Importing Libraries

In [1]:

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

Reading and Analysing the Data

In [2]:

```
data=pd.read_csv("11-4-Dataset-Predicting Placement in Campus Recruitment.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   sl_no           215 non-null    int64  
 1   gender          215 non-null    object 
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object 
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object 
 6   hsc_s           215 non-null    object 
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object 
 9   workex          215 non-null    object 
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object 
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object 
 14  salary          148 non-null    float64
dtypes: float64(6), int64(1), object(8)
memory usage: 25.3+ KB
```

In [5]:

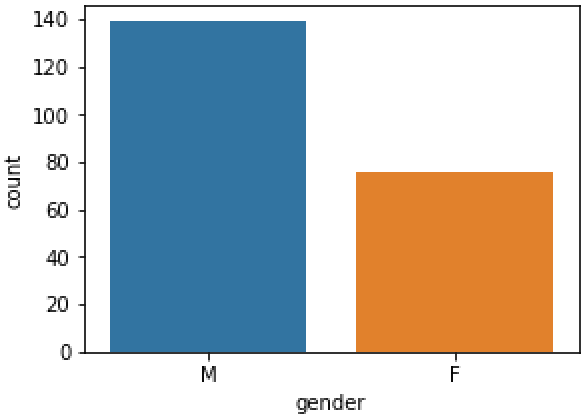
```
data.isnull().sum()
```

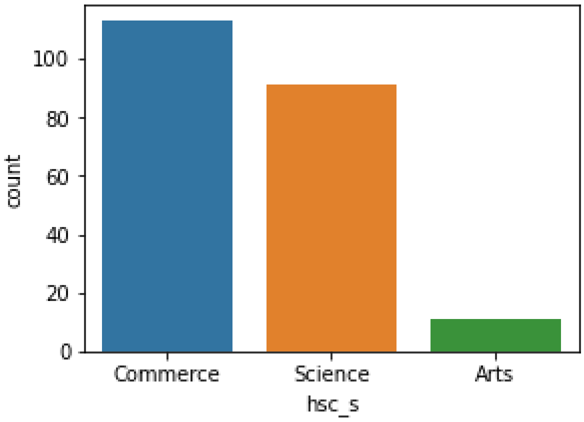
```
Out[5]: sl_no              0
        gender             0
        ssc_p              0
        ssc_b              0
        hsc_p              0
        hsc_b              0
        hsc_s              0
        degree_p           0
        degree_t           0
        workex             0
        etest_p            0
        specialisation     0
        mba_p              0
        status             0
        salary            67
        dtype: int64
```

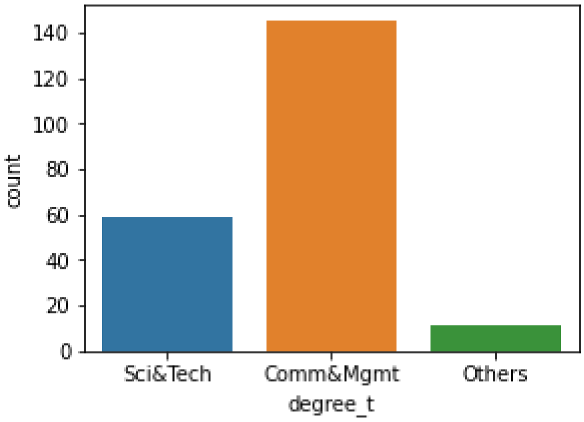
Data Visualization

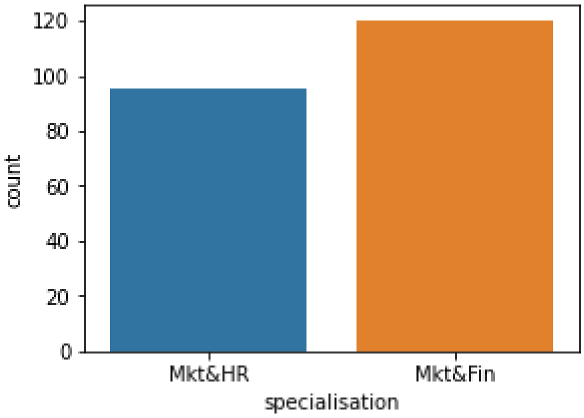
```
In [6]:
print("General Specifications about Data using Count Plot")


plt.figure(figsize = (15,7))
plt.subplot(231)
ax = sns.countplot(x = 'gender', data = data)

plt.figure(figsize = (15,7))
plt.subplot(232)
ax = sns.countplot(x = 'hsc_s', data = data)

plt.figure(figsize = (15,7))
plt.subplot(233)
ax = sns.countplot(x = 'degree_t', data = data)

plt.figure(figsize = (15,7))
plt.subplot(234)
ax = sns.countplot(x = 'specialisation', data = data)

plt.figure(figsize = (15,7))
plt.subplot(235)
ax = sns.countplot(x = 'workex', data = data)

plt.figure(figsize = (15,7))
plt.subplot(236)
ax = sns.countplot(x = 'status', data = data)
```

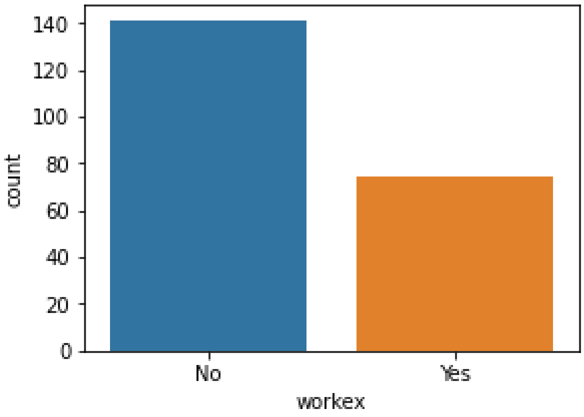
```
General Specifications about Data using Count Plot
```

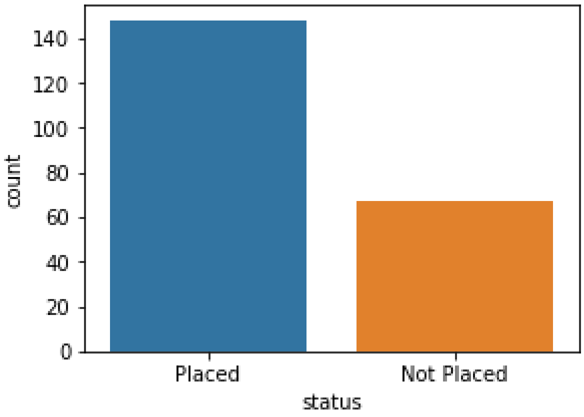



count
0
20
40
60
80
100
Commerce
Science
Arts
hsc_s


count
0
20
40
60
80
100
120
140
Sci&Tech
Comm&Mgmt
Others
degree_t


count
0
20
40
60
80
100
120
Mkt&HR
Mkt&Fin
specialisation


count
0
20
40
60
80
100
120
140
No
Yes
workex



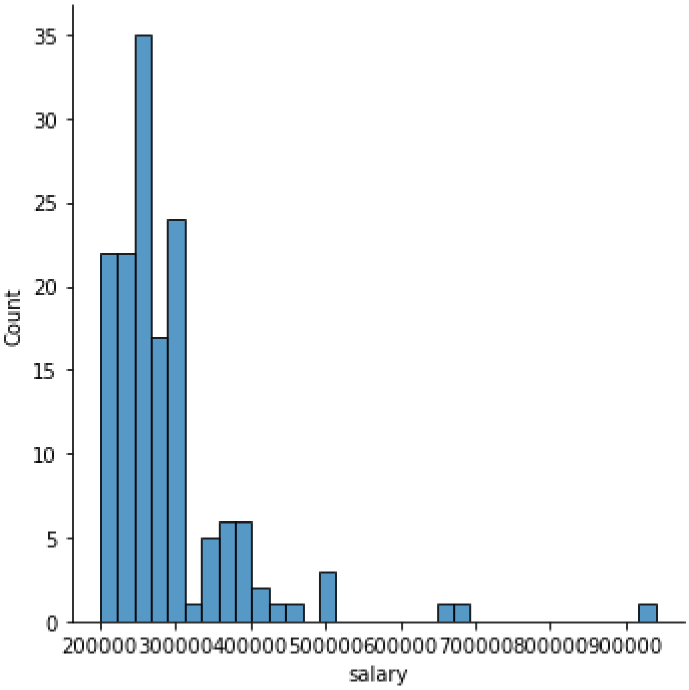
In [7]:

```
placed = data[data.salary != 0]
print("Graph to show Salary Distribution: ")
sns.displot(placed['salary'])
```

```
Graph to show Salary Distribution:
```

Out[7]:

```
<seaborn.axisgrid.FacetGrid at 0x2819ae34550>
```

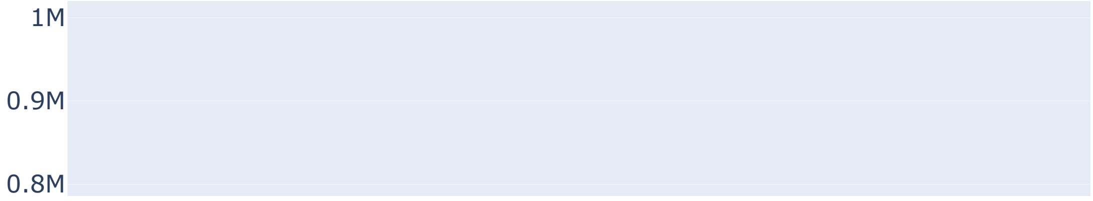


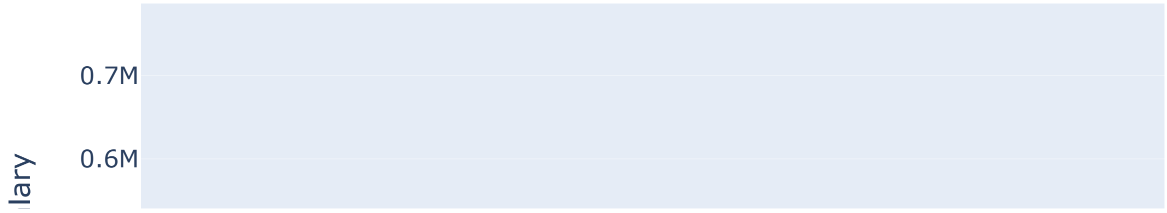
In [ ]:

In [8]:

```
import plotly_express as px
print("Distribution of Male and Female across different Specialisations and Salary")
px.violin(placed, y = 'salary', x = 'specialisation', color = 'gender', box = True,
```

```
Distribution of Male and Female across different Specialisations and Salary
```





Dropping unnecessary Columns

In [9]:

```
data.drop(['salary','ssc_b','hsc_b','sl_no'],inplace=True, axis=1)
```

In [10]:

```
data.head()
```

Out[10]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_
0	M	67.00	91.00	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.8
1	M	79.33	78.33	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.2
2	M	65.00	68.00	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.8
3	M	56.00	52.00	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.4
4	M	85.80	73.60	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.5

Labeling the Data

In [11]:

```
from sklearn.preprocessing import LabelEncoder
le_gender = LabelEncoder()
le_hscs = LabelEncoder()
le_degreet = LabelEncoder()
le_workex = LabelEncoder()
le_specialisation = LabelEncoder()
le_status = LabelEncoder()

data['gender'] = le_gender.fit_transform(data['gender'])
data['degree_t'] = le_degreet.fit_transform(data['degree_t'])
data['workex'] = le_workex.fit_transform(data['workex'])
data['specialisation'] = le_specialisation.fit_transform(data['specialisation'])
```

```
data['status'] = le_status.fit_transform(data['status'])
data['hsc_s'] = le_hscs.fit_transform(data['hsc_s'])
```

Splitting the Data into 2 sets for predictions

i.e. one df for input and other as predictions

In [12]:

```
X=data.drop('status',axis=1)
y=data['status']
print(X)
print(y)
```

```
     gender  ssc_p  hsc_p  hsc_s  degree_p  degree_t  workex  etest_p  \
0         1  67.00  91.00      1     58.00         2       0     55.0
1         1  79.33  78.33      2     77.48         2       1     86.5
2         1  65.00  68.00      0     64.00         0       0     75.0
3         1  56.00  52.00      2     52.00         2       0     66.0
4         1  85.80  73.60      1     73.30         0       0     96.8
..      ...    ...    ...    ...       ...       ...     ...      ...
210       1  80.60  82.00      1     77.60         0       0     91.0
211       1  58.00  60.00      2     72.00         2       0     74.0
212       1  67.00  67.00      1     73.00         0       1     59.0
213       0  74.00  66.00      1     58.00         0       0     70.0
214       1  62.00  58.00      2     53.00         0       0     89.0

     specialisation  mba_p
0                 1  58.80
1                 0  66.28
2                 0  57.80
3                 1  59.43
4                 0  55.50
..              ...    ...
210               0  74.49
211               0  53.62
212               0  69.72
213               1  60.23
214               1  60.22

[215 rows x 10 columns]
0      1
1      1
2      1
3      0
4      1
      ..
210    1
211    1
212    1
213    1
214    0
Name: status, Length: 215, dtype: int32
```

In [13]:

```
X.head()
```

Out[13]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p
0	1	67.00	91.00	1	58.00	2	0	55.0	1	58.80
1	1	79.33	78.33	2	77.48	2	1	86.5	0	66.28

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p
2	1	65.00	68.00	0	64.00	0	0	75.0	0	57.80
3	1	56.00	52.00	2	52.00	2	0	66.0	1	59.43
4	1	85.80	73.60	1	73.30	0	0	96.8	0	55.50

In [14]:

```
y.head()
```

Out[14]:

```
0    1
1    1
2    1
3    0
4    1
Name: status, dtype: int32
```

Train-Test-Split

In [15]:

```
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_state=1)
```

In [16]:

```
data.describe()
```

Out[16]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.0000
mean	0.646512	67.303395	66.333163	1.372093	66.370186	0.600000	0.344186	72.1005
std	0.479168	10.827205	10.897509	0.580978	7.358743	0.890238	0.476211	13.2759
min	0.000000	40.890000	37.000000	0.000000	50.000000	0.000000	0.000000	50.0000
25%	0.000000	60.600000	60.900000	1.000000	61.000000	0.000000	0.000000	60.0000
50%	1.000000	67.000000	65.000000	1.000000	66.000000	0.000000	0.000000	71.0000
75%	1.000000	75.700000	73.000000	2.000000	72.000000	2.000000	1.000000	83.5000
max	1.000000	89.400000	97.700000	2.000000	91.000000	2.000000	1.000000	98.0000

Model Seletion and Training

In [17]:

```
from sklearn.ensemble import RandomForestClassifier

model=RandomForestClassifier(n_estimators=50, criterion='entropy', random_state=1)
model.fit(X_train,y_train)
```

Out[17]:

```
RandomForestClassifier(criterion='entropy', n_estimators=50, random_state=1)
```

Predictions

In [18]:

```
predictions=model.predict(X_test)
predictions
```

Out[18]:

```
array([1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1])
```

Evalution of the Model

In [19]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
```

In [20]:

```
print('Accuracy score:', accuracy_score(y_test,predictions))
```

```
Accuracy score: 0.8837209302325582
```

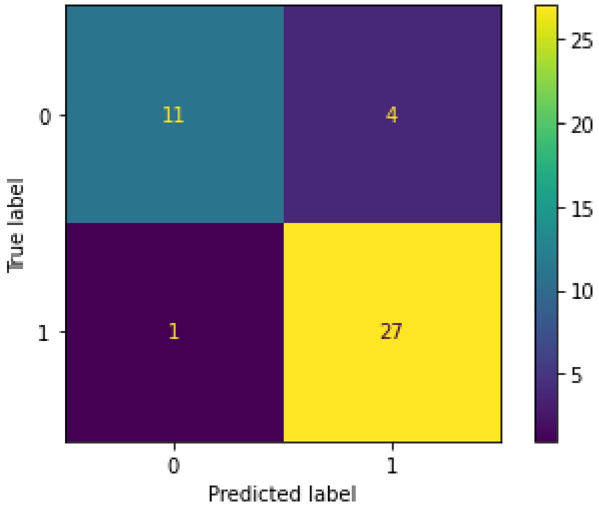
In [21]:

```
print('confusion_matrix:\n' )
cm = confusion_matrix(y_test,predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot()
```

```
confusion_matrix:
```

Out[21]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2819d596700>
```



In [22]:

```
print('classification report:\n', classification_report(y_test,predictions))
```

```
classification report:
               precision    recall  f1-score   support

           0       0.92      0.73      0.81        15
           1       0.87      0.96      0.92        28

    accuracy                           0.88        43
   macro avg       0.89      0.85      0.87        43
weighted avg       0.89      0.88      0.88        43
```

In [23]:

```
print('mean absolute error:\n', mean_absolute_error(y_test,predictions))
```

```
mean absolute error:
 0.11627906976744186
```

In [24]:

```
#random forest=0.88
#svm=0.86
#logistic regression = 0.86
#decision tree=0.76
# k neighbors=0.81
```