Importing neccessary Libraries

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

import warnings
warnings.filterwarnings('ignore')
```

Importing Data set

In [2]:
```python
df=pd.read_csv("Placement_Data_Full_Class.csv")
df.head()
```

Out[2]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

EDA

In [3]:
```python
df.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: 18.5+ KB
```

As we seen in data, there is missing values present in Salary column. We will handle missing values in further process.

Total No Features = 15

Total no of Observations = 215

In [4]:
```python
df.describe()
```

Out[4]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [5]:
```python
df.columns
```

Out[5]:
```
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
```

In [6]:
```python
df.dtypes
```

Out[6]:
```
sl_no               int64
gender             object
ssc_p             float64
ssc_b              object
hsc_p             float64
hsc_b              object
hsc_s              object
degree_p          float64
degree_t           object
workex             object
etest_p           float64
specialisation     object
mba_p             float64
status             object
salary            float64
dtype: object
```

In [7]:
```python
df.shape
```

Out[7]:
```
(215, 15)
```

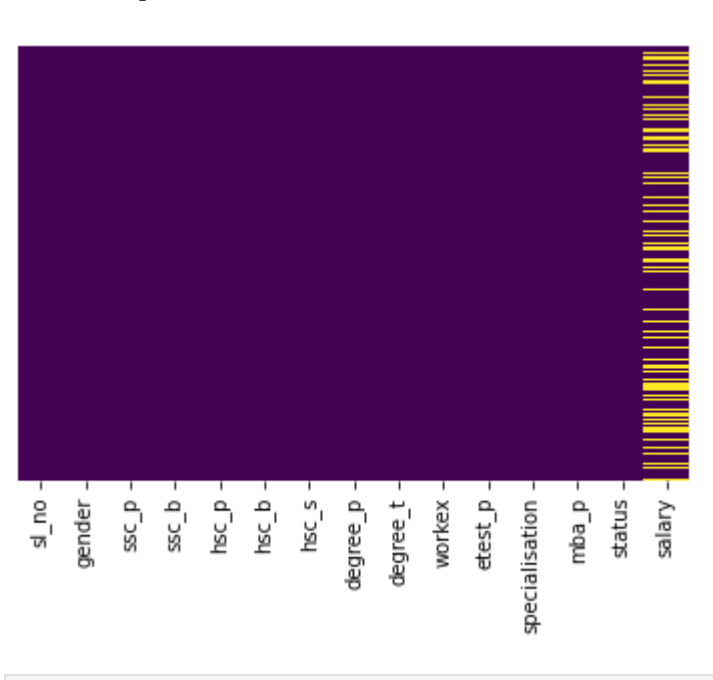
In [8]:
```python
df.size
```

Out[8]:
```
3225
```

Handling missing values

In [9]:
```python
sns.heatmap(df.isna(),yticklabels=False,cbar=False,cmap="viridis")
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```python
df.isna().sum()
```

Out[10]:
```
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
```

There are 67 null values in our data We can't drop these values as this will provide a valuable information on why candidates failed to get hired. We can't impute it with mean values and because there is no any criteria to give average salary irresepective of students qualifications and thier performance.it will have adverse effect as salaries migh be shown to the students those are not hired. so It should be the best way to deal with these missing values is to fill it with '0' which shows they don't get any salary or 0 salary

In [11]:
```python
# Handling missing values

df["salary"].fillna(0,inplace=True)
```

In [12]:
```python
df.isna().sum()
```

Out[12]:
```
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            0
dtype: int64
```

In [13]:
```python
df.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          215 non-null    float64
dtypes: float64(6), int64(1), object(8)
memory usage: 18.5+ KB
```

Feature Selection

In [14]:
```python
df.drop(['sl_no','ssc_b','hsc_b'],axis=1,inplace=True)
df.head()
```

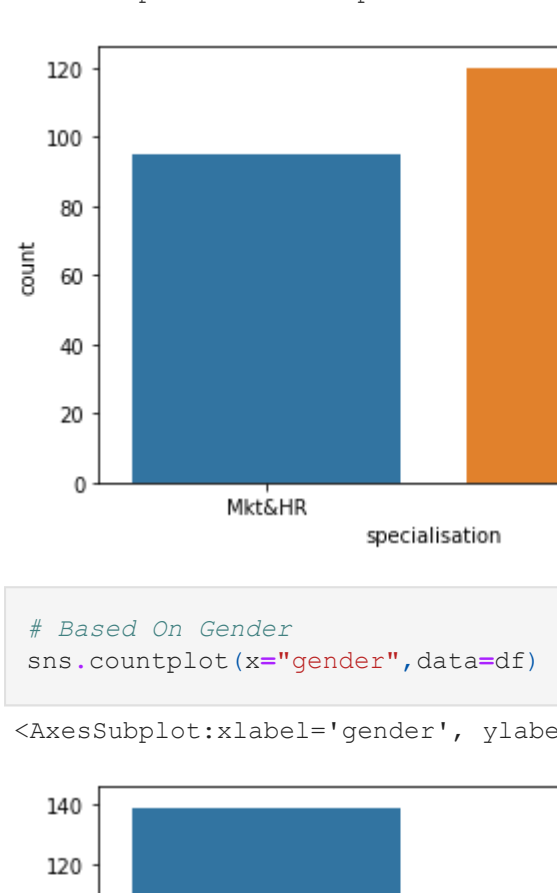
Out[14]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	M	67.00	91.00	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	M	79.33	78.33	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	M	65.00	68.00	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	M	56.00	52.00	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	0.0
4	M	85.80	73.60	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

Data Visualization

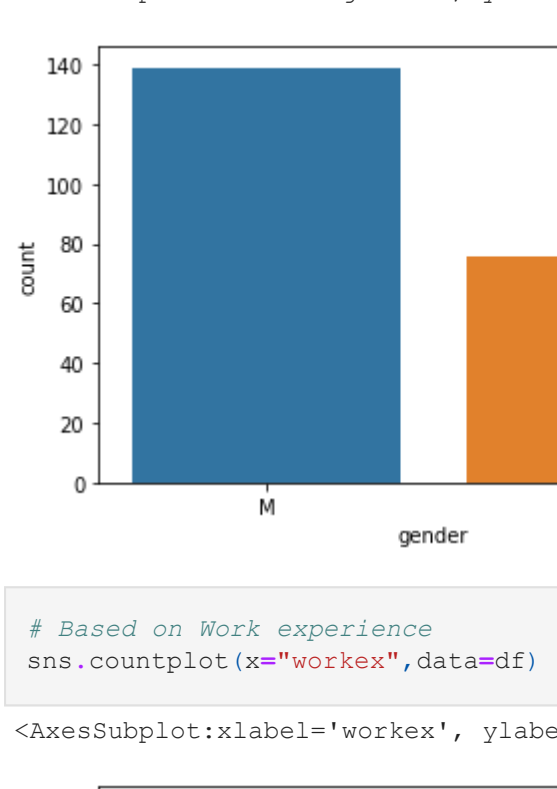
In [15]:
```python
# Based On Specialization
sns.countplot(x="specialisation",data=df)
```

Out[15]:
```
<AxesSubplot:xlabel='specialisation', ylabel='count'>
```



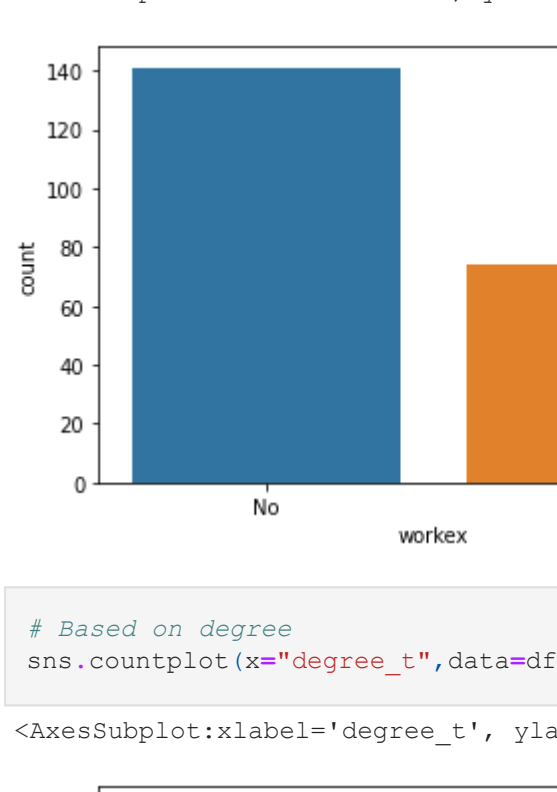
In [16]:
```python
# Based On Gender
sns.countplot(x="gender",data=df)
```

Out[16]:
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



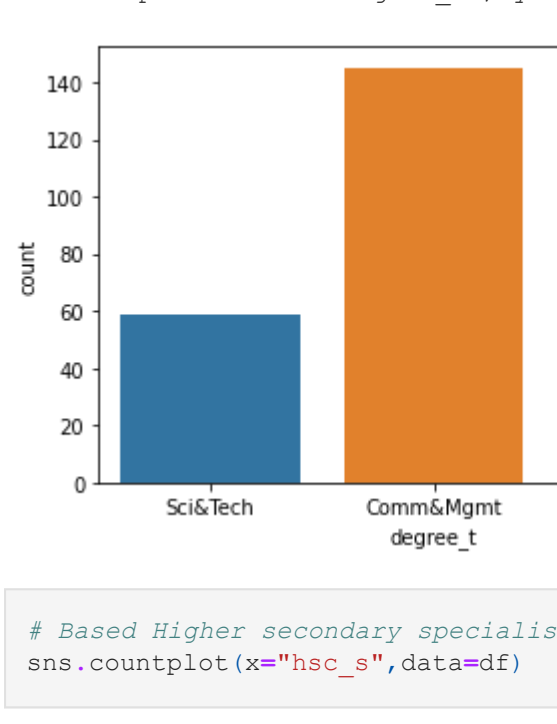
In [17]:
```python
# Based on Work experience
sns.countplot(x="workex",data=df)
```

Out[17]:
```
<AxesSubplot:xlabel='workex', ylabel='count'>
```



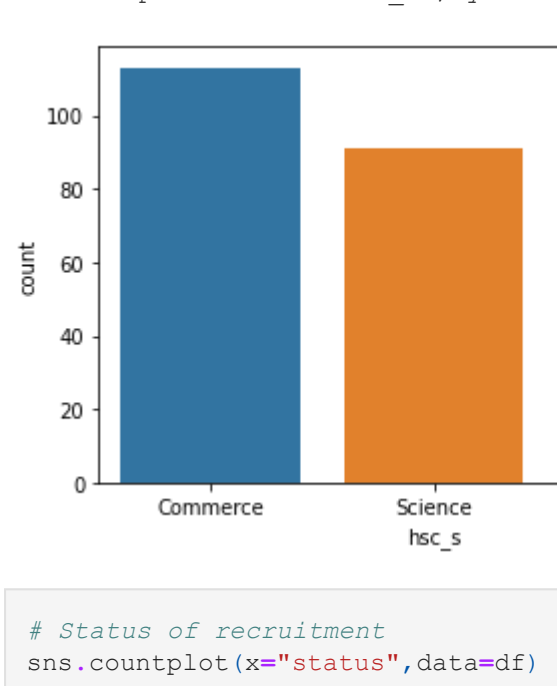
In [18]:
```python
# Based on degree
sns.countplot(x="degree_t",data=df)
```

Out[18]:
```
<AxesSubplot:xlabel='degree_t', ylabel='count'>
```



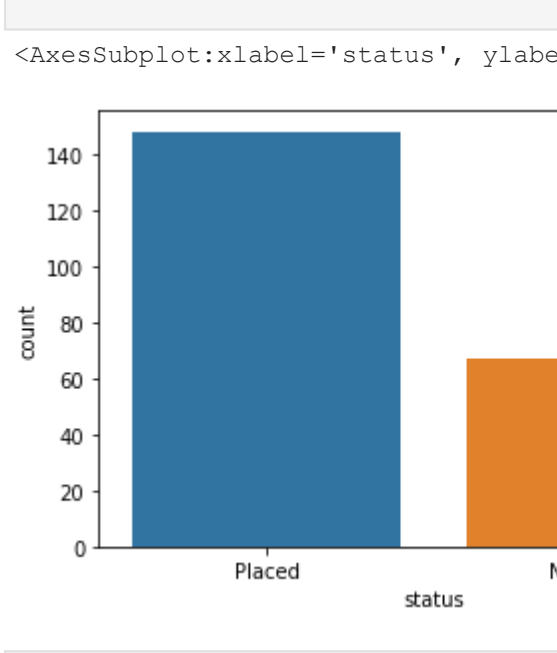
In [19]:
```python
# Based Higher secondary specialisation
sns.countplot(x="hsc_s",data=df)
```

Out[19]:
```
<AxesSubplot:xlabel='hsc_s', ylabel='count'>
```

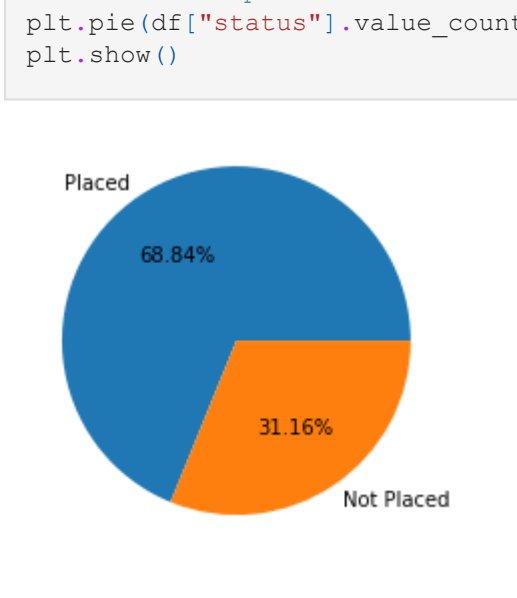


In [20]:
```python
# Status of recruitment
sns.countplot(x="status",data=df)
```

Out[20]:
```
<AxesSubplot:xlabel='status', ylabel='count'>
```



In [21]:
```python
# Does work experience matter in recruitment?
plt.pie(df["status"].value_counts(), labels=df["status"].value_counts().index, autopct='%1.2f%%')
plt.show()
```



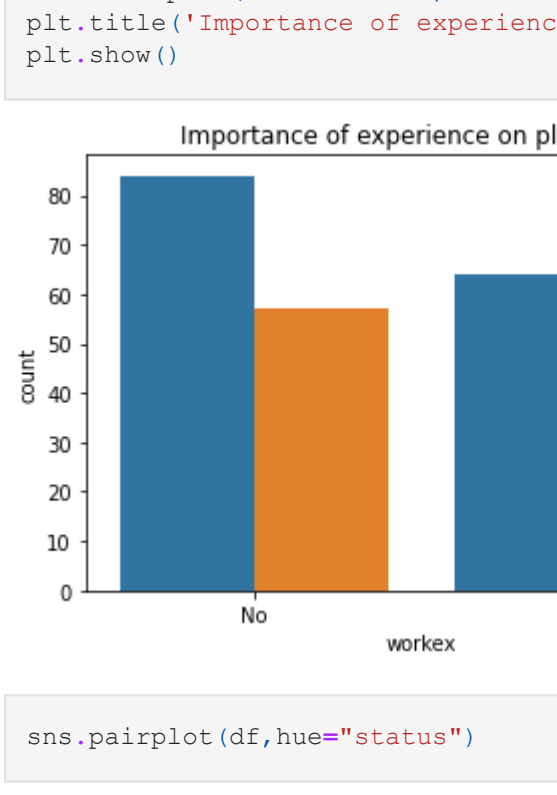
In [22]:
```python
df["workex"].value_counts()
```

Out[22]:
```
No     141
Yes     74
Name: workex, dtype: int64
```

In [23]:
```python
df["status"].value_counts()
```

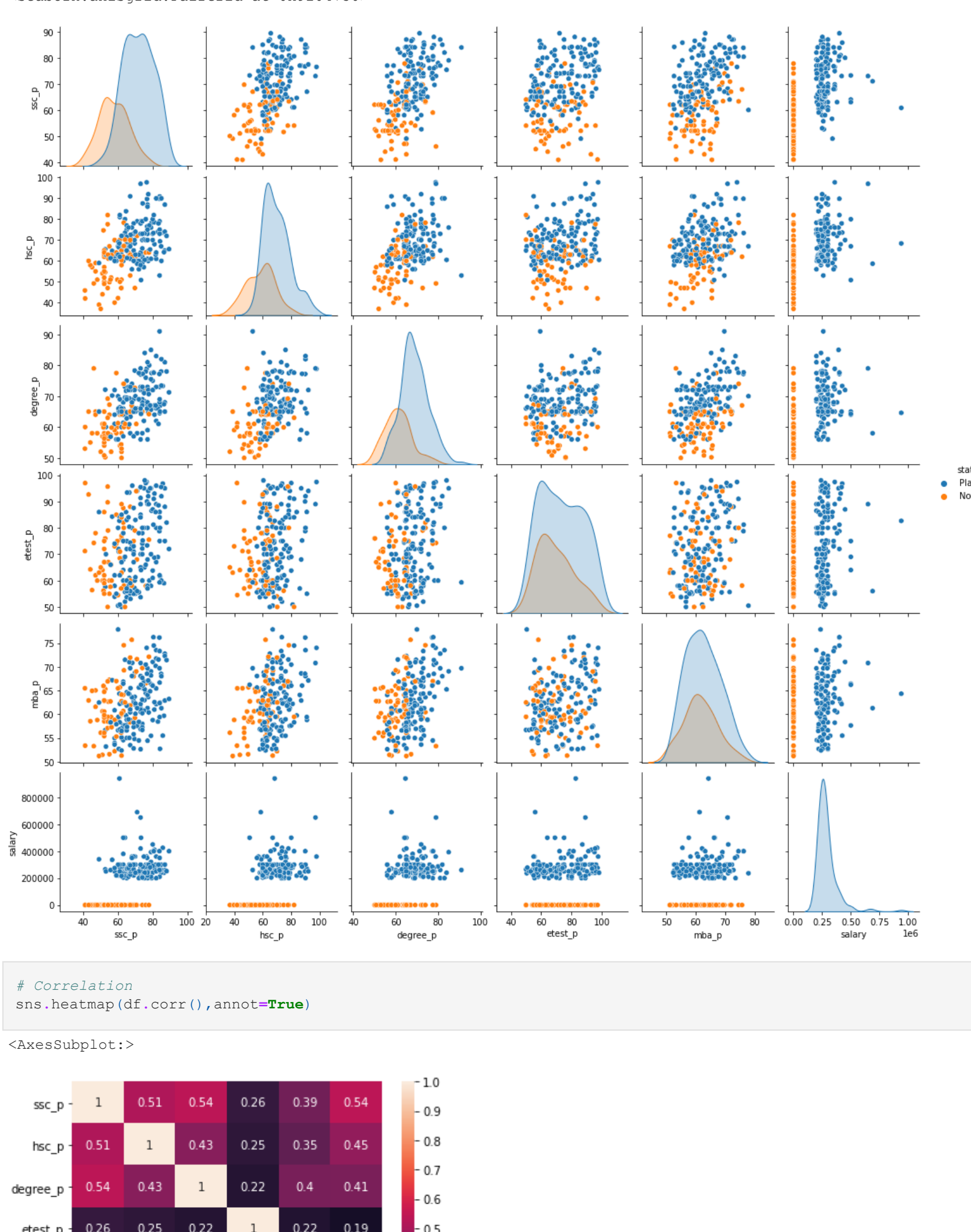
Out[23]:
```
Placed        148
Not Placed     67
Name: status, dtype: int64
```

In [24]:
```python
sns.countplot(x = 'workex',hue = "status",data = df)
plt.title('Importance of experience on placement')
plt.show()
```



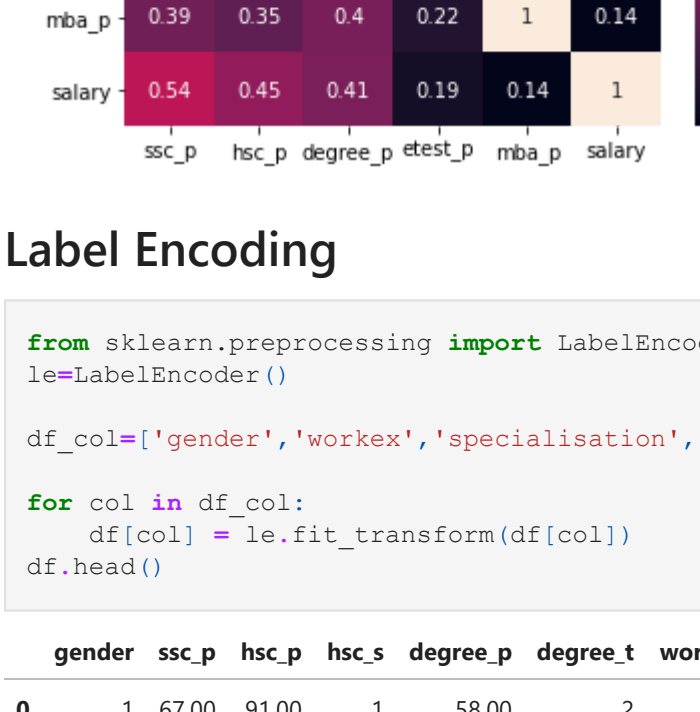
In [25]:
```python
sns.pairplot(df,hue="status")
```

Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x9204730>
```



In [26]:
```python
# Correlation
sns.heatmap(df.corr(),annot=True)
```

Out[26]:
```
<AxesSubplot:>
```



Label Encoding

In [27]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

df_col=['gender','workex','specialisation','status','degree_t','hsc_s']

for col in df_col:
    df[col] = le.fit_transform(df[col])
df.head()
```

Out[27]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	67.00	91.00	1	58.00	2	0	55.0	1	58.80	1	270000.0
1	1	79.33	78.33	2	77.48	2	1	86.5	0	66.28	1	200000.0
2	1	65.00	68.00	0	64.00	0	0	75.0	0	57.80	1	250000.0
3	1	56.00	52.00	2	52.00	2	0	66.0	1	59.43	0	0.0
4	1	85.80	73.60	1	73.30	0	0	96.8	0	55.50	1	425000.0

Dividing data into the target(y) and Independant variable(x)

In [28]:
```python
x=df.drop(["status"],axis=1)
x
```

Out[28]:

	gender	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	salary
0	1	67.00	91.00	1	58.00	2	0	55.0	1	58.80	270000.0
1	1	79.33	78.33	2	77.48	2	1	86.5	0	66.28	200000.0
2	1	65.00	68.00	0	64.00	0	0	75.0	0	57.80	250000.0
3	1	56.00	52.00	2	52.00	2	0	66.0	1	59.43	0.0
4	1	85.80	73.60	1	73.30	0	0	96.8	0	55.50	425000.0
...	...	...	...	...	...	...	...	...	...	...	...
210	1	80.60	82.00	1	77.60	0	0	91.0	0	74.49	400000.0
211	1	58.00	60.00	2	72.00	2	0	74.0	0	53.62	275000.0
212	1	67.00	67.00	1	73.00	0	1	59.0	0	69.72	295000.0
213	0	74.00	66.00	1	58.00	0	0	70.0	1	60.23	204000.0
214	1	62.00	58.00	2	53.00	0	0	89.0	1	60.22	0.0

215 rows × 11 columns

In [29]:
```python
y=df["status"]
y
```

Out[29]:
```
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
```

Splitting the Data into Training and testing data

In [30]:
```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=0)
```

Creating Machine Learning Model

In [31]:
```python
# Using Logistic regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(xtrain,ytrain)
```

Out[31]:
```
LogisticRegression()
```

In [32]:
```python
ypred=logreg.predict(xtest)
ypred
```

Out[32]:
```
array([0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1])
```

In [33]:
```python
ytest
```

Out[33]:
```
198    0
37     1
89     1
168    0
171    1
75     0
96     1
137    1
5      0
83     1
55     1
145    1
160    1
112    1
74     1
203    1
126    1
12     0
153    1
158    0
169    0
141    0
209    1
190    0
144    0
18     0
185    1
15     1
86     1
71     1
7      1
63     0
143    1
97     0
136    0
162    1
33     1
154    1
90     1
211    1
106    0
181    0
139    1
16     1
135    1
122    1
22     1
80     1
45     0
76     1
4      1
108    1
197    1
44     1
Name: status, dtype: int32
```

Evaluation of model

In [34]:
```python
from sklearn.metrics import confusion_matrix as cm,classification_report as cr,accuracy_score as acc
print(f"confusion matrix:-\n {cm(ytest,ypred)}")
print(f"\nAccuracy:- {acc(ytest,ypred)}")
```

```
confusion matrix:-
 [[17  0]
 [ 0 37]]

Accuracy:- 1.0
```

In [35]:
```python
print(f"Classification Report:-\n {cr(ytest,ypred)}")
```

```
Classification Report:-
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        37

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```