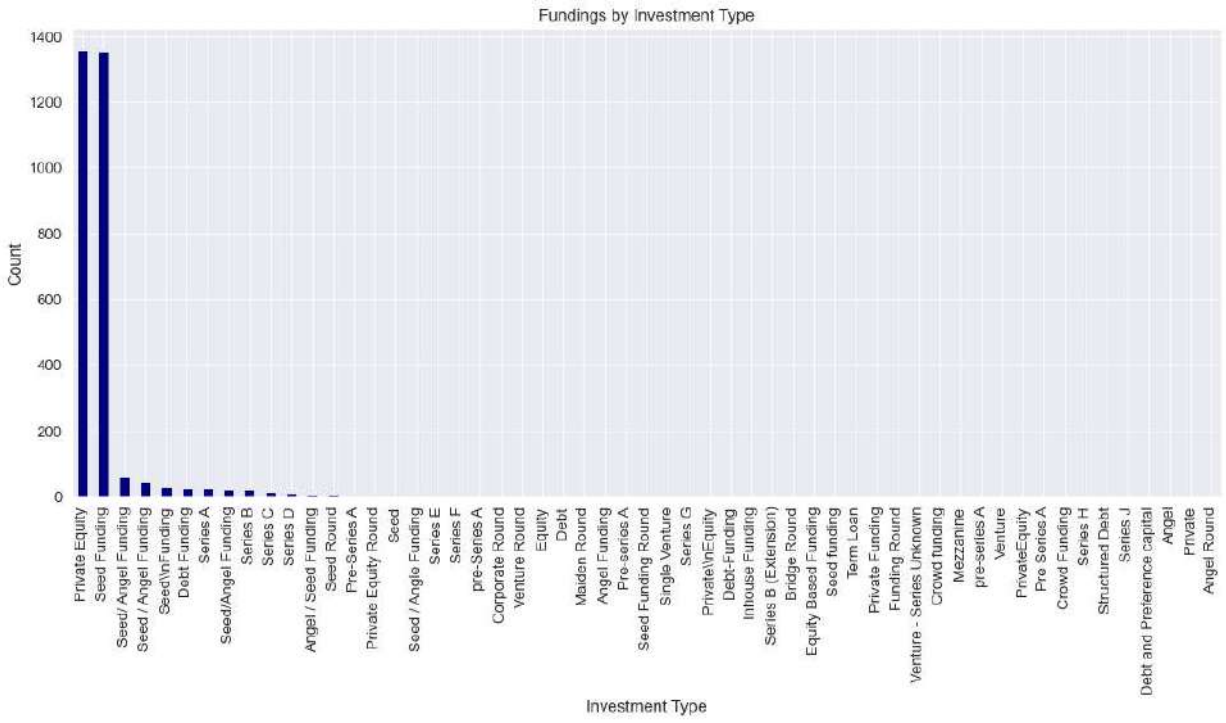
Out[103]:
```
array(['Private Equity Round', 'Series C', 'Series B', 'Pre-series A',
       'Seed Round', 'Series A', 'Series D', 'Seed', 'Series F',
       'Series E', 'Debt Funding', 'Series G', 'Series H', 'Venture',
       'Seed Funding', nan, 'Funding Round', 'Corporate Round',
       'Maiden Round', 'pre-series A', 'Seed Funding Round',
       'Single Venture', 'Venture Round', 'Pre-Series A', 'Angel',
       'Series J', 'Angel Round', 'pre-Series A',
       'Venture - Series Unknown', 'Bridge Round', 'Private Equity',
       'Debt and Preference capital', 'Inhouse Funding',
       'Seed/ Angel Funding', 'Debt', 'Pre Series A', 'Equity',
       'Debt-Funding', 'Mezzanine', 'Series B (Extension)',
       'Equity Based Funding', 'Private Funding', 'Seed / Angel Funding',
       'Seed/Angel Funding', 'Seed funding', 'Seed / Angle Funding',
       'Angel / Seed Funding', 'Private', 'Structured Debt', 'Term Loan',
       'PrivateEquity', 'Angel Funding', 'Seed\\\\nFunding',
       'Private\\\\nEquity', 'Crowd funding', 'Crowd Funding'],
      dtype=object)
```

In [104]:
```
data['InvestmentnType'].value_counts()
```

Out[104]:
```
Private Equity                1356
Seed Funding                  1355
Seed/ Angel Funding             60
Seed / Angel Funding            47
Seed\\nFunding                  30
Debt Funding                    25
Series A                        24
Seed/Angel Funding              23
Series B                        20
Series C                        14
Series D                        12
Angel / Seed Funding             8
Seed Round                       7
Pre-Series A                     4
Private Equity Round             4
Seed                             4
Seed / Angle Funding             3
Series E                         2
Series F                         2
pre-Series A                     2
Corporate Round                  2
Venture Round                    2
Equity                           2
Debt                             1
Maiden Round                     1
Angel Funding                    1
Pre-series A                     1
Seed Funding Round               1
Single Venture                   1
Series G                         1
Private\\nEquity                 1
Debt-Funding                     1
Inhouse Funding                  1
Series B (Extension)             1
Bridge Round                     1
Equity Based Funding             1
Seed funding                     1
Term Loan                        1
Private Funding                  1
Funding Round                    1
Venture - Series Unknown         1
Crowd funding                    1
Mezzanine                        1
pre-series A                     1
Venture                          1
PrivateEquity                    1
Pre Series A                     1
Crowd Funding                    1
Series H                         1
Structured Debt                  1
Series J                         1
Debt and Preference capital      1
Angel                            1
Private                          1
```

```
Angel Round                          1
Name: InvestmentnType, dtype: int64
```

In [105]:
```
data['InvestmentnType'].value_counts().plot.bar(color='navy')
plt.title('Fundings by Investment Type')
plt.xlabel('Investment Type')
plt.ylabel('Count')
plt.show()
```



In [106]:
```
data.isnull().sum()
```

Out[106]:
```
Sr No                    0
Date dd/mm/yyyy          0
Startup Name             0
Industry Vertical      171
SubVertical            936
City  Location         180
Investors Name          24
InvestmentnType          4
Amount in USD          960
Remarks               2625
dtype: int64
```

In [107]:
```
missing_values = data.isnull().sum()
missing_value_frame = missing_values.to_frame()

missing_value_frame.columns=['count']
missing_value_frame.index.names = ['Name']
missing_value_frame['Name'] = missing_value_frame.index
missing_value_frame.head(10)
```

Out[107]:

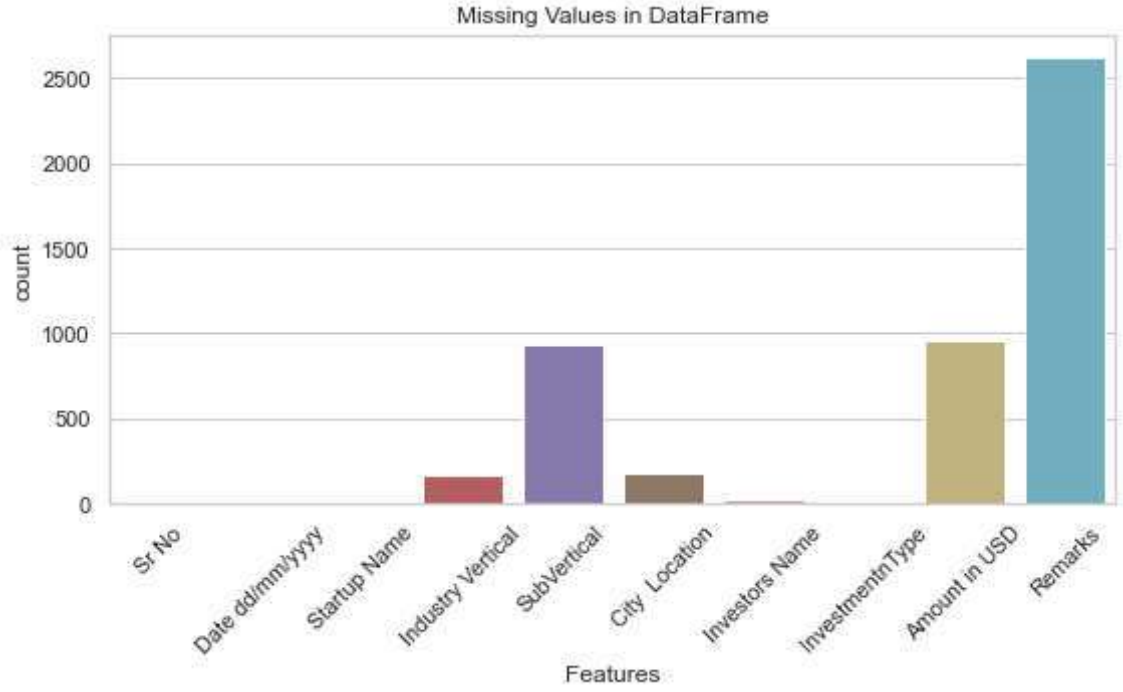
Name	count	Name
Sr No	0	Sr No
Date dd/mm/yyyy	0	Date dd/mm/yyyy
Startup Name	0	Startup Name
Industry Vertical	171	Industry Vertical
SubVertical	936	SubVertical
City Location	180	City Location
Investors Name	24	Investors Name
InvestmentnType	4	InvestmentnType
Amount in USD	960	Amount in USD
Remarks	2625	Remarks

In [108]:
```
newdata=data.copy()#backup cleansed data
del newdata['Remarks']#remaks is deleted to overcome stability in analysis
```

In [109]:
```
data.isnull().sum().sum()
```

Out[109]: 4900

In [110]:
```
plt.figure(figsize=(8,4))
sns.set(style="whitegrid", color_codes = True)
sns.barplot(x='Name', y='count', data=missing_value_frame);
plt.title("Missing Values in DataFrame")
plt.xlabel("Features")
plt.xticks(rotation=45);
```



In [111]:
```
# Converting datatype of values in 'AmountUSD' column from string to float. Marki
data.loc[data['Amount in USD'].isin(['undisclosed', 'unknown', 'Undisclosed']), '

data['Amount in USD'] = data['Amount in USD'].astype(str)
data['NewAmount in USD'] = data['Amount in USD'].apply(lambda x : re.sub("[^0-9]'
data.loc[data['NewAmount in USD']=='', 'NewAmount in USD'] = 0 #'nan' # replace w
data['NewAmount in USD'] = data['NewAmount in USD'].astype(float)
data.head()
```

Out[111]:

	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name
0	1	09/01/2020	BYJU'S	E-Tech	E-learning	Bengaluru	Tiger Global Management
1	2	13/01/2020	Shuttl	Transportation	App based shuttle service	Gurgaon	Susquehanna Growth Equity
2	3	09/01/2020	Mamaearth	E-commerce	Retailer of baby and toddler products	Bengaluru	Sequoia Capital India
3	4	02/01/2020	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatumal
4	5	02/01/2020	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprout Venture Partners

In [112]:
```
data['Remarks'].fillna('None', inplace=True)
data.head()
```

Out[112]:

	Sr No	Date dd/mm/yyyy	Startup Name	Industry Vertical	SubVertical	City Location	Investors Name
0	1	09/01/2020	BYJU'S	E-Tech	E-learning	Bengaluru	Tiger Global Management
1	2	13/01/2020	Shuttl	Transportation	App based shuttle service	Gurgaon	Susquehanna Growth Equity
2	3	09/01/2020	Mamaearth	E-commerce	Retailer of baby and toddler products	Bengaluru	Sequoia Capital India
3	4	02/01/2020	https://www.wealthbucket.in/	FinTech	Online Investment	New Delhi	Vinod Khatumal
4	5	02/01/2020	Fashor	Fashion and Apparel	Embroiled Clothes For Women	Mumbai	Sprout Venture Partners

In [113]:
```
data['City  Location'].unique()
```

Out[113]:
```
array(['Bengaluru', 'Gurgaon', 'New Delhi', 'Mumbai', 'Chennai', 'Pune',
       'Noida', 'Faridabad', 'San Francisco', 'San Jose,', 'Amritsar',
       'Delhi', 'Kormangala', 'Tulangan', 'Hyderabad', 'Burnsville',
       'Menlo Park', 'Gurugram', 'Palo Alto', 'Santa Monica', 'Singapore',
       'Taramani', 'Andheri', 'Chembur', 'Nairobi', 'Haryana', 'New York',
       'Karnataka', 'Mumbai/Bengaluru', 'Bhopal',
       'Bengaluru and Gurugram', 'India/Singapore', 'Jaipur', 'India/US',
       'Nagpur', 'Indore', 'New York, Bengaluru', 'California', 'India',
       'Ahemadabad', 'Rourkela', 'Srinagar', 'Bhubneswar', 'Chandigarh',
       'Delhi & Cambridge', 'Kolkatta', 'Kolkata', 'Coimbatore',
       'Bangalore', 'Udaipur', nan, 'Ahemdabad', 'Bhubaneswar',
       'Ahmedabad', 'Surat', 'Goa', 'Uttar Pradesh', 'Nw Delhi', 'Gaya',
       'Vadodara', 'Trivandrum', 'Missourie', 'Panaji', 'Gwalior',
       'Karur', 'Udupi', 'Kochi', 'Agra', 'Bangalore/ Bangkok', 'Hubli',
       'Kerala', 'Kozhikode', 'US', 'Siliguri', 'USA', 'Lucknow',
       'Kanpur', 'SFO / Bangalore', 'London', 'Seattle / Bangalore',
       'Pune/Seattle', 'Pune / Dubai', 'Bangalore / SFO', 'Varanasi',
       'New Delhi / US', 'Mumbai / UK', 'Jodhpur', 'Hyderabad/USA',
       'Boston', 'Bangalore / Palo Alto', 'Mumbai / NY', 'USA/India',
       'Goa/Hyderabad', 'Noida / Singapore', 'Belgaum', 'Pune / US',
       'Chennai/ Singapore', 'Pune / Singapore', 'Bangalore / San Mateo',
       'New York/ India', 'US/India', 'Gurgaon / SFO', 'Bangalore / USA',
       'New Delhi/ Houston', 'Mumbai / Global', 'India / US',
       '\\\\xc2\\\\xa0Noida', '\\\\xc2\\\\xa0Bangalore',
       '\\\\xc2\\\\xa0Gurgaon', '\\\\xc2\\\\xa0New Delhi',
       '\\\\xc2\\\\xa0Mumbai', 'New Delhi / California',
       'Dallas / Hyderabad'], dtype=object)
```

In [116]:
```
## creating new list to having startups with their total funding
maxtenstartup=[]
for startup in data['Startup Name'].unique():
    df=data[data['Startup Name']==startup]    ## get the dataframe for each start
    sum=np.sum(df['Amount in USD'])         ## sum total funding of startup
    maxtenstartup.append([startup,sum])

startup=pd.DataFrame(maxtenstartup,columns=['startup','Revenue" in million"'])
#converting the list to dataframe and sort them by the fundin amount
startup.sort_values(by='Revenue" in million"',ascending=False,inplace=True)
```

In [117]:
```
startup.head(10)
```

Out[117]:

	startup	Revenue" in million"
291	RailYatri	nannannannan
217	Zoctr	nannannan10,00,000
990	BookEventz	nannannan
340	Wydr	nannannan
1459	Faircent	nannannan
281	Lenskart	nannan6,00,00,000
1066	WorkIndia	nannan5,00,000
548	Sattviko	nannan3,20,000
148	Mad Street Den	nannan15,00,000
1686	Tracxn	nannan1,00,00,00035,00,000

In [118]:
```
data.isnull().sum().sum()
```

Out[118]: 1315

In [119]:
```
# in which sector there are most startups
d=data[data['Industry Vertical']!='others']['Industry Vertical'].value_counts().h
explode = (0.1, 0, 0, 0,0,0)
fig1,ax1=plt.subplots(figsize=(20,10))

ax1.pie(d.values,explode=explode, labels=d.index,autopct='%1.1f%%', shadow=True,
ax1.axis('equal')
plt.title("Famous industries of startup",fontsize=30)
plt.show()
```