```python
In [1]: # importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

```python
In [2]: # reading a dataset
placement = pd.read_csv("Placement.csv")
```

```python
In [3]: # reading first 10 in the dataset
placement.head(10)
```

Out[3]:

	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	55.00	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.50	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.00	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.00	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.80	Mkt&Fin	55.50	Placed	425000.0
5	6	M	55.00	Others	49.80	Others	Science	67.25	Sci&Tech	Yes	55.00	Mkt&Fin	51.58	Not Placed	NaN
6	7	F	46.00	Others	49.20	Others	Commerce	79.00	Comm&Mgmt	No	74.28	Mkt&Fin	53.29	Not Placed	NaN
7	8	M	82.00	Central	64.00	Central	Science	66.00	Sci&Tech	Yes	67.00	Mkt&Fin	62.14	Placed	252000.0
8	9	M	73.00	Central	79.00	Central	Commerce	72.00	Comm&Mgmt	No	91.34	Mkt&Fin	61.29	Placed	231000.0
9	10	M	58.00	Central	70.00	Central	Commerce	61.00	Comm&Mgmt	No	54.00	Mkt&Fin	52.21	Not Placed	NaN

```python
In [4]: # defining a placement copy variable
placement_copy=placement.copy()
```

```python
In [5]: # knowing the shape of the dataset
placement_copy.shape
```

```
Out[5]: (215, 15)
```

```python
In [6]: # to define the datatype of the dataset
placement_copy.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
```

```python
In [7]: # to define the average
placement_copy.isnull().sum()
```

```
Out[7]:
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
```

```python
In [8]: # condition for printing the axis and the required fields
placement_copy.drop(['sl_no','ssc_b','hsc_b'], axis = 1 , inplace = True)
```

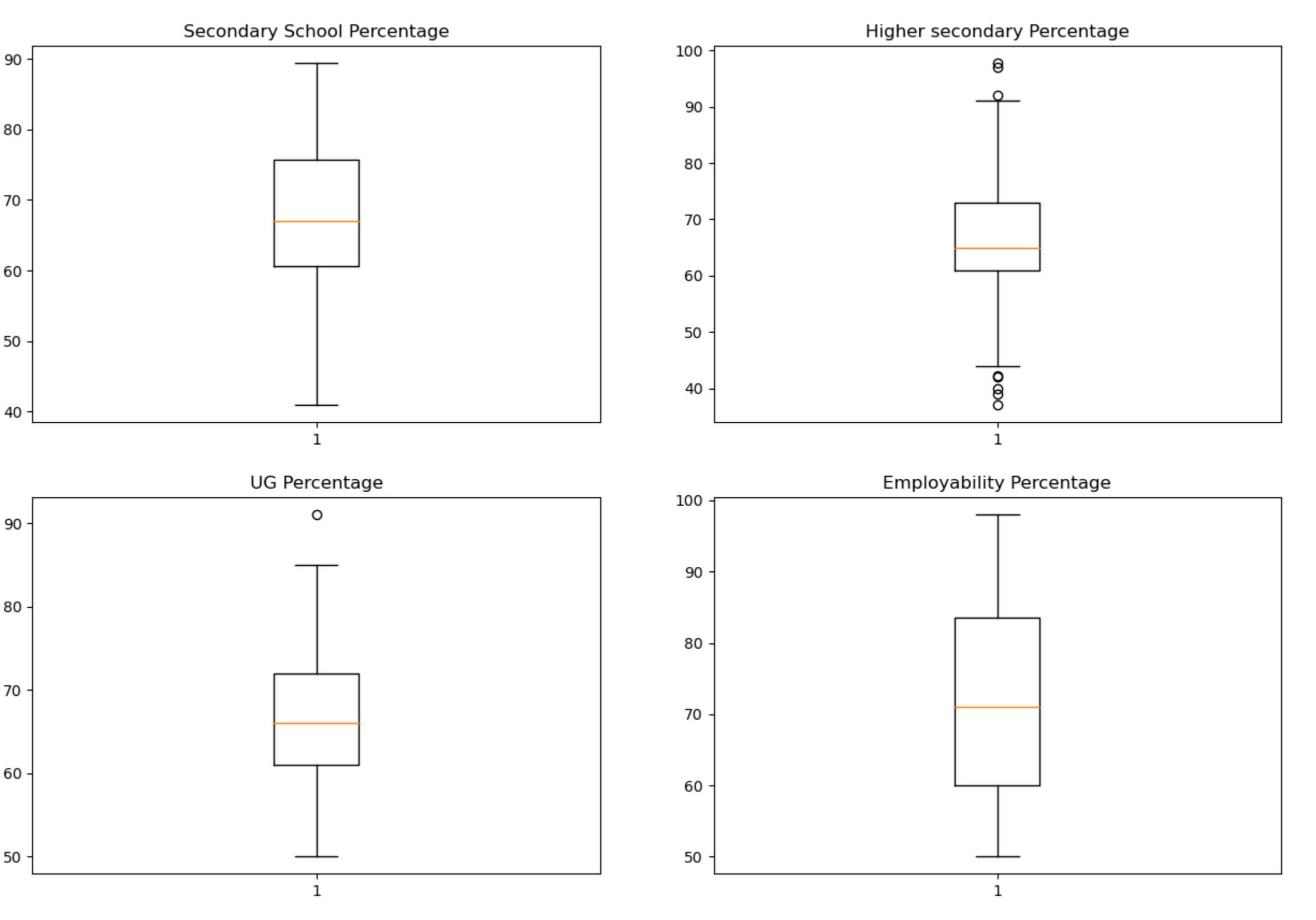
```python
In [9]: # to display the elements
placement_copy.head()
```

Out[9]:

	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	NaN
4	M	85.80	73.60	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

```python
In [10]: # plotting the figure according to percentage
plt.figure(figsize = (15,10))

ax = plt.subplot(221)
plt.boxplot(placement_copy['ssc_p'])
ax.set_title('Secondary School Percentage')

ax = plt.subplot(222)
plt.boxplot(placement_copy['hsc_p'])
ax.set_title('Higher secondary Percentage')

ax = plt.subplot(223)
plt.boxplot(placement_copy['degree_p'])
ax.set_title('UG Percentage')

ax = plt.subplot(224)
plt.boxplot(placement_copy['etest_p'])
ax.set_title('Employability Percentage')
```

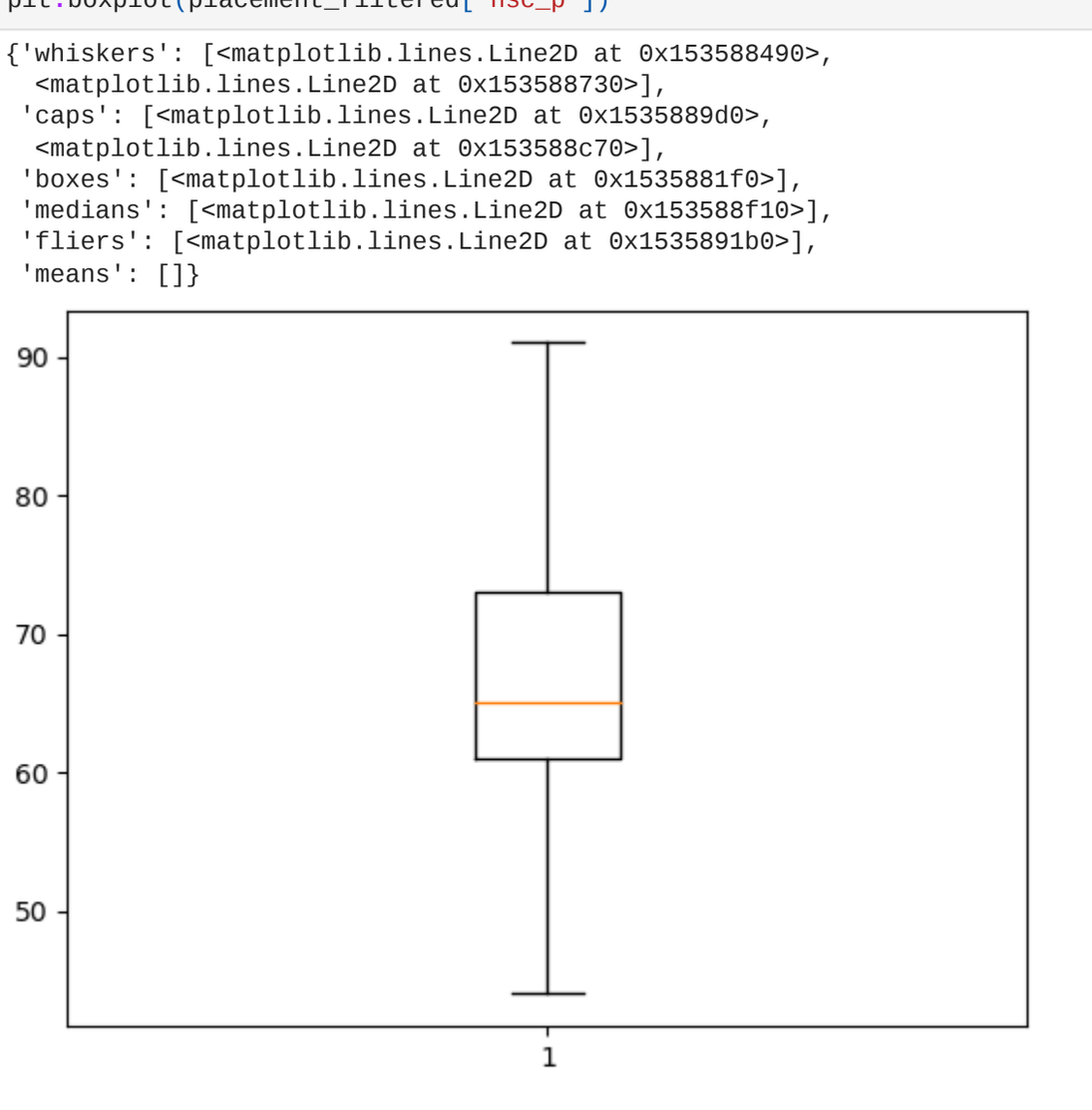
```
Out[10]: Text(0.5, 1.0, 'Employability Percentage')
```



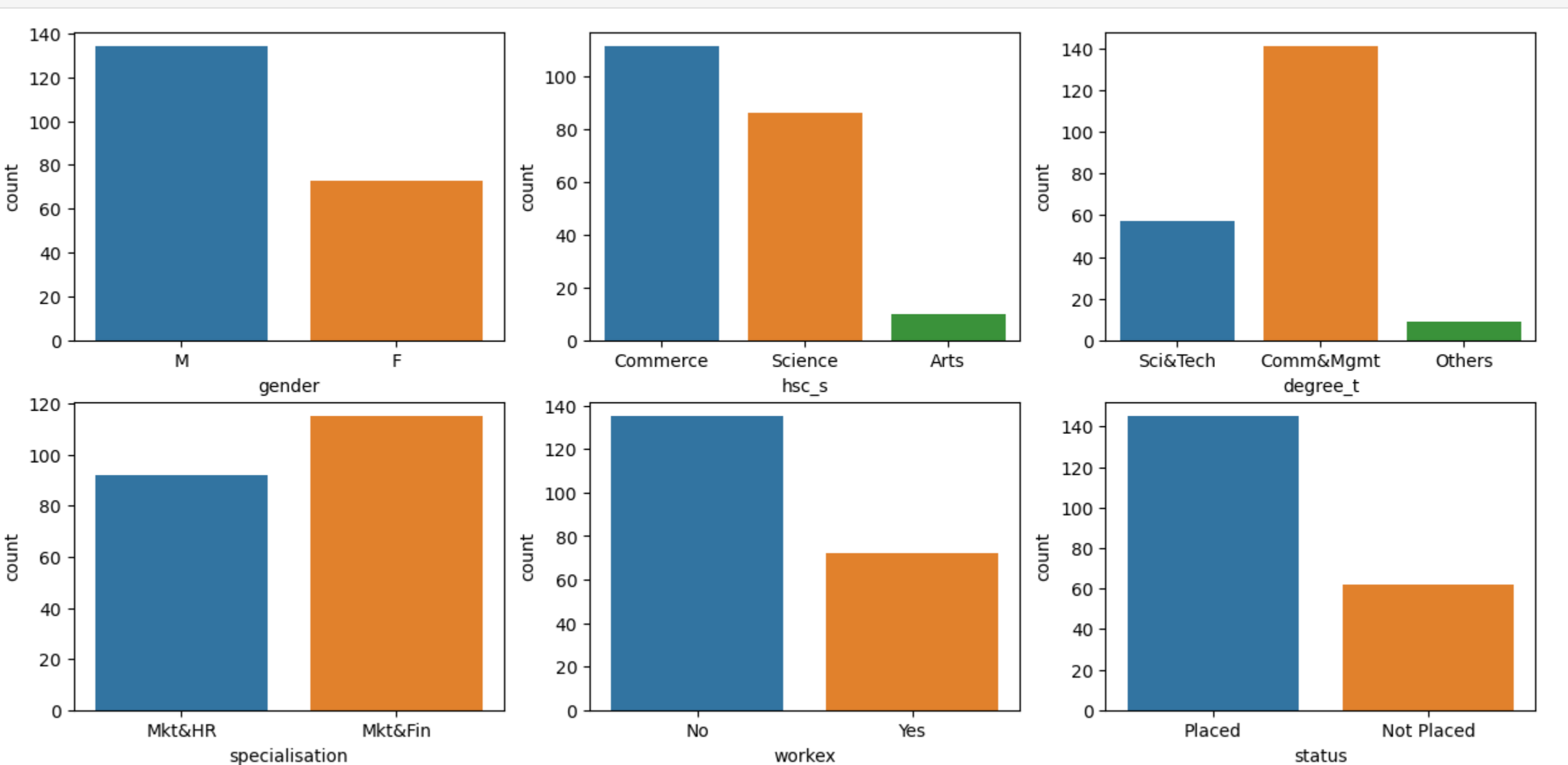

```python
In [14]: # definig the variable
Q1 = placement_copy['hsc_p'].quantile(0.25)
Q3 = placement_copy['hsc_p'].quantile(0.75)
IQR = Q3 - Q1

filter = (placement_copy['hsc_p'] >= Q1 - 1.5 * IQR) & (placement_copy['hsc_p']<= Q3+ 1.5*IQR)
placement_filtered= placement_copy.loc[filter]
```

```python
In [13]: # visualizing the output
plt.boxplot(placement_filtered['hsc_p'])
```

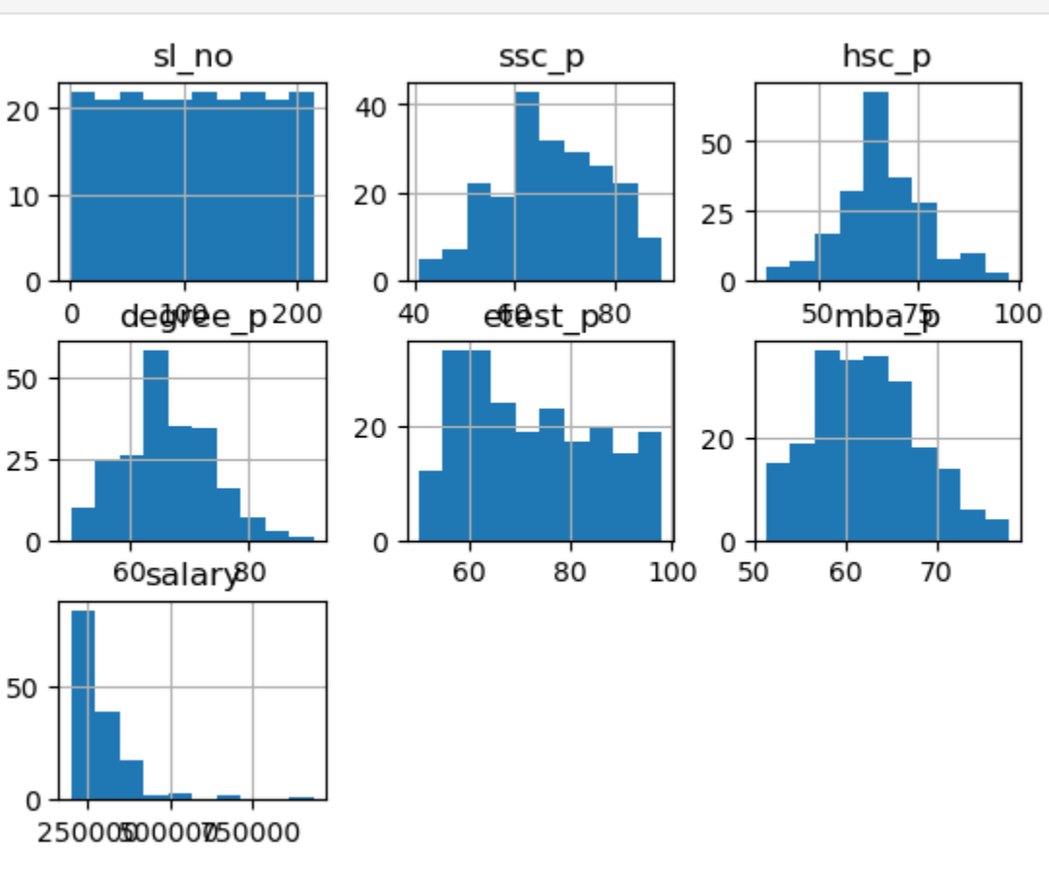
```
Out[13]: {'whiskers': [<matplotlib.lines.Line2D at 0x153588490>,
  <matplotlib.lines.Line2D at 0x153588730>],
 'caps': [<matplotlib.lines.Line2D at 0x1535889d0>,
  <matplotlib.lines.Line2D at 0x153588c70>],
 'boxes': [<matplotlib.lines.Line2D at 0x1535881f0>],
 'medians': [<matplotlib.lines.Line2D at 0x153588f10>],
 'fliers': [<matplotlib.lines.Line2D at 0x1535891b0>],
 'means': []}
```



```python
In [15]: # plotting the figure using seaborn(sns)
plt.figure(figsize = (15,7))

plt.subplot(231)
ax = sns.countplot(x= 'gender' , data = placement_filtered)

plt.subplot(232)
ax = sns.countplot(x= 'hsc_s' , data = placement_filtered)

plt.subplot(233)
ax = sns.countplot(x= 'degree_t' , data = placement_filtered)

plt.subplot(234)
ax = sns.countplot(x= 'specialisation' , data = placement_filtered)

plt.subplot(235)
ax = sns.countplot(x= 'workex' , data = placement_filtered)

plt.subplot(236)
ax = sns.countplot(x= 'status' , data = placement_filtered)
```

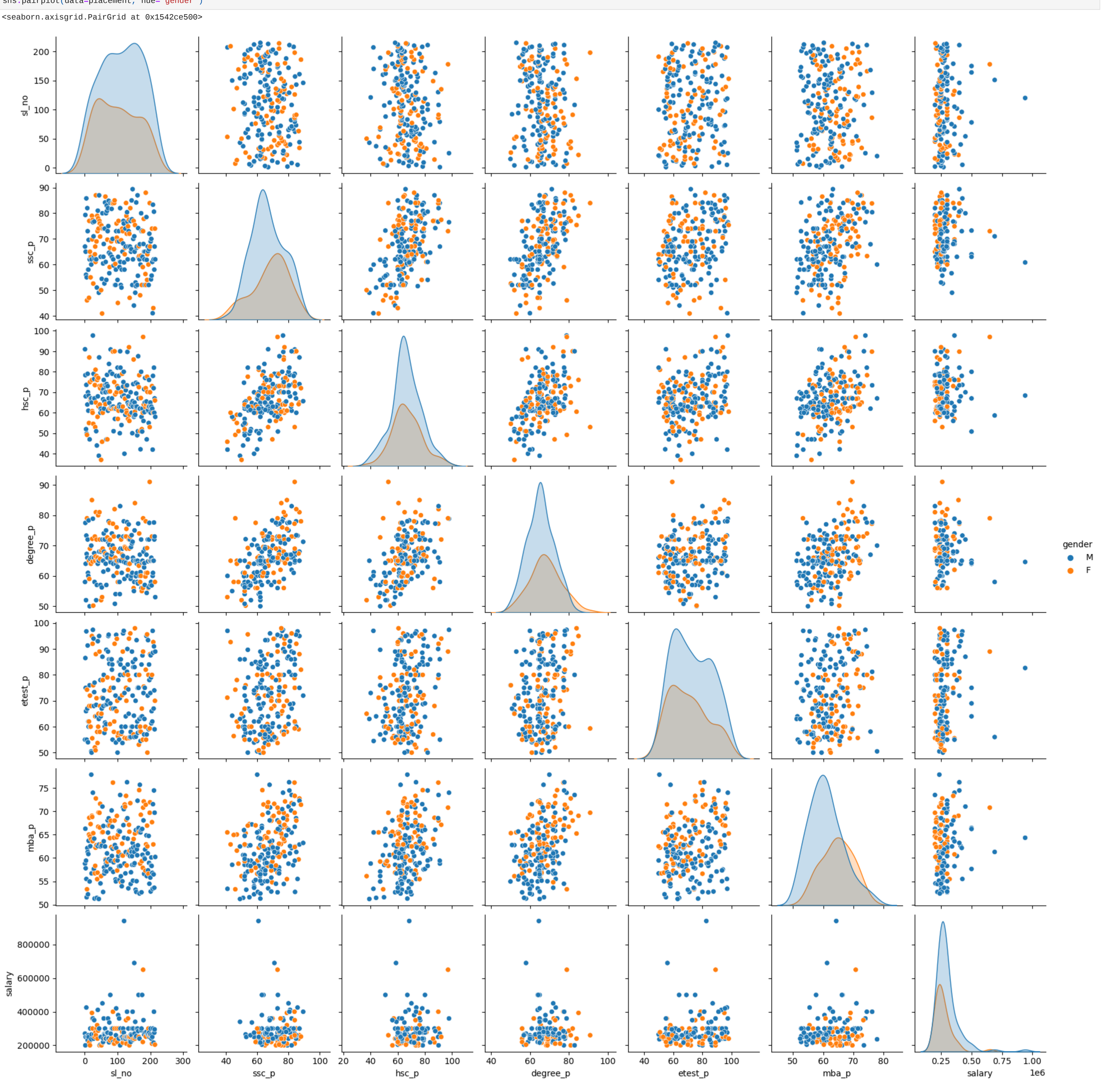



```python
In [16]: # analyzing with histogram model
placement.hist()
plt.show()
```




```python
In [17]: # pair plotting with all the models
sns.pairplot(data=placement, hue='gender')
```

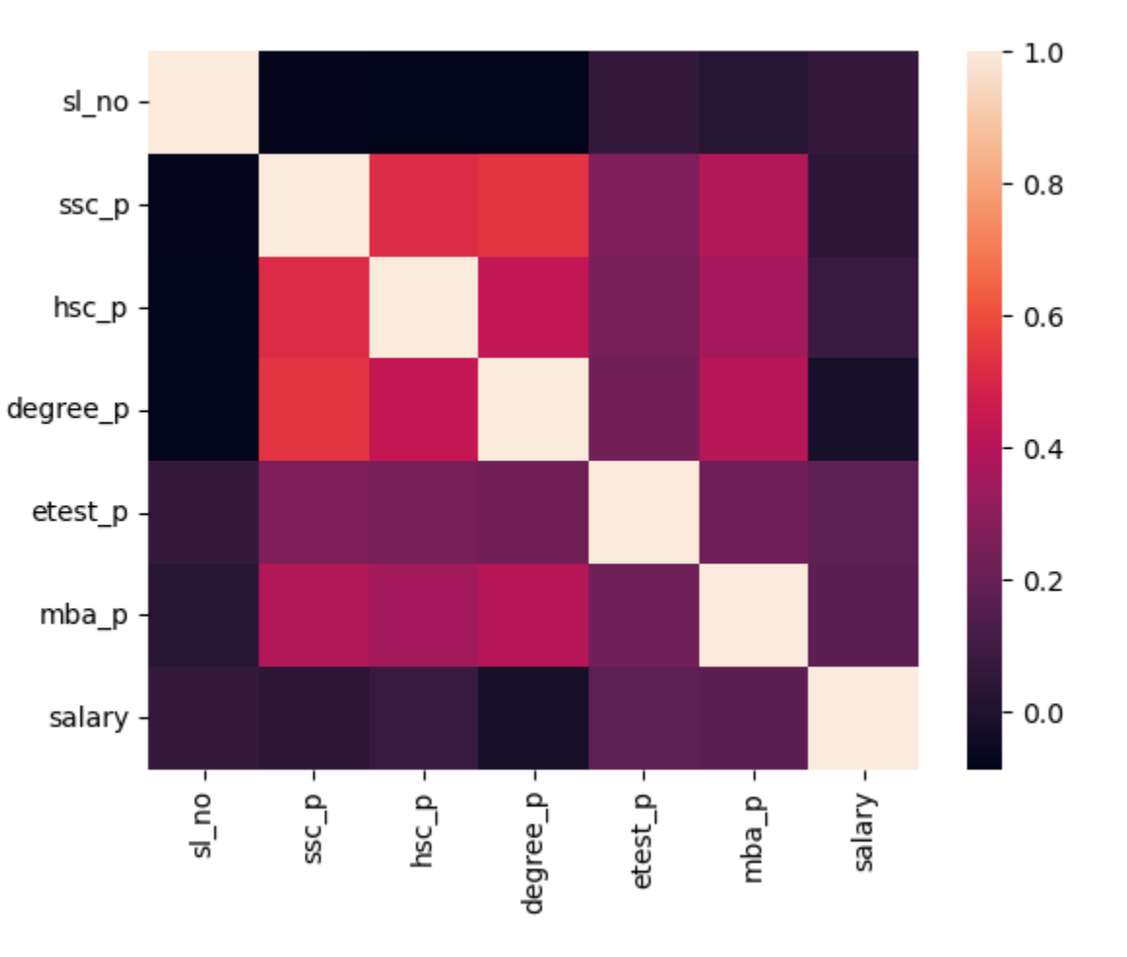
```
Out[17]: <seaborn.axisgrid.PairGrid at 0x1542ce500>
```




```python
In [18]: # showing corelation between the parameters
sns.heatmap(placement.corr())
```

```
/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/2969336950.py:2: 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 specify the value of numeric_only to silence this warning.
  sns.heatmap(placement.corr())
```

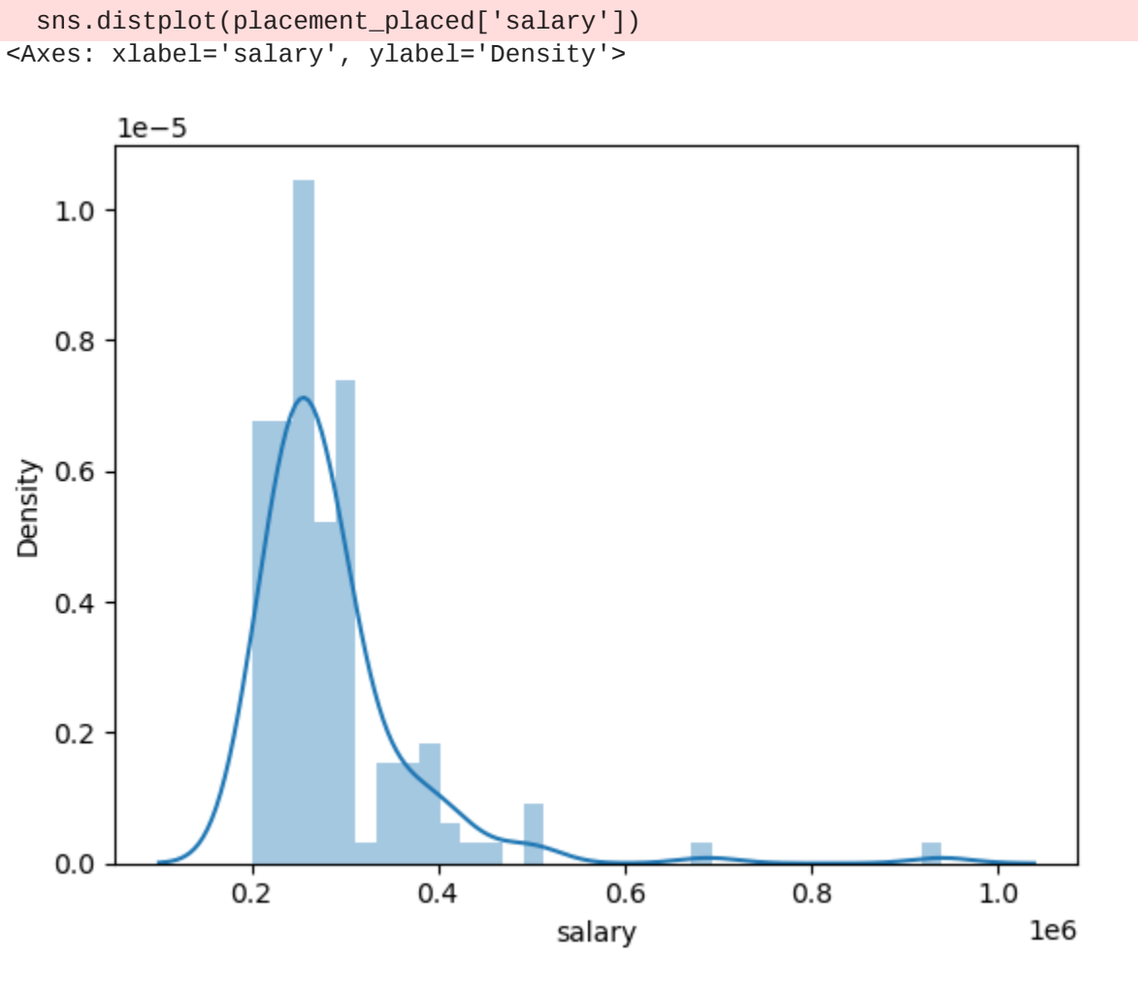
```
Out[18]: <Axes: >
```



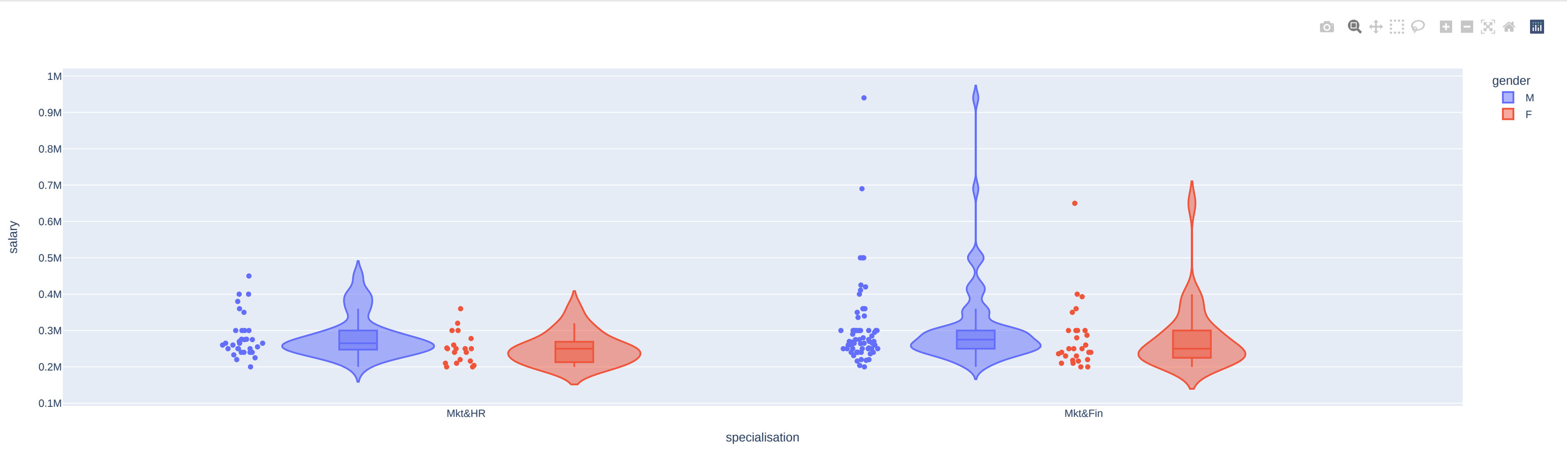

```python
In [19]: # graphical representation on the basis of salary
placement_placed = placement_filtered[placement_filtered.salary!= 0]
sns.distplot(placement_placed['salary'])
```

```
/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/3195076206.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(placement_placed['salary'])
```

```
Out[19]: <Axes: xlabel='salary', ylabel='Density'>
```




```python
In [20]: # representation on gender
import plotly.express as px
px.violin(placement_copy , y = 'salary' , x = 'specialisation' , color = 'gender' , box = True , points = 'all')
```




```python
In [21]: # executing label encoding where the data is assigned with numeric values
from sklearn.preprocessing import LabelEncoder

object_cols= ['gender','workex','specialisation','status']

label_encoder = LabelEncoder()

for col in object_cols:
    placement_filtered[col]= label_encoder.fit_transform(placement_filtered[col])

placement_filtered.head(10)
```

```
/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/1488444959.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/1488444959.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/1488444959.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/jv/slm25nb54q99gns1wg62p7840000gn/T/ipykernel_49298/1488444959.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[21]:

	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	Commerce	58.00	Sci&Tech	0	55.00	1	58.80	1	270000.0
1	1	79.33	78.33	Science	77.48	Sci&Tech	1	86.50	0	66.28	1	200000.0
2	1	65.00	68.00	Arts	64.00	Comm&Mgmt	0	75.00	0	57.80	1	250000.0
3	1	56.00	52.00	Science	52.00	Sci&Tech	0	66.00	1	59.43	0	NaN
4	1	85.80	73.60	Commerce	73.30	Comm&Mgmt	0	96.80	0	55.50	1	425000.0
5	1	55.00	49.80	Science	67.25	Sci&Tech	1	55.00	0	51.58	0	NaN
6	0	46.00	49.20	Commerce	79.00	Comm&Mgmt	0	74.28	0	53.29	0	NaN
7	1	82.00	64.00	Science	66.00	Sci&Tech	1	67.00	0	62.14	1	252000.0
8	1	73.00	79.00	Commerce	72.00	Comm&Mgmt	0	91.34	0	61.29	1	231000.0
9	1	58.00	70.00	Commerce	61.00	Comm&Mgmt	0	54.00	0	52.21	0	NaN

```python
In [22]: # One Hot Encoding(creating a number of columns using dummy variable)
dummy_hsc_s = pd.get_dummies(placement_filtered['hsc_s'], prefix = 'dummy')
dummy_degree_t = pd.get_dummies(placement_filtered['degree_t'], prefix = 'dummy')

placement_coded = pd.concat([placement_filtered , dummy_hsc_s , dummy_degree_t],axis = 1)
placement_coded.drop(['hsc_s','degree_t','salary'],axis = 1 , inplace = True)
placement_coded.head()
```

Out[22]:

	gender	ssc_p	hsc_p	degree_p	workex	etest_p	specialisation	mba_p	status	dummy_Arts	dummy_Commerce	dummy_Science	dummy_Comm&Mgmt	dummy_Others	dummy_Sci&Tech
0	1	67.00	91.00	58.00	0	55.0	1	58.80	1	0	1	0	0	0	1
1	1	79.33	78.33	77.48	1	86.5	0	66.28	1	0	0	1	0	0	1
2	1	65.00	68.00	64.00	0	75.0	0	57.80	1	1	0	0	1	0	0
3	1	56.00	52.00	52.00	0	66.0	1	59.43	0	0	0	1	0	0	1
4	1	85.80	73.60	73.30	0	96.8	0	55.50	1	0	1	0	1	0	0

```python
In [23]: #splitting the data
X = placement_coded.drop(['status'],axis=1)
y = placement_coded.status
```

```python
In [24]: # splitting it into train and test model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X , y , train_size = 0.8 , random_state = 1)
```

```python
In [25]: #using logistic regression to know the acurracy
import warnings
warnings.filterwarnings('ignore')

from sklearn.linear_model import LogisticRegression
from sklearn import metrics

logreg = LogisticRegression()

logreg.fit(X_train , y_train)

y_pred = logreg.predict(X_test)

print(logreg.score(X_test , y_test))
```

```
0.8333333333333334
```

```python
In [26]: # using decision tree classifier to print the accuracy of the placement
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion = 'gini' , max_depth = 3)

dt = dt.fit(X_train , y_train)
y_pred = dt.predict(X_test)

print("Accuracy", metrics.accuracy_score(y_test , y_pred))
```

```
Accuracy 0.7380952380952381
```

```python
In [27]: #using random forest classsifier to print the classifier
from sklearn.ensemble import RandomForestClassifier

rt = RandomForestClassifier(n_estimators = 100)

rt.fit(X_train , y_train)
y_pred = rt.predict(X_test)

print("Accuracy", metrics.accuracy_score(y_test , y_pred))
```

```
Accuracy 0.7857142857142857
```

```python
In [28]: # here the placement prediction is done by showing all the model graphs, bar graphs , and also by using the regressions
```

```python
In [ ]:
```