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

%matplotlib inline
```

```
data=pd.read_csv('/content/fresherssalarypredection.csv')
data.head()
```

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

```
data.shape
```

```
(215, 15)
```

```
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
```

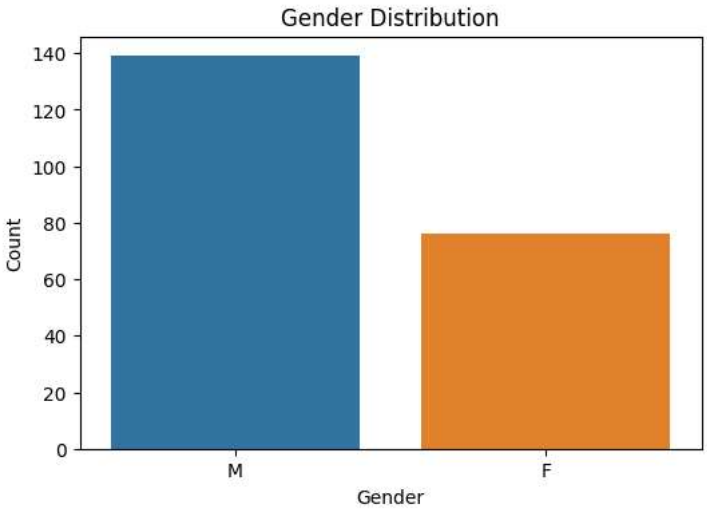
```
data.describe()
```

	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

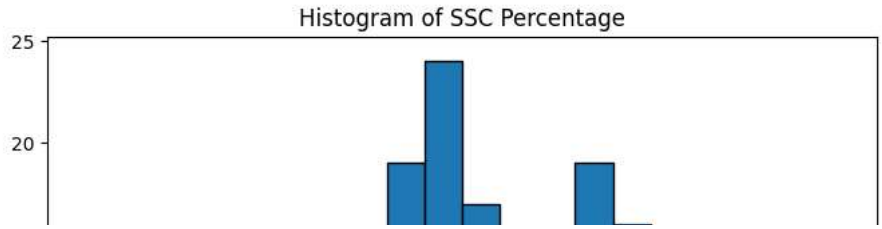
```
data.isnull().sum()
```

```
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
```

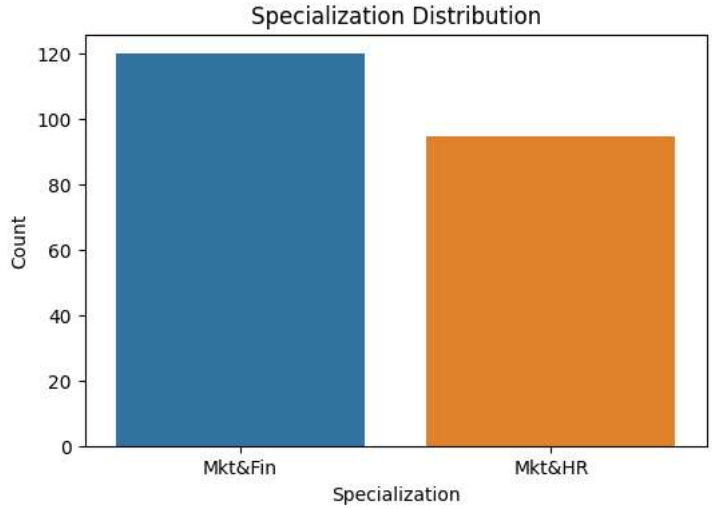
```
df = pd.DataFrame(data)

gender_count = df['gender'].value_counts()
plt.figure(figsize=(6, 4))
sns.barplot(x=gender_count.index, y=gender_count.values)
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```




```
plt.figure(figsize=(8, 5))
plt.hist(df['ssc_p'], bins=20, edgecolor='black')
plt.title('Histogram of SSC Percentage')
plt.xlabel('SSC Percentage')
plt.ylabel('Frequency')
plt.show()
```

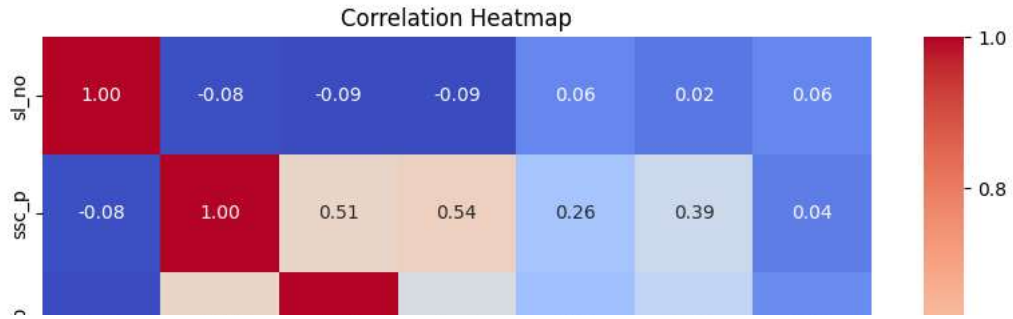



```
specialization_count = df['specialisation'].value_counts()
plt.figure(figsize=(6, 4))
sns.barplot(x=specialization_count.index, y=specialization_count.values)
plt.title('Specialization Distribution')
plt.xlabel('Specialization')
plt.ylabel('Count')
plt.show()
```



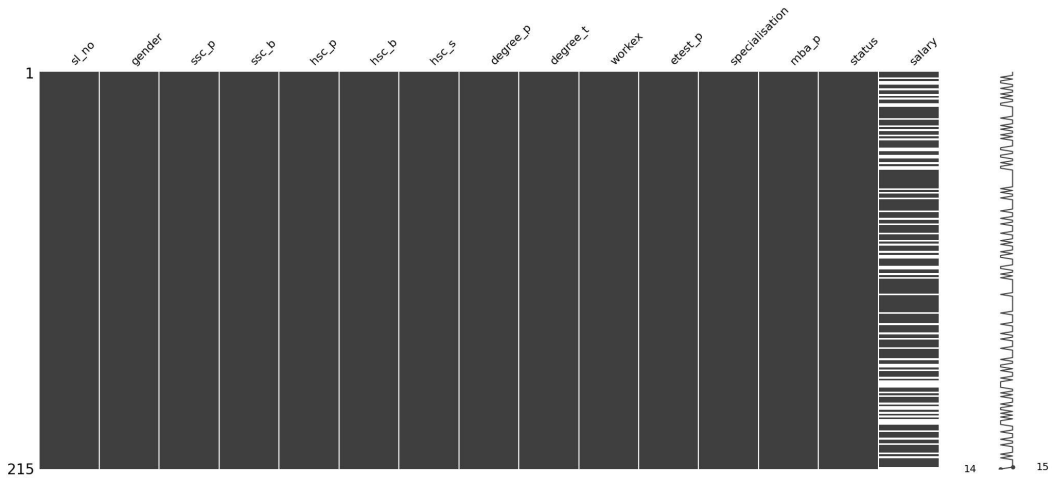

```
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-33-d7cec65cd797>:2: FutureWarning: The default value of numeric_only in DataFrame.corr :
  correlation_matrix = df.corr()
```




```
ms.matrix(data)
```

```
<Axes: >
```




```
data['salary'].fillna(data['salary'].mean(), inplace=True)
```

```
ms.matrix(data)
```

<Axes: >

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
categorical_columns = ['gender', 'ssc_b', 'hsc_b', 'hsc_s', 'degree_t', 'workex', 'specialisation','status']
data_encoded = pd.get_dummies(data, columns=categorical_columns, drop_first=True)
data_encoded['total_percentage'] = data_encoded['ssc_p'] + data_encoded['hsc_p'] + data_encoded['degree_p'] + data_encoded['etest_p'] + data_
data_encoded.head()
```

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary	gender_M	ssc_b_Others	hsc_b_Others	h
0	1	67.00	91.00	58.00	55.0	58.80	270000.000000	1	1	1	
1	2	79.33	78.33	77.48	86.5	66.28	200000.000000	1	0	1	
2	3	65.00	68.00	64.00	75.0	57.80	250000.000000	1	0	0	
3	4	56.00	52.00	52.00	66.0	59.43	288655.405405	1	0	0	
4	5	85.80	73.60	73.30	96.8	55.50	425000.000000	1	0	0	

```
X = data_encoded.drop(columns=['status_Placed'])
y = data_encoded['status_Placed']
```

```
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=42)
```

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

```
Accuracy: 0.9069767441860465
Precision: 0.9090909090909091
Recall: 0.967741935483871
F1-score: 0.9374999999999999
```

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
}

grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')

grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
```

```
y_pred_best = best_model.predict(X_test)
best_accuracy = accuracy_score(y_test, y_pred_best)
best_precision = precision_score(y_test, y_pred_best)
best_recall = recall_score(y_test, y_pred_best)
best_f1 = f1_score(y_test, y_pred_best)

print("Best Model:")
print("Accuracy:", best_accuracy)
print("Precision:", best_precision)
print("Recall:", best_recall)
print("F1-score:", best_f1)
```

```
Best Model:
Accuracy: 0.9302325581395349
Precision: 0.9375
Recall: 0.967741935483871
F1-score: 0.9523809523809523
```

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

xgb_classifier = XGBClassifier(random_state=42)
xgb_classifier.fit(X_train_scaled, y_train)

y_pred = xgb_classifier.predict(X_test_scaled)


accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("ROC-AUC Score:", roc_auc)
```

```
Accuracy: 0.9767441860465116
Precision: 1.0
Recall: 0.967741935483871
F1 Score: 0.9836065573770492
ROC-AUC Score: 0.9838709677419355
```