EMPLOYEE PROMOTION CASE STUDY

Importing module package

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

Accessing .csv file raw data using pandas

In [2]:
```
emp_data = pd.read_csv("employee_promotion.csv")
```

Data Inspection

In [3]:
```
emp_data.describe()
```

Out[3]:

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service
count	54808.000000	54808.000000	54808.000000	50684.000000	54808.000000
mean	39195.830627	1.253011	34.803915	3.329256	5.865512
std	22586.581449	0.609264	7.660169	1.259993	4.265094
min	1.000000	1.000000	20.000000	1.000000	1.000000
25%	19669.750000	1.000000	29.000000	3.000000	3.000000
50%	39225.500000	1.000000	33.000000	3.000000	5.000000
75%	58730.500000	1.000000	39.000000	4.000000	7.000000
max	78298.000000	10.000000	60.000000	5.000000	37.000000

In [4]:
```
emp_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    52248 non-null  float64
 12  is_promoted           54808 non-null  int64  
dtypes: float64(2), int64(6), object(5)
memory usage: 5.4+ MB
```

In [5]:
```
emp_data.columns
```

Out[5]:
```
Index(['employee_id', 'department', 'region', 'education', 'gende
r',
       'recruitment_channel', 'no_of_trainings', 'age', 'previous_
year_rating',
       'length_of_service', 'awards_won', 'avg_training_score', 'i
s_promoted'],
      dtype='object')
```

In [6]:
```
emp_data.dtypes
```

Out[6]:
```
employee_id               int64
department               object
region                   object
education                object
gender                   object
recruitment_channel      object
no_of_trainings           int64
age                       int64
previous_year_rating    float64
length_of_service         int64
awards_won                int64
avg_training_score      float64
is_promoted               int64
dtype: object
```

Data Cleaning

In [7]:
```
# Checkking Null Values

emp_data.isnull().sum()*100/emp_data.shape[0]
```

Out[7]:
```
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      4.670851
is_promoted             0.000000
dtype: float64
```

In [8]:
```
emp_data.education.describe()
```

Out[8]:
```
count          52399
unique             3
top       Bachelor's
freq           36669
Name: education, dtype: object
```

In [9]:
```
emp_data.education.isna().value_counts()
```

Out[9]:
```
False    52399
True      2409
Name: education, dtype: int64
```

In [10]:
```
emp_data.previous_year_rating.describe()
```

Out[10]:
```
count    50684.000000
mean         3.329256
std          1.259993
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          5.000000
Name: previous_year_rating, dtype: float64
```

In [11]:
```
emp_data.previous_year_rating.isna().value_counts()
```

Out[11]:
```
False    50684
True      4124
Name: previous_year_rating, dtype: int64
```

In [12]:
```
emp_data.isnull().info()
print("\n\nDataset is absolutely affirmative\n")
```

```
<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  bool
 1   department            54808 non-null  bool
 2   region                54808 non-null  bool
 3   education             54808 non-null  bool
 4   gender                54808 non-null  bool
 5   recruitment_channel   54808 non-null  bool
 6   no_of_trainings       54808 non-null  bool
 7   age                   54808 non-null  bool
 8   previous_year_rating  54808 non-null  bool
 9   length_of_service     54808 non-null  bool
 10  awards_won            54808 non-null  bool
 11  avg_training_score    54808 non-null  bool
 12  is_promoted           54808 non-null  bool
dtypes: bool(13)
memory usage: 695.9 KB


Dataset is absolutely affirmative
```

In [13]:
```
emp_non_test_data = emp_data
emp_non_test_data
```

Out[13]:

	employee_id	department	region	education	gender	recruitment_channel	no_of_
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	
1	65141	Operations	region_22	Bachelor's	m	other	
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	
4	48945	Technology	region_26	Bachelor's	m	other	
...	...	...	...	...	...	...	
54803	3030	Technology	region_14	Bachelor's	m	sourcing	
54804	74592	Operations	region_27	Master's & above	f	other	
54805	13918	Analytics	region_1	Bachelor's	m	other	
54806	13614	Sales & Marketing	region_9	NaN	m	sourcing	
54807	51526	HR	region_22	Bachelor's	m	other	

54808 rows × 13 columns

In [14]:
```
emp_data.drop(['employee_id'],axis=1,inplace=True)
```

In [15]:
```
emp_data.describe().style.background_gradient(cmap = "summer")
```

Out[15]:

	no_of_trainings	age	previous_year_rating	length_of_service	awards_won
count	54808.000000	54808.000000	50684.000000	54808.000000	54808.000000
mean	1.253011	34.803915	3.329256	5.865512	0.023172
std	0.609264	7.660169	1.259993	4.265094	0.150450
min	1.000000	20.000000	1.000000	1.000000	0.000000
25%	1.000000	29.000000	3.000000	3.000000	0.000000
50%	1.000000	33.000000	3.000000	5.000000	0.000000
75%	1.000000	39.000000	4.000000	7.000000	0.000000
max	10.000000	60.000000	5.000000	37.000000	1.000000

Exploratory data analysis (EDA) part

In [16]:
```
emp_data
```

Out[16]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35
1	Operations	region_22	Bachelor's	m	other	1	30
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34
3	Sales & Marketing	region_23	Bachelor's	m	other	2	39
4	Technology	region_26	Bachelor's	m	other	1	45
...	...	...	...	...	...	...	...
54803	Technology	region_14	Bachelor's	m	sourcing	1	48
54804	Operations	region_27	Master's & above	f	other	1	37
54805	Analytics	region_1	Bachelor's	m	other	1	27
54806	Sales & Marketing	region_9	NaN	m	sourcing	1	29
54807	HR	region_22	Bachelor's	m	other	1	27

54808 rows × 12 columns

In [17]:
```
emp_data.is_promoted.value_counts()
```

Out[17]:
```
0    50140
1     4668
Name: is_promoted, dtype: int64
```

In [18]:
```
emp_data["is_promoted"]
```

Out[18]:
```
0        0
1        0
2        0
3        0
4        0
        ..
54803    0
54804    0
54805    0
54806    0
54807    0
Name: is_promoted, Length: 54808, dtype: int64
```

In [19]:
```
emp_data.head()
```

Out[19]:

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

In [20]:
```
emp_data.tail()
```

Out[20]:

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

In [21]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

In [22]:
```
emp_data.describe()
```

Out[22]:

	no_of_trainings	age	previous_year_rating	length_of_service	awards_won
count	54808.000000	54808.000000	50684.000000	54808.000000	54808.000000
mean	1.253011	34.803915	3.329256	5.865512	0.023172
std	0.609264	7.660169	1.259993	4.265094	0.150450
min	1.000000	20.000000	1.000000	1.000000	0.000000
25%	1.000000	29.000000	3.000000	3.000000	0.000000
50%	1.000000	33.000000	3.000000	5.000000	0.000000
75%	1.000000	39.000000	4.000000	7.000000	0.000000
max	10.000000	60.000000	5.000000	37.000000	1.000000

In [23]:
```
emp_data.corr()
```

```
/tmp/ipykernel_123452/1395959586.py:1: FutureWarning: The default
value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or sp
ecify the value of numeric_only to silence this warning.
  emp_data.corr()
```

Out[23]:

	no_of_trainings	age	previous_year_rating	length_of_service	av
no_of_trainings	1.000000	-0.081278	-0.063126	-0.057275	
age	-0.081278	1.000000	0.006008	0.657111	
previous_year_rating	-0.063126	0.006008	1.000000	0.000253	
length_of_service	-0.057275	0.657111	0.000253	1.000000	
awards_won	-0.007628	-0.008169	0.027738	-0.039927	
avg_training_score	0.044430	-0.049500	0.075474	-0.039381	
is_promoted	-0.024896	-0.017166	0.159320	-0.010670	

In [24]:
```
emp_data.corr().value_counts()
```

```
/tmp/ipykernel_123452/2348875662.py:1: FutureWarning: The default
value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or sp
ecify the value of numeric_only to silence this warning.
  emp_data.corr().value_counts()
```

Out[24]:
```
no_of_trainings  age        previous_year_rating  length_of_servic
e  awards_won  avg_training_score  is_promoted
-0.081278         1.000000   0.006008              0.657111
-0.008169   -0.049500           -0.017166      1
-0.063126         0.006008   1.000000              0.000253
0.027738    0.075474            0.159320      1
-0.057275         0.657111   0.000253              1.000000
-0.039927   -0.039381           -0.010670      1
-0.024896        -0.017166   0.159320             -0.010670
0.195871    0.184386            1.000000      1
-0.007628        -0.008169   0.027738             -0.039927
1.000000    0.073963            0.195871      1
 0.044430        -0.049500   0.075474             -0.039381
0.073963    1.000000            0.184386      1
 1.000000        -0.081278  -0.063126             -0.057275
-0.007628    0.044430           -0.024896      1
dtype: int64
```

In [25]:
```
emp_data['department']
```

Out[25]:
```
0        Sales & Marketing
1               Operations
2        Sales & Marketing
3        Sales & Marketing
4               Technology
                   ...
54803           Technology
54804           Operations
54805            Analytics
54806    Sales & Marketing
54807                   HR
Name: department, Length: 54808, dtype: object
```

In [26]:
```
emp_data['department'].unique()
```

Out[26]:
```
array(['Sales & Marketing', 'Operations', 'Technology', 'Analytic
s',
       'R&D', 'Procurement', 'Finance', 'HR', 'Legal'], dtype=obje
ct)
```

In [27]:
```
emp_data['department'].count()
```

Out[27]:
```
54808
```

In [28]:
```
emp_data
```

Out[28]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35
1	Operations	region_22	Bachelor's	m	other	1	30
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34
3	Sales & Marketing	region_23	Bachelor's	m	other	2	39
4	Technology	region_26	Bachelor's	m	other	1	45
...	...	...	...	...	...	...	...
54803	Technology	region_14	Bachelor's	m	sourcing	1	48
54804	Operations	region_27	Master's & above	f	other	1	37
54805	Analytics	region_1	Bachelor's	m	other	1	27
54806	Sales & Marketing	region_9	NaN	m	sourcing	1	29
54807	HR	region_22	Bachelor's	m	other	1	27

54808 rows × 12 columns

In [29]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

In [30]:
```
emp_data['awards_won'].unique()
```

Out[30]:
```
array([0, 1])
```

In [31]:
```
#Crosstab between education and gender

cross_tab_data_edu_gen = pd.crosstab(emp_data["education"],emp_data
cross_tab_data_edu_gen
```

Out[31]:

gender education	f	m
Bachelor's	10854	25815
Below Secondary	289	516
Master's & above	4778	10147

In [32]:
```
#Crosstab between department and region

cross_tab_data_dept_reg = pd.crosstab(emp_data["department"],emp_da
cross_tab_data_dept_reg
```

Out[32]:

region department	region_1	region_10	region_11	region_12	region_13	region_14	region_15
Analytics	76	26	128	37	133	48	234
Finance	7	13	57	10	87	23	76
HR	7	15	41	9	83	16	67
Legal	1	0	22	12	32	0	18
Operations	61	87	281	153	540	192	965
Procurement	18	152	174	62	425	118	360
R&D	0	2	47	2	26	2	31
Sales & Marketing	367	235	414	178	923	321	838
Technology	73	118	151	37	399	107	219

9 rows × 34 columns

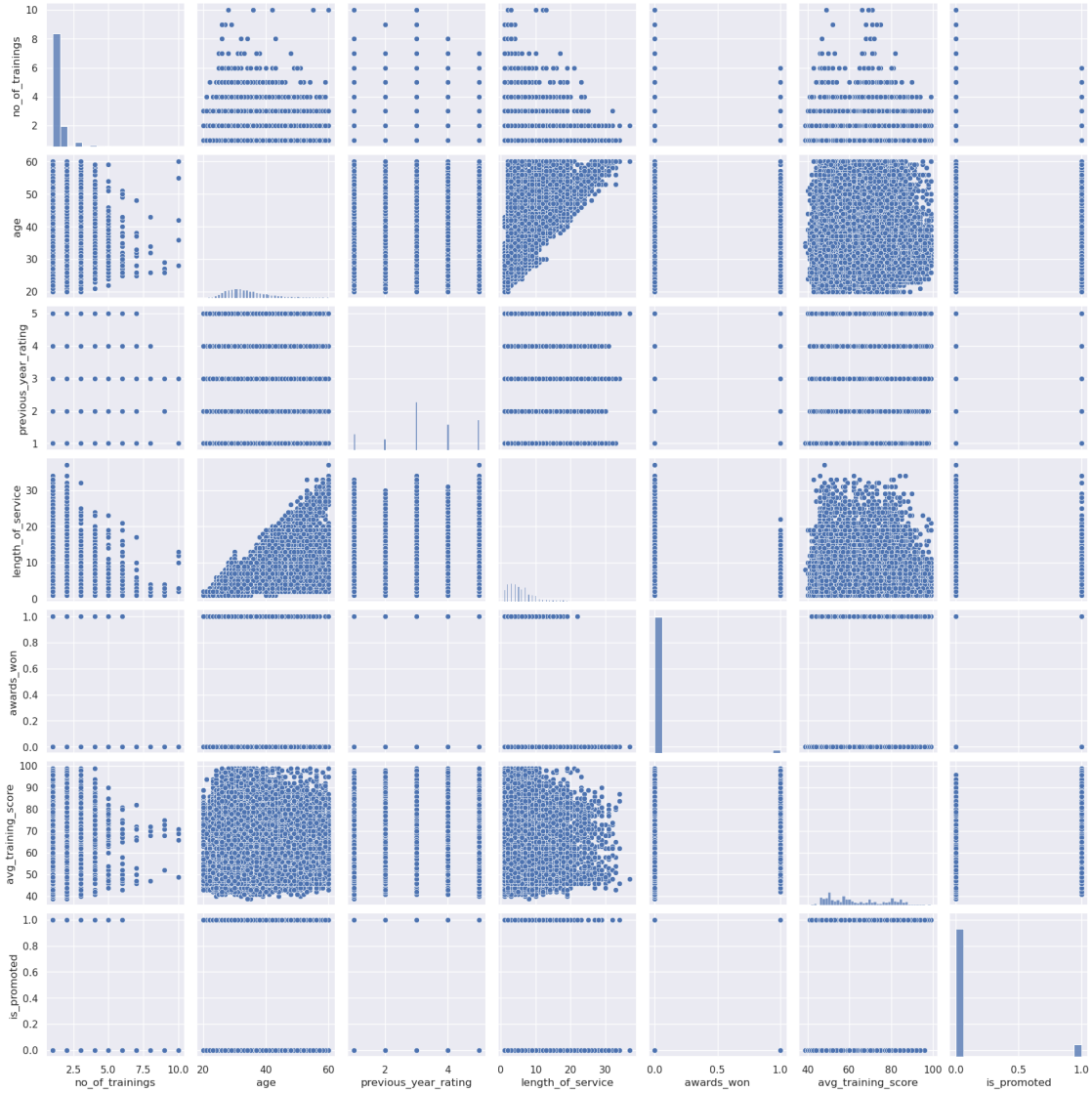
Data visualization using Seaborn and MatplotLib.Pyplot

Setting seaborn environment

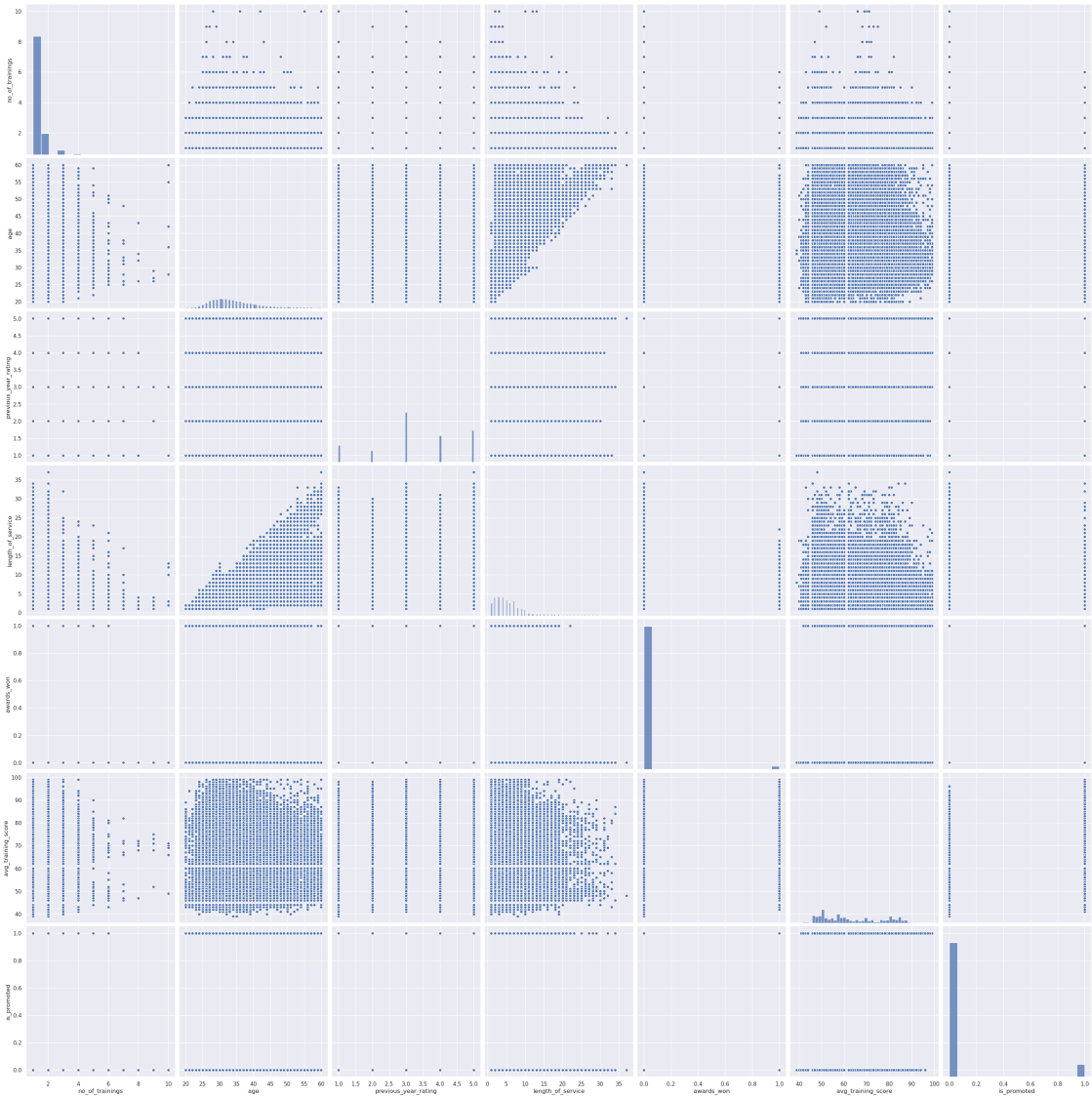
In [33]:
```
sns.set()
```

Pairplot of Employee Promotion Data (Bi-variate analysis)

In [34]:
```
# Normal Pairplot of Employee Promotion Data

sns.pairplot(emp_data)
plt.show()
```



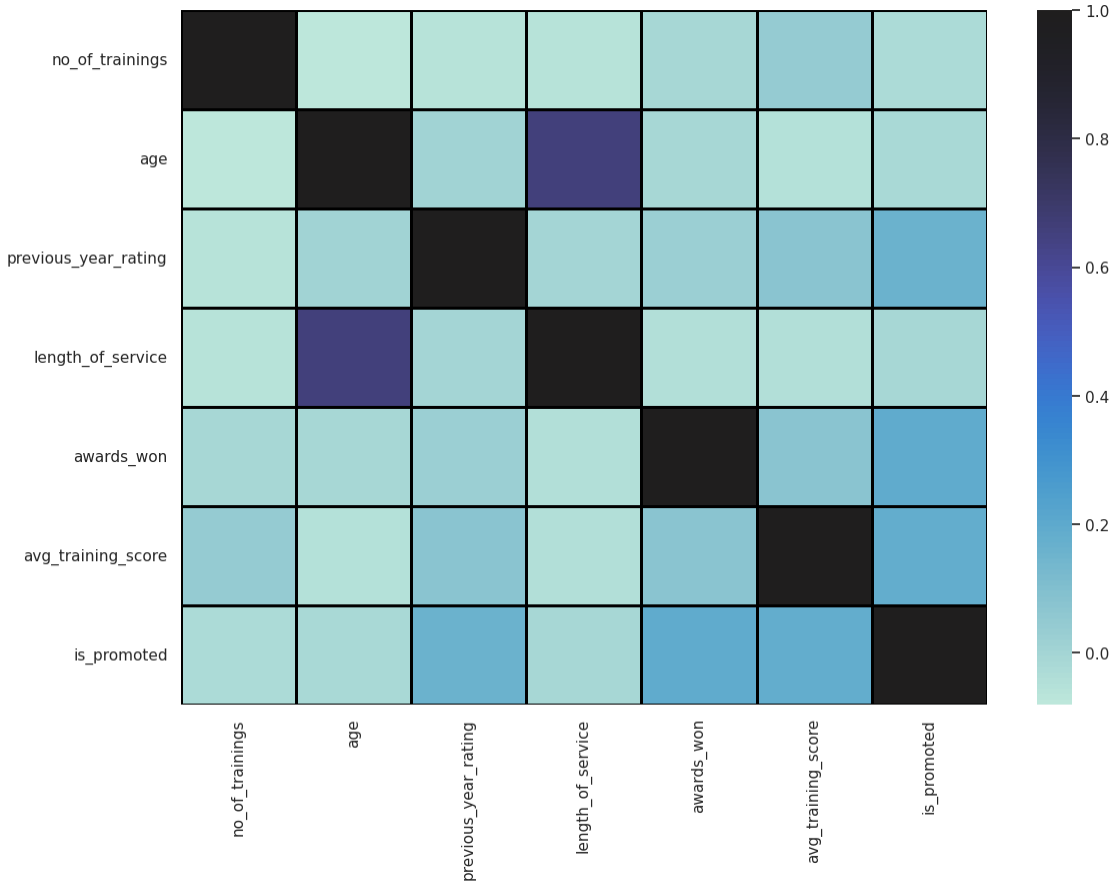
In [35]:
```
#Pairplot of Employee Promotion w.r.t "Sex,Day"

sns.pairplot(emp_data,height=4.5,hue_order=["sex","day"],diag_kind=
plt.show()
```



Heatmap of Employee Promotion Data

In [36]:
```
plt.figure(figsize=(13,9))
sns.set()
sns.heatmap(emp_data.corr(),center=True,linewidths=2,linecolor='bla
plt.show()
```

```
/tmp/ipykernel_123452/2446887256.py:3: FutureWarning: The default
value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or sp
ecify the value of numeric_only to silence this warning.
  sns.heatmap(emp_data.corr(),center=True,linewidths=2,linecolor
='black')
```



Jointplot between region and department

In [42]:
```
plt.figure(figsize=(19,6))
sns.jointplot(data=emp_data,x="region",y="department")
plt.show()
```

```
<Figure size 1900x600 with 0 Axes>
```

In [43]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

In [44]:
```
emp_data
```

Out[44]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35
1	Operations	region_22	Bachelor's	m	other	1	30
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34
3	Sales & Marketing	region_23	Bachelor's	m	other	2	39
4	Technology	region_26	Bachelor's	m	other	1	45
...	...	...	...	...	...	...	...
54803	Technology	region_14	Bachelor's	m	sourcing	1	48
54804	Operations	region_27	Master's & above	f	other	1	37
54805	Analytics	region_1	Bachelor's	m	other	1	27
54806	Sales & Marketing	region_9	NaN	m	sourcing	1	29
54807	HR	region_22	Bachelor's	m	other	1	27

54808 rows × 12 columns

Adding Column "is_promoted"

In [45]:
```
import numpy.random as rd
```

In [46]:
```
max_set=[]
for i in range(0,54808):
    if(i%4==0):
        max_set.append(1)
    else:
        max_set.append(0)
```

EDA on appended dataset

In [47]:
```
emp_data["is_promoted"] = max_set
```

In [48]:
```
emp_data
```

Out[48]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
0	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35
1	Operations	region_22	Bachelor's	m	other	1	30
2	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34
3	Sales & Marketing	region_23	Bachelor's	m	other	2	39
4	Technology	region_26	Bachelor's	m	other	1	45
...	...	...	...	...	...	...	...
54803	Technology	region_14	Bachelor's	m	sourcing	1	48
54804	Operations	region_27	Master's & above	f	other	1	37
54805	Analytics	region_1	Bachelor's	m	other	1	27
54806	Sales & Marketing	region_9	NaN	m	sourcing	1	29
54807	HR	region_22	Bachelor's	m	other	1	27

54808 rows × 12 columns

In [49]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

In [50]:
```
emp_data["is_promoted"].unique()
```

Out[50]:
```
array([1, 0])
```

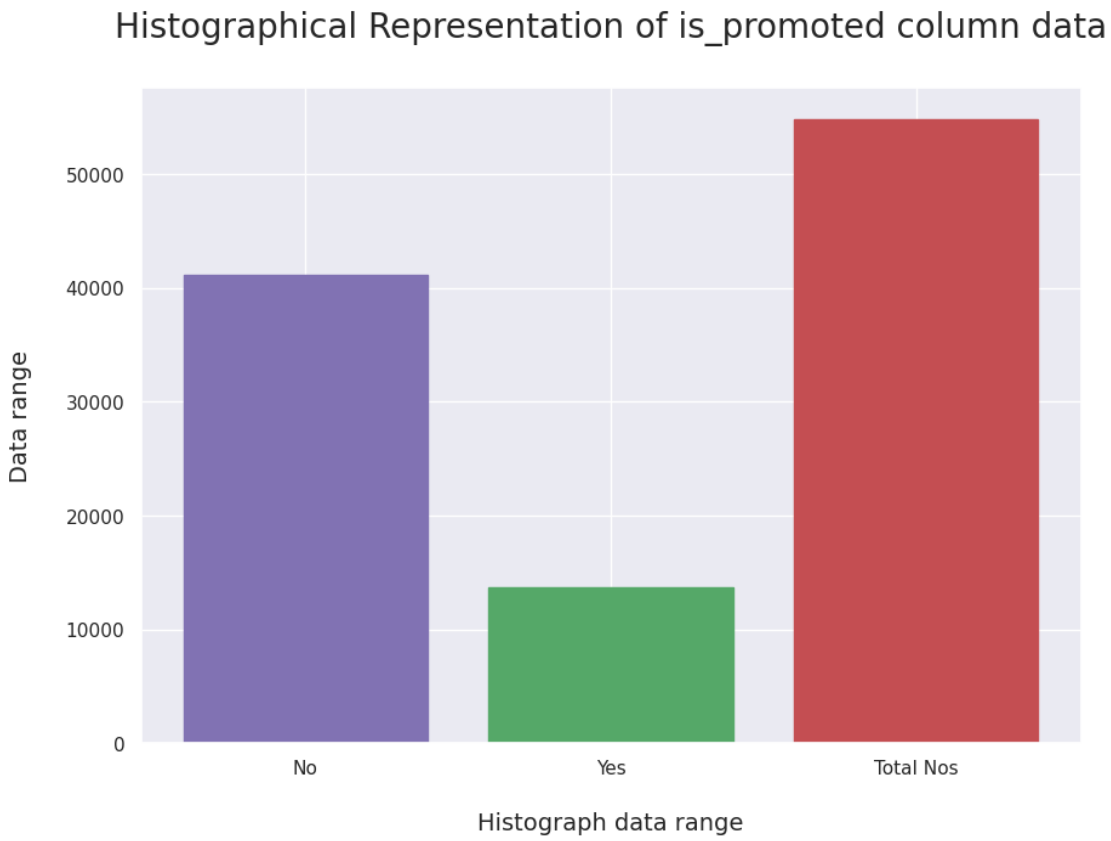
In [51]:
```
emp_data["is_promoted"].value_counts()
```

Out[51]:
```
0    41106
1    13702
Name: is_promoted, dtype: int64
```

Visualization of "is_promoted"

In [52]:
```
is_promoted_no = 41106
is_promoted_yes = 13702
is_promoted_tot = is_promoted_no+is_promoted_yes
```

In [53]:
```
plt.figure(figsize=(10,7))
is_promoted_bar = plt.bar(["No","Yes","Total Nos"],[is_promoted_no,
is_promoted_bar[0].set_color("m")
is_promoted_bar[1].set_color("g")
is_promoted_bar[2].set_color("r")
plt.xlabel("\nHistograph data range\n",fontsize=14)
plt.ylabel("Data range\n",fontsize=14)
plt.title("\nHistographical Representation of is_promoted column da
plt.show()
```

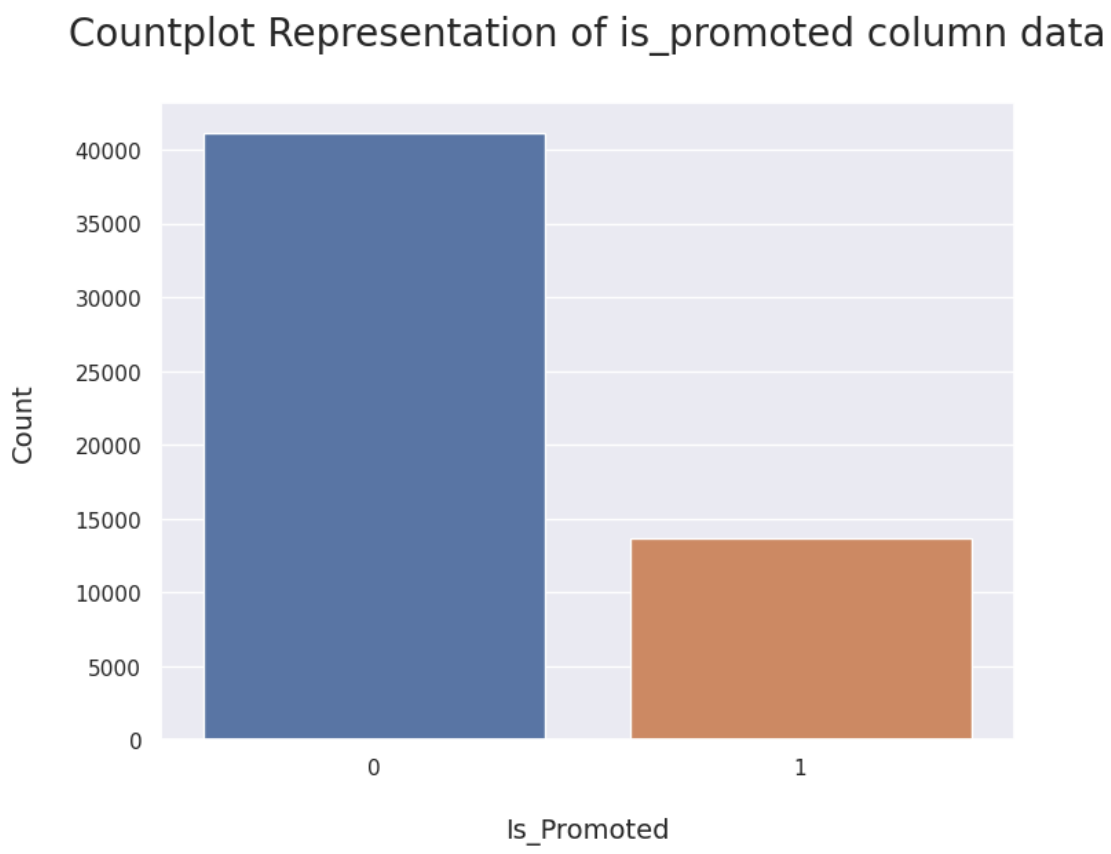


In [54]:
```
emp_data.shape
print(f"\nShape of emp_data = 54808 (rows) * 13 (cols)\n")
```

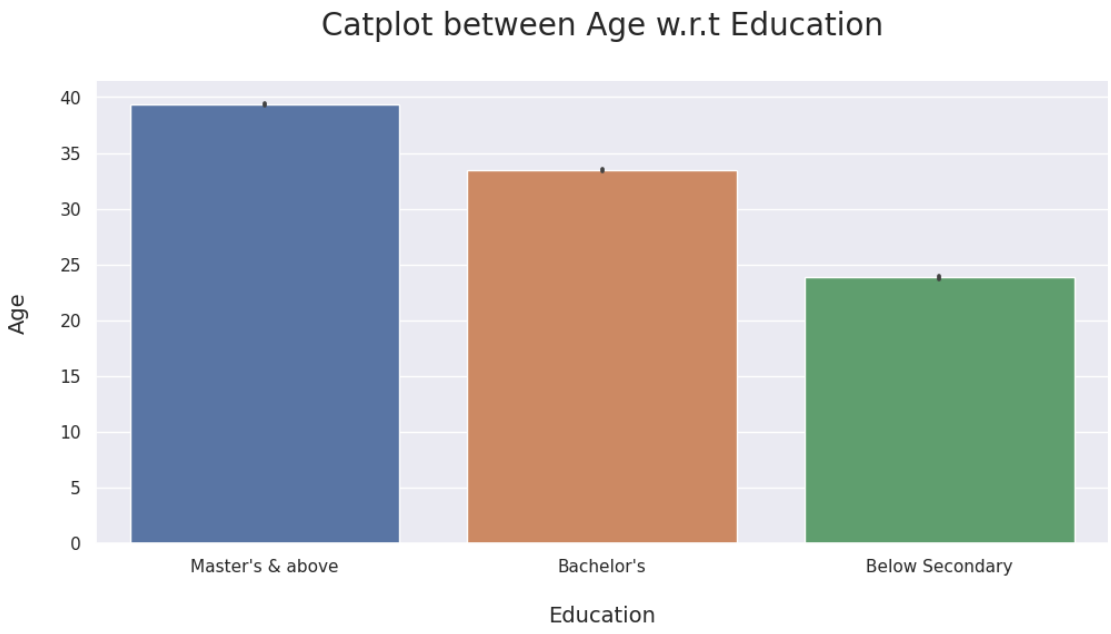
```
Shape of emp_data = 54808 (rows) * 13 (cols)
```

Countplot of is_promoted

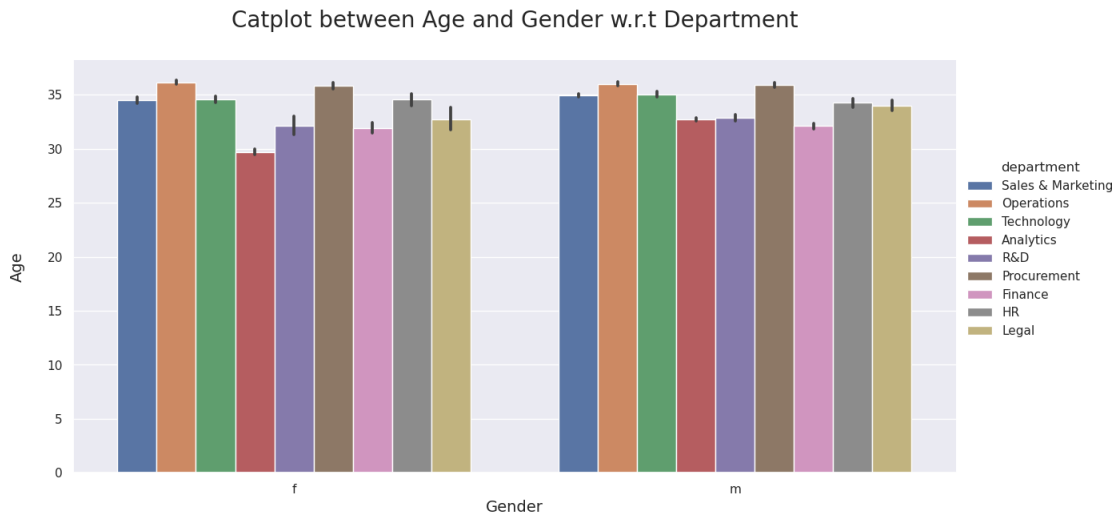
In [55]:
```
plt.figure(figsize=(8,6))
sns.countplot(x=emp_data.is_promoted)
plt.title("\nCountplot Representation of is_promoted column data\n"
plt.xlabel("\nIs_Promoted",fontsize=14)
plt.ylabel("Count\n",fontsize=14)
plt.show()
```



In [56]:
```
sns.catplot(x="education", y="age", kind="bar", data=emp_data, aspe
plt.title("\nCatplot between Age w.r.t Education\n",fontsize=20)
plt.xlabel("\nEducation",fontsize=14)
plt.ylabel("Age\n",fontsize=14)
plt.show()
```



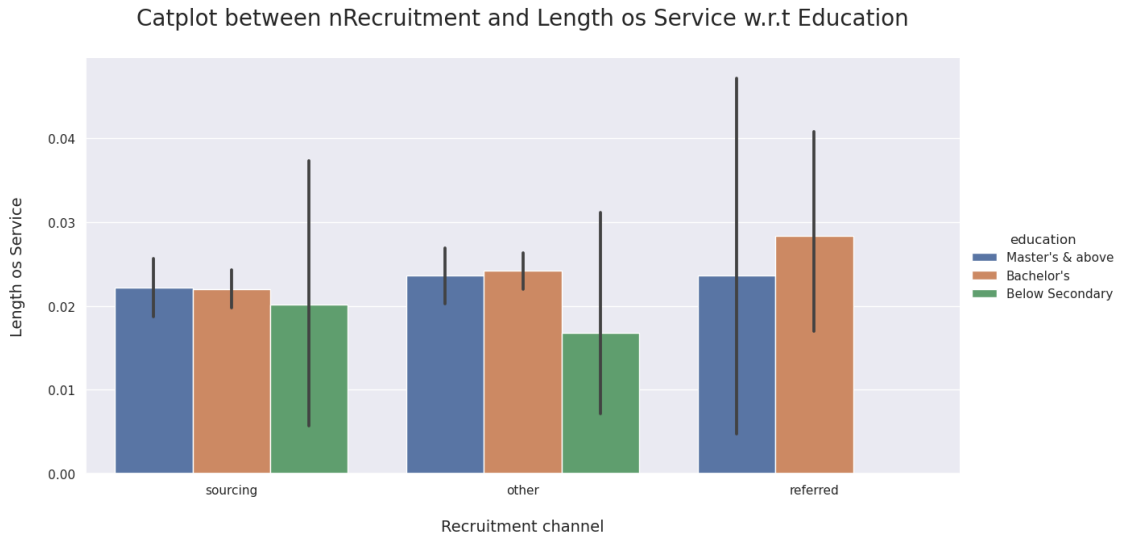
In [57]:
```
sns.catplot(x="gender", y="age", hue="department", kind="bar", data
plt.xlabel("Gender",fontsize=14)
plt.ylabel('Age\n',fontsize=14)
plt.title("\nCatplot between Age and Gender w.r.t Department\n",fon
plt.show()
```



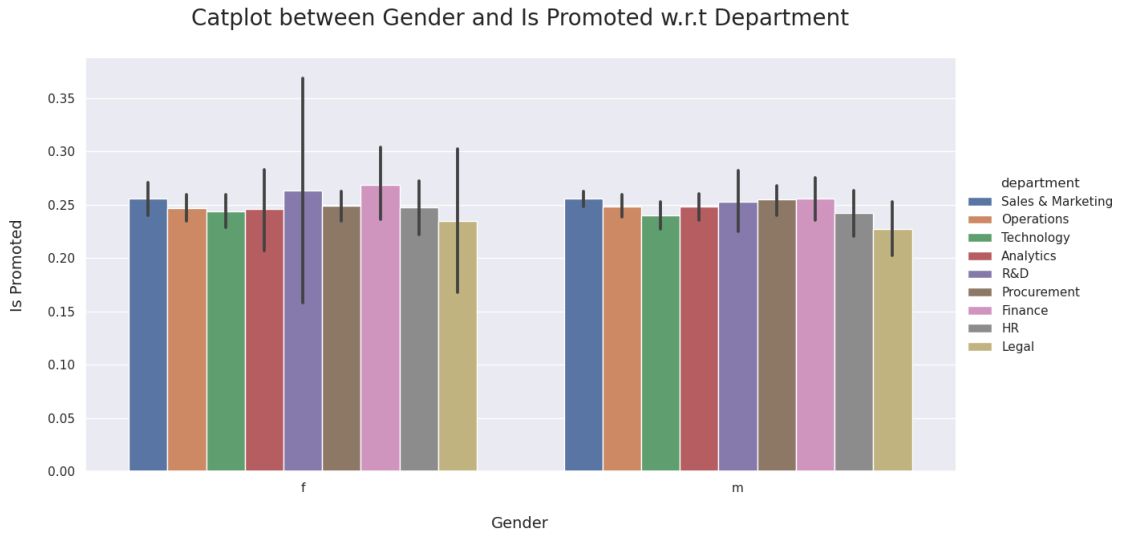
In [58]:
```
min(emp_data.age)
```

Out[58]: 20

In [59]:
```
sns.catplot(x="recruitment_channel", y="awards_won", hue="education
plt.xlabel("\nRecruitment channel\n",fontsize=14)
plt.ylabel('Length os Service\n',fontsize=14)
plt.title("\nCatplot between nRecruitment and Length os Service w.r
plt.show()
```

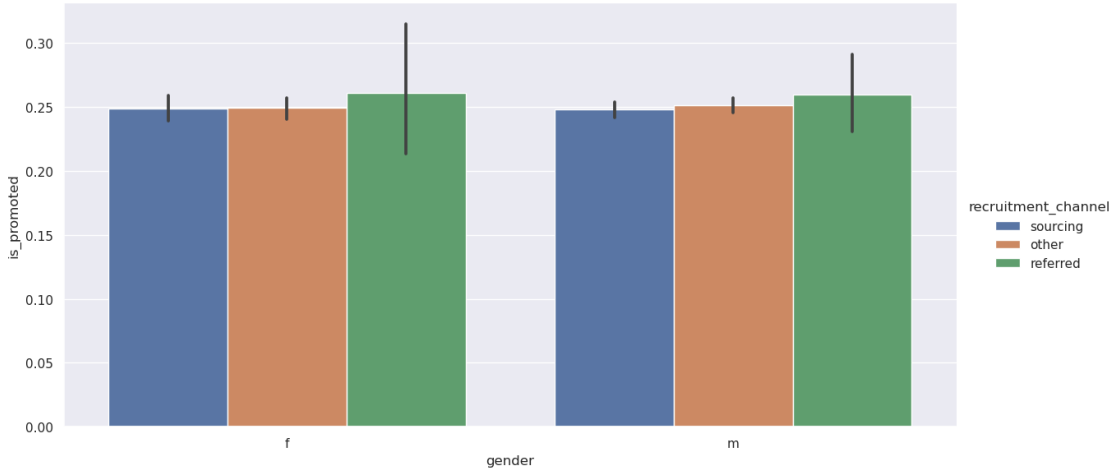


In [60]:
```
sns.catplot(x="gender", y="is_promoted", hue="department", kind="ba
plt.xlabel("\nGender\n",fontsize=14)
plt.ylabel('Is Promoted\n',fontsize=14)
plt.title("\nCatplot between Gender and Is Promoted w.r.t Departmen
plt.show()
```



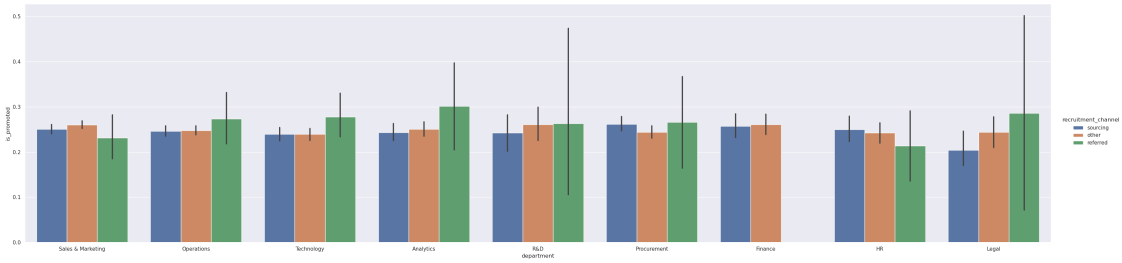
In [61]:
```
sns.catplot(x="gender", y="is_promoted", hue="recruitment_channel",
```

Out[61]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa02386e470>
```



In [62]:
```
sns.catplot(x="department", y="is_promoted", hue="recruitment_chann
```

Out[62]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa023050550>
```



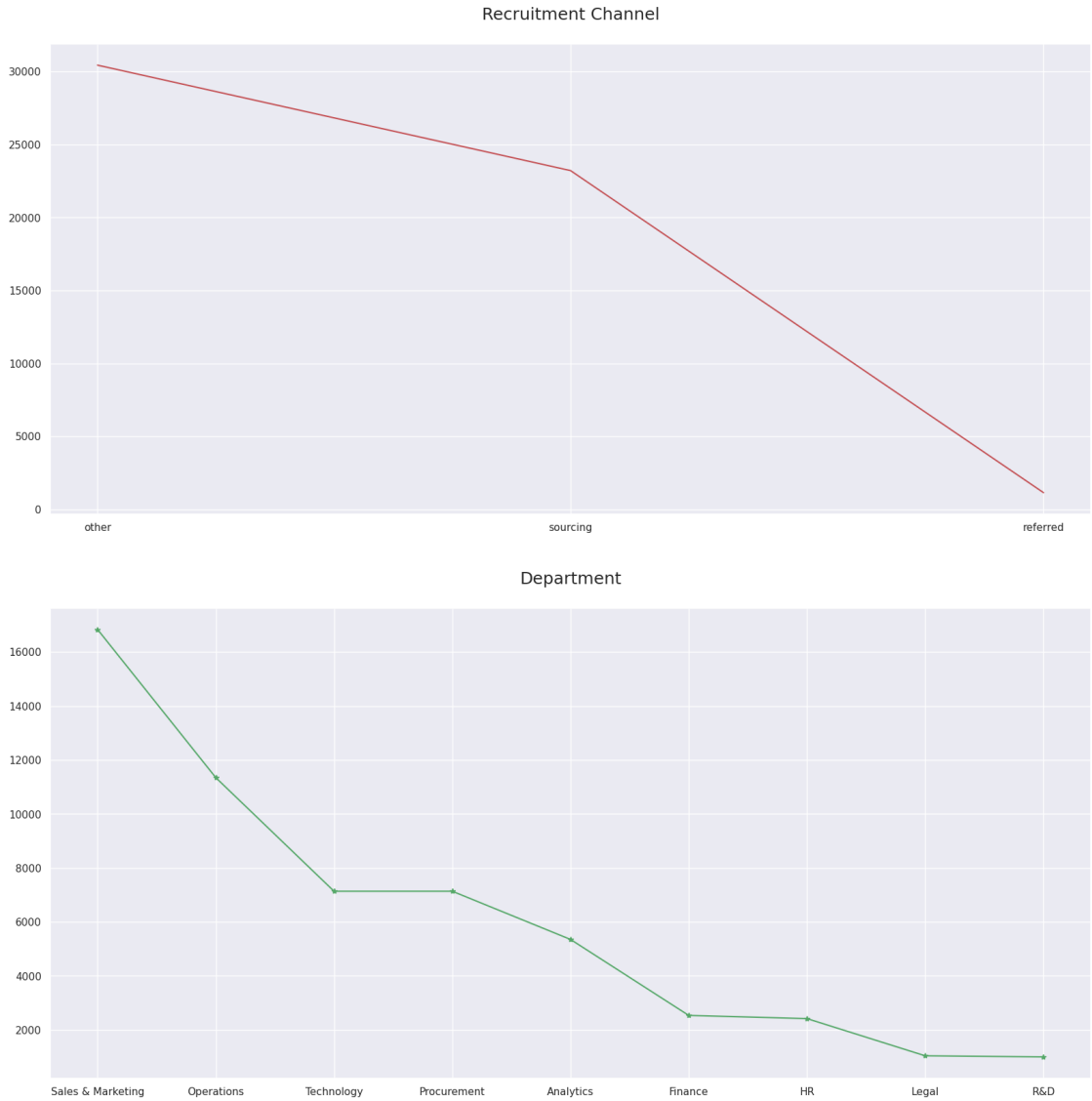
In [63]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

Line & Scatter Representation of Data (Uni-variate analysis)

In [64]:

```
fig = plt.figure(figsize=(20,20))
a1 = fig.add_subplot(2,1,1)
a2 = fig.add_subplot(2,1,2)
# ploting line & scatter chart

a1.plot(emp_data.recruitment_channel.value_counts(),color="r")
a1.set_title('\nRecruitment Channel\n',fontsize=18)
a2.plot(emp_data.department.value_counts(),color="g",marker="*")
a2.set_title('\n\n\n\nDepartment\n',fontsize=18)
#a3.plot(emp_data.avg_training_score.unique(),color="m",marker="*")
#a3.set_title('\n\n\n\nAverage Training Score\n',fontsize=18)
#a4.plot(emp_data.previous_year_rating.value_counts(),color="b",mar
#a4.set_title('\n\n\n\nPrevious Year Rating\n',fontsize=18)
#
plt.show()
```



Pie chart representation

In [65]:
```
emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   department            54808 non-null  object
 1   region                54808 non-null  object
 2   education             52399 non-null  object
 3   gender                54808 non-null  object
 4   recruitment_channel   54808 non-null  object
 5   no_of_trainings       54808 non-null  int64
 6   age                   54808 non-null  int64
 7   previous_year_rating  50684 non-null  float64
 8   length_of_service     54808 non-null  int64
 9   awards_won            54808 non-null  int64
 10  avg_training_score    52248 non-null  float64
 11  is_promoted           54808 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 5.0+ MB
```

In [66]:
```
emp_data.education.value_counts()
```

Out[66]:
```
Bachelor's          36669
Master's & above    14925
Below Secondary       805
Name: education, dtype: int64
```

In [67]:
```
ed_data = np.array([36669,14925,805])
ed_names = ['Bachelors','Masters & above','Below Secondary']
```

In [68]:
```
emp_data.department.value_counts()
```

Out[68]:
```
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 [69]:
```
dep_data = np.array([16840,11348,7138,7138,5352,2536,2418,1039,999]
dep_names = ['Sales & Marketing','Operations','Technology','Procure
```

In [70]:
```
emp_data.gender.value_counts()
```

Out[70]:
```
m    38496
f    16312
Name: gender, dtype: int64
```

In [71]:
```
gen_data = np.array([38496,16312])
gen_names = ['Male','Female']
```

In [72]:
```
emp_data.recruitment_channel.value_counts()
```

Out[72]:
```
other       30446
sourcing    23220
referred     1142
Name: recruitment_channel, dtype: int64
```

In [73]:
```
rec_data = np.array([30446,23220,1142])
rec_names = ['Other','Sourcing','Referred']
```

In [74]:
```
emp_data.previous_year_rating.value_counts()
```

Out[74]:
```
3.0    18618
5.0    11741
4.0     9877
1.0     6223
2.0     4225
Name: previous_year_rating, dtype: int64
```

In [75]:
```
pyr_data = np.array([18618,11741,9877,6223,4225])
pyr_names = ['3.0 (***)','5.0 (*****)','4.0 (****)','1.0 (*)','2.0
```

In [76]:
```
emp_data.no_of_trainings.value_counts()
```

Out[76]:
```
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 [77]:
```
not_data = np.array([44378,7987,1776,468,128,44,122,5,5,5])
not_names = ["One","Two","Three","Four","Five","Six","Seven","Eight
```

In [78]:
```
emp_age_data = emp_data.age.value_counts()

emp_age_data
```

Out[78]:
```
30    3665
32    3534
31    3534
29    3405
33    3210
28    3147
34    3076
27    2827
35    2711
36    2517
37    2165
26    2060
38    1923
39    1695
40    1663
25    1299
41    1289
42    1149
43     992
44     847
24     845
45     760
46     697
48     557
47     557
50     521
49     441
23     428
51     389
53     364
52     351
54     313
55     294
56     264
57     238
22     231
60     217
58     213
59     209
20     113
21      98
Name: age, dtype: int64
```

In [79]:
```
ts_min = emp_data.avg_training_score.value_counts().min()
ts_max = emp_data.avg_training_score.value_counts().max()
ts_mean = emp_data.avg_training_score.value_counts().mean()
ts_median = emp_data.avg_training_score.value_counts().median()
```

In [80]:
```
ts_data = [ts_min,ts_mean,ts_max,ts_median]
ts_names = ['Minimum','Mean','Maximum','Median']
```

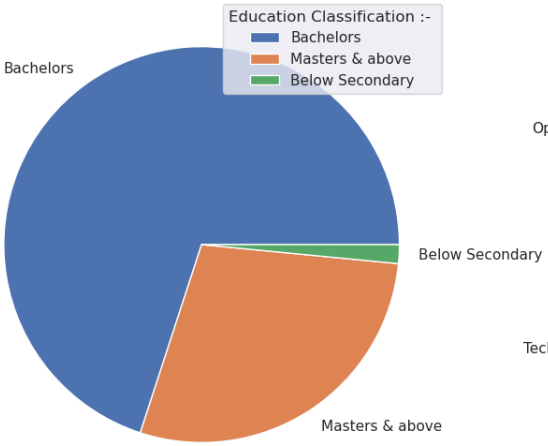
In [81]:
```
emp_data.awards_won.value_counts()
```

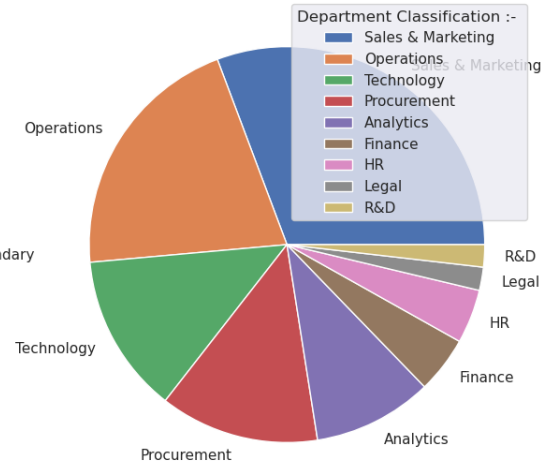
Out[81]:
```
0    53538
1     1270
Name: awards_won, dtype: int64
```

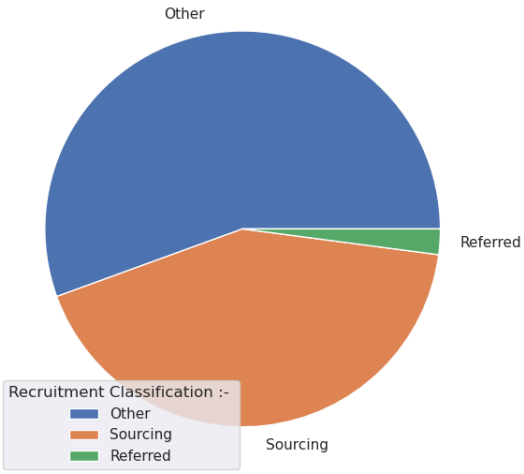
In [82]:
```
awd_data = [53538,1270]
awd_names = ['No','Yes']
```

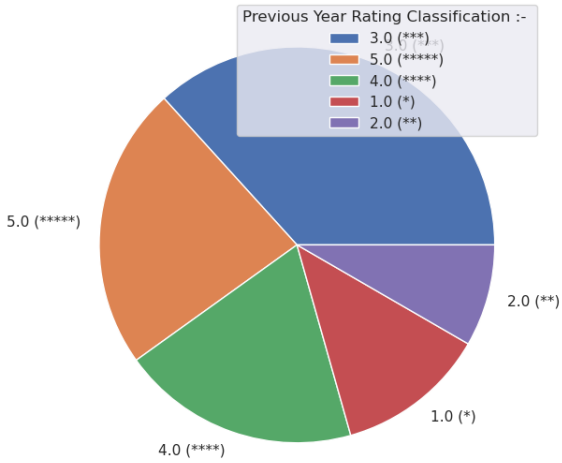
In [83]:
```
print("Uni-variate visualisation of data :- \n")
fig = plt.figure(figsize=(15,25))
b1 = fig.add_subplot(3,2,1)
b2 = fig.add_subplot(3,2,2)
b3 = fig.add_subplot(3,2,3)
b4 = fig.add_subplot(3,2,4)
b5 = fig.add_subplot(3,2,5)
b6 = fig.add_subplot(3,2,6)

# b1

b1.pie(ed_data, labels = ed_names)
b1.legend(title = "Education Classification :- ")

# b2

b2.pie(dep_data, labels = dep_names)
b2.legend(title = "Department Classification :- ")

# b3

b3.pie(rec_data, labels = rec_names)
b3.legend(title = "Recruitment Classification :- ")

# b4

b4.pie(pyr_data, labels = pyr_names)
b4.legend(title = "Previous Year Rating Classification :- ")

# b5

b5.pie(not_data, labels = not_names)
b5.legend(title = "Previous Year Rating Classification :- ")

# b6

b6.pie(awd_data, labels = awd_names)
b6.legend(title = "Awards Won Classification :- ")
plt.show()
```

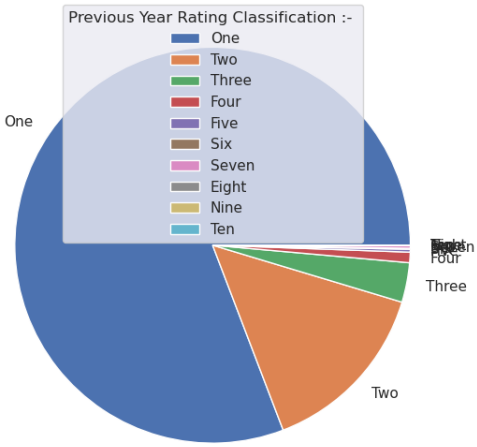
```
Uni-variate visualisation of data :-
```

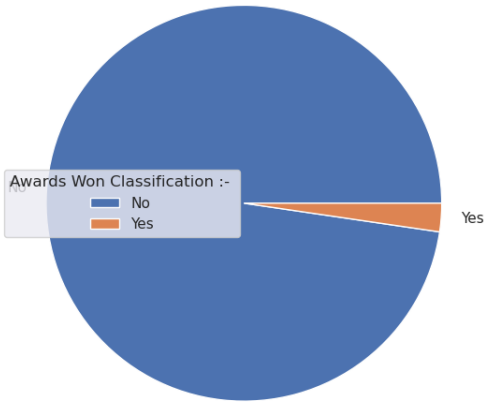

Education Classification :-
Bachelors
Masters & above
Below Secondary
Bachelors
Below Secondary
Masters & above


Department Classification :-
Sales & Marketing
Operations
Technology
Procurement
Analytics
Finance
HR
Legal
R&D
Operations
Technology
Procurement
Analytics
Finance
HR
Legal
R&D


Other
Referred
Sourcing
Recruitment Classification :-
Other
Sourcing
Referred


Previous Year Rating Classification :-
3.0 (***)
5.0 (*****)
4.0 (****)
1.0 (*)
2.0 (**)
5.0 (*****)
2.0 (**)
1.0 (*)
4.0 (****)


Previous Year Rating Classification :-
One
Two
Three
Four
Five
Six
Seven
Eight
Nine
Ten
One
Four
Three
Two


Awards Won Classification :-
No
Yes
Yes

Train Test Split

In [84]:
```
X = emp_data.length_of_service
y = emp_data.age
```

In [85]:
```
from sklearn.datasets import make_classification
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

Random state for X, y for the classification will be 62, while for train_test_split, random state will be 70 for getting more accuracy

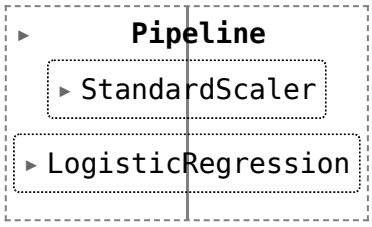
In [86]:
```
X, y = make_classification(random_state=62)
```

Logistic Regression

In [87]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, random_st
```

In [88]:
```
pipe = make_pipeline(StandardScaler(), LogisticRegression())
pipe.fit(X_train, y_train)  # apply scaling on training data
```

Out[88]:



In [89]:
```
lg_score = pipe.score(X_test, y_test)
print(f"\nAccuracy Score (Logistic Regression) between X_test and y_
```

```
Accuracy Score (Logistic Regression) between X_test and y_test :
0.76
```

In [90]:
```
log_model = LogisticRegression()
log_model = log_model.fit(X_train,y_train)
```

In [91]:
```
log_model.coef_
```

Out[91]:
```
array([[ 0.09194538, -0.17283617, -1.00896715,  0.44497664,  0.710
37889,
         0.21369523,  0.5009666 ,  0.17293997,  0.34574511, -0.763
20242,
         0.61621837, -0.87294727, -0.70087922, -0.1340033 , -0.308
31412,
        -0.11937485,  1.64475794, -0.24711697, -1.20735615,  0.064
91954]])
```

In [92]:
```
log_model.intercept_
```

Out[92]:
```
array([-0.97865662])
```

Decision Tree

In [93]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, random_st
```

In [94]:
```
pipe = make_pipeline(StandardScaler(), DecisionTreeClassifier())
pipe.fit(X_train, y_train)  # apply scaling on training data
```

Out[94]:
```
▸ Pipeline
  ▸ StandardScaler
  ▸ DecisionTreeClassifier
```

In [95]:
```
dt_score = pipe.score(X_test, y_test)
print(f"\nAccuracy Score (Decision Tree) between X_test and y_test
```

```
Accuracy Score (Decision Tree) between X_test and y_test : 0.8
```

In [96]:
```
dt_model = LogisticRegression()
dt_model = dt_model.fit(X_train,y_train)
```

In [97]:
```
dt_model.coef_
```

Out[97]:
```
array([[ 0.09194538, -0.17283617, -1.00896715,  0.44497664,  0.710
37889,
         0.21369523,  0.5009666 ,  0.17293997,  0.34574511, -0.763
20242,
         0.61621837, -0.87294727, -0.70087922, -0.1340033 , -0.308
31412,
        -0.11937485,  1.64475794, -0.24711697, -1.20735615,  0.064
91954]])
```

In [98]:
```
dt_model.intercept_
```

Out[98]:
```
array([-0.97865662])
```

Random Forest

In [99]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, random_st
```

In [100]:
```
pipe = make_pipeline(StandardScaler(), RandomForestClassifier())
pipe.fit(X_train, y_train)  # apply scaling on training data
```

Out[100]:
```
▸ Pipeline
  ▸ StandardScaler
  ▸ RandomForestClassifier
```

In [101]:
```
rd_score = pipe.score(X_test, y_test)
print(f"\nAccuracy Score (Random Forest) between X_test and y_test
```

```
Accuracy Score (Random Forest) between X_test and y_test : 0.88
```

In [102]:
```
rdf_model = RandomForestClassifier()
rdf_model = rdf_model.fit(X_train,y_train)
```

In [103]:
```
rdf_model.feature_importances_
```

Out[103]:
```
array([0.02428116, 0.00958306, 0.03634889, 0.01335838, 0.0320331 ,
       0.02375722, 0.0400056 , 0.02471293, 0.01639397, 0.02822725,
       0.01928004, 0.16134408, 0.02982548, 0.01522636, 0.01597183,
       0.01775638, 0.22411597, 0.01766749, 0.22936071, 0.0207500
9])
```

Classification result to show the accuracy score of all the executed models

Creating 'models_accuracy_score'.csv file to store the result

In [104]:
```
import csv
```

In [105]:
```
header = ['Logistic_Regression','Decision_Tree','Random_Forest']
datavalue = [lg_score,dt_score,rd_score]
```

In [106]:
```
with open('models_accuracy_score.csv','w',encoding='UTF8') as f:
    writer = csv.writer(f)
    writer.writerow(header)
    writer.writerow(datavalue)
```

In [107]:
```
# Fetching result

check = pd.read_csv('models_accuracy_score.csv')
check
```

Out[107]:

	Logistic_Regression	Decision_Tree	Random_Forest
0	0.76	0.8	0.88

In [108]:
```
# Visualizing result
fig = plt.figure(figsize=(11,8))
#sns.barplot(header,datavalue,x="Execute Models",y="Accuracy Score"
plt.plot(datavalue,header,marker='*',c='black')
plt.title("\nVisualization of accuracy score of all the executed mo
plt.xlabel("\nAccuracy Score",fontsize=16)
plt.ylabel("Different Executed Models\n",fontsize=16)

plt.show()
```