In [1]:

```
import pandas as pd
import numpy as np
file_path= r"C:\Users\niruw\OneDrive\Desktop\Project\Placement_Data_Full_Class.csv"
df=pd.read_csv(file_path)
df.head()
```

Out[1]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	work
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	N
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Y
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	N
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	N
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	N

In [2]:

```
#data exploration
# Check basic structure
print(df.shape)
print(df.columns)
print(df.info())
print(df.describe())
print(df.head())
```

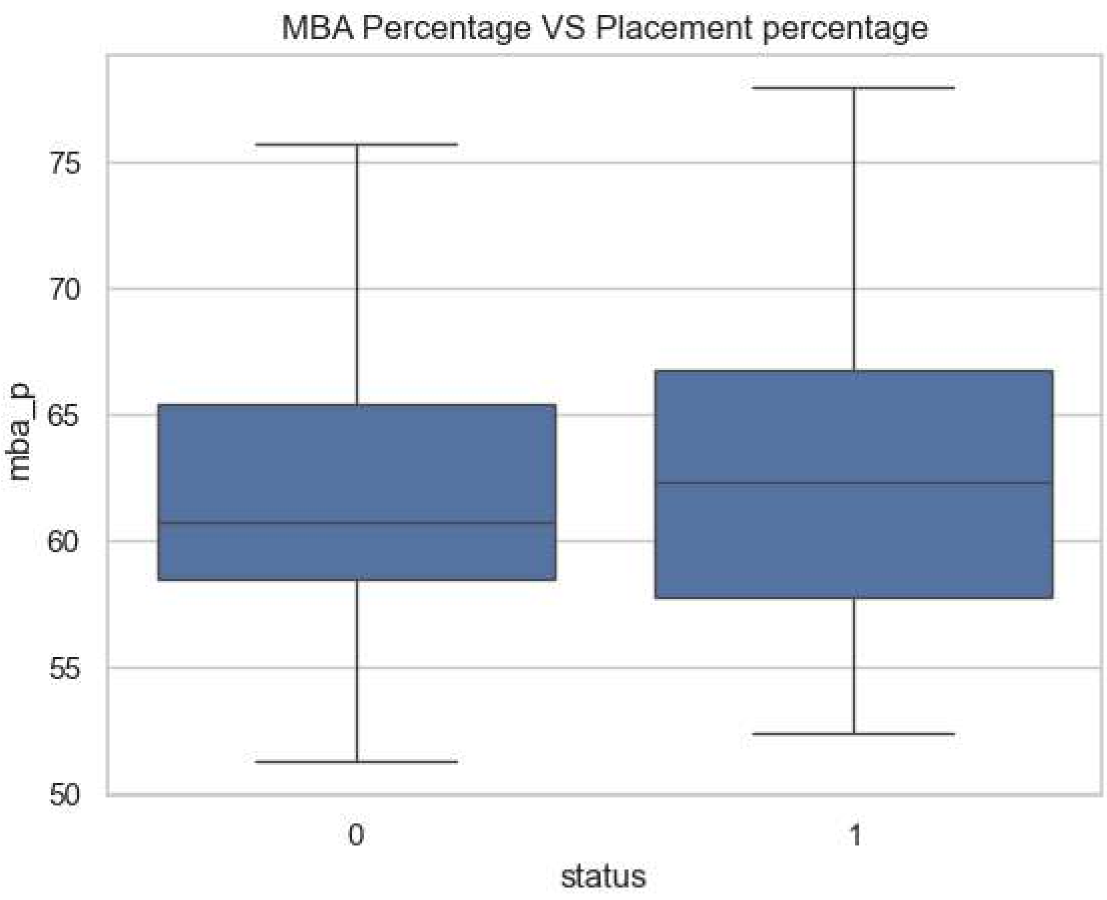
```
(215, 15)
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 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
None
            sl_no       ssc_p       hsc_p    degree_p     etest_p       mba_p  \
count  215.000000  215.000000  215.000000  215.000000  215.000000  215.000000   
mean   108.000000   67.303395   66.333163   66.370186   72.100558   62.278186   
std     62.209324   10.827205   10.897509    7.358743   13.275956    5.833385   
min      1.000000   40.890000   37.000000   50.000000   50.000000   51.210000   
25%     54.500000   60.600000   60.900000   61.000000   60.000000   57.945000   
50%    108.000000   67.000000   65.000000   66.000000   71.000000   62.000000   
75%    161.500000   75.700000   73.000000   72.000000   83.500000   66.255000   
max    215.000000   89.400000   97.700000   91.000000   98.000000   77.890000   

              salary  
count     148.000000  
mean   288655.405405  
std     93457.452420  
min    200000.000000  
25%    240000.000000  
50%    265000.000000  
75%    300000.000000  
max    940000.000000  
   sl_no gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p  \
0      1      M  67.00   Others  91.00   Others  Commerce     58.00   
1      2      M  79.33  Central  78.33   Others   Science     77.48   
2      3      M  65.00  Central  68.00  Central      Arts     64.00   
3      4      M  56.00  Central  52.00  Central   Science     52.00   
4      5      M  85.80  Central  73.60  Central  Commerce     73.30   

    degree_t workex  etest_p specialisation  mba_p      status    salary  
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  270000.0  
```

```
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed       NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed  425000.0
```

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

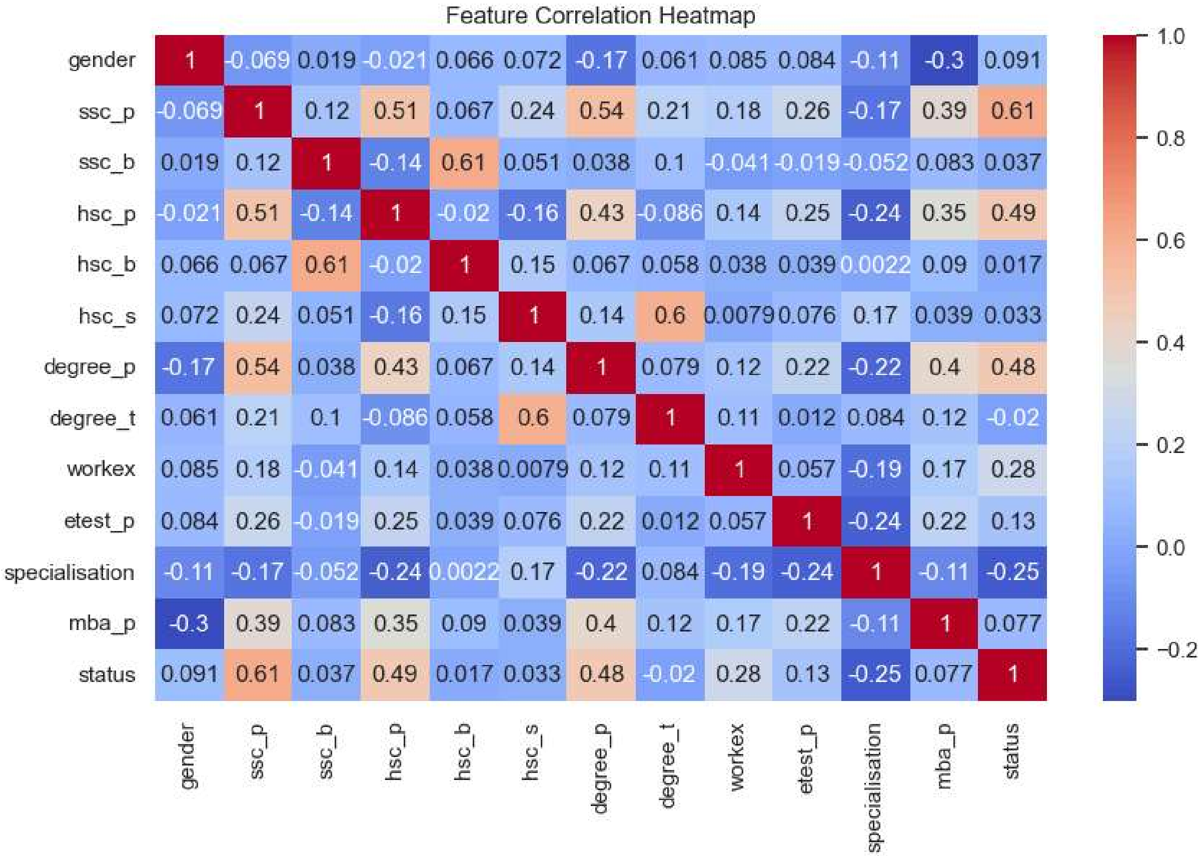
In [9]:
```
#Data Preprocessing
df.columns = df.columns.str.strip()
df.drop(columns=['sl_no', 'salary'], inplace=True, errors='ignore')

from sklearn.preprocessing import LabelEncoder
for column in df.select_dtypes(include='object').columns:
    df[column]=LabelEncoder().fit_transform(df[column])
```

In [10]:
```
#Bar Plot
sns.boxplot(x='status',y='mba_p',data=df)
plt.title('MBA Percentage VS Placement percentage')
plt.show()
```



In [11]:
```
#Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
```

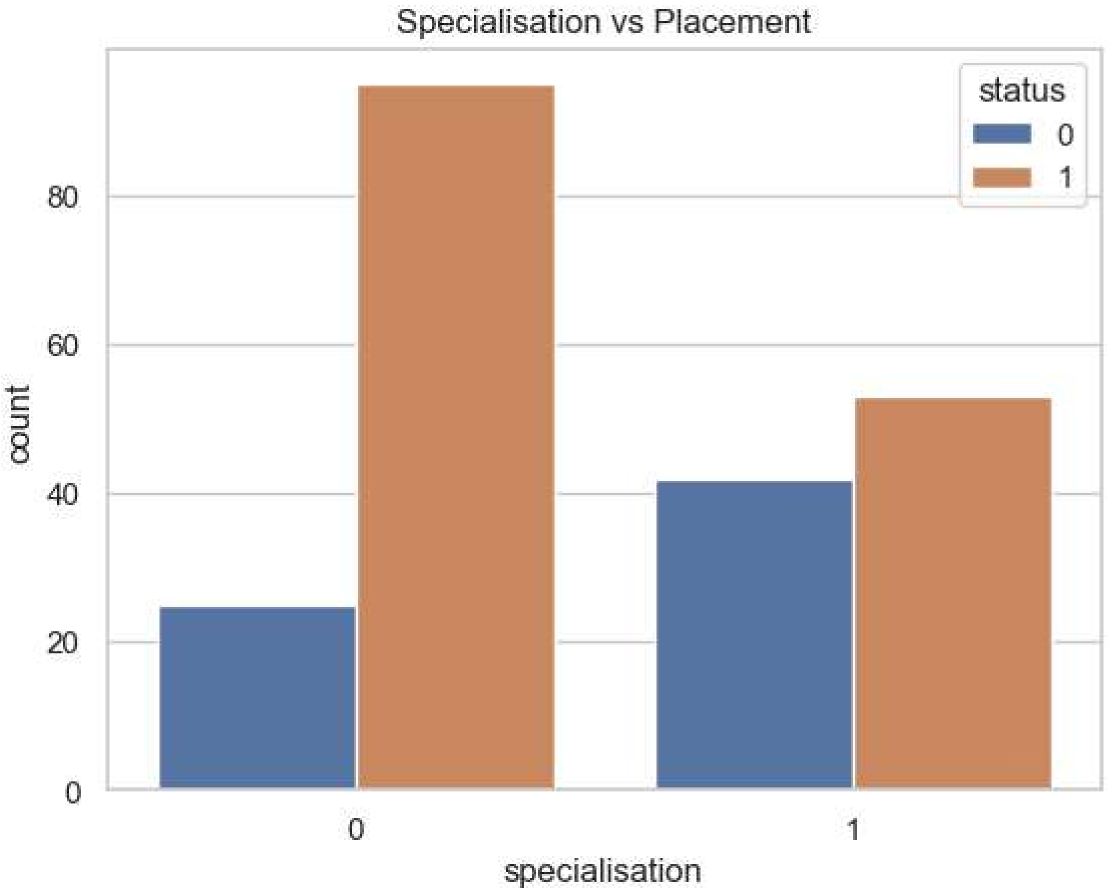
```
plt.title('Feature Correlation Heatmap')
plt.show()
```

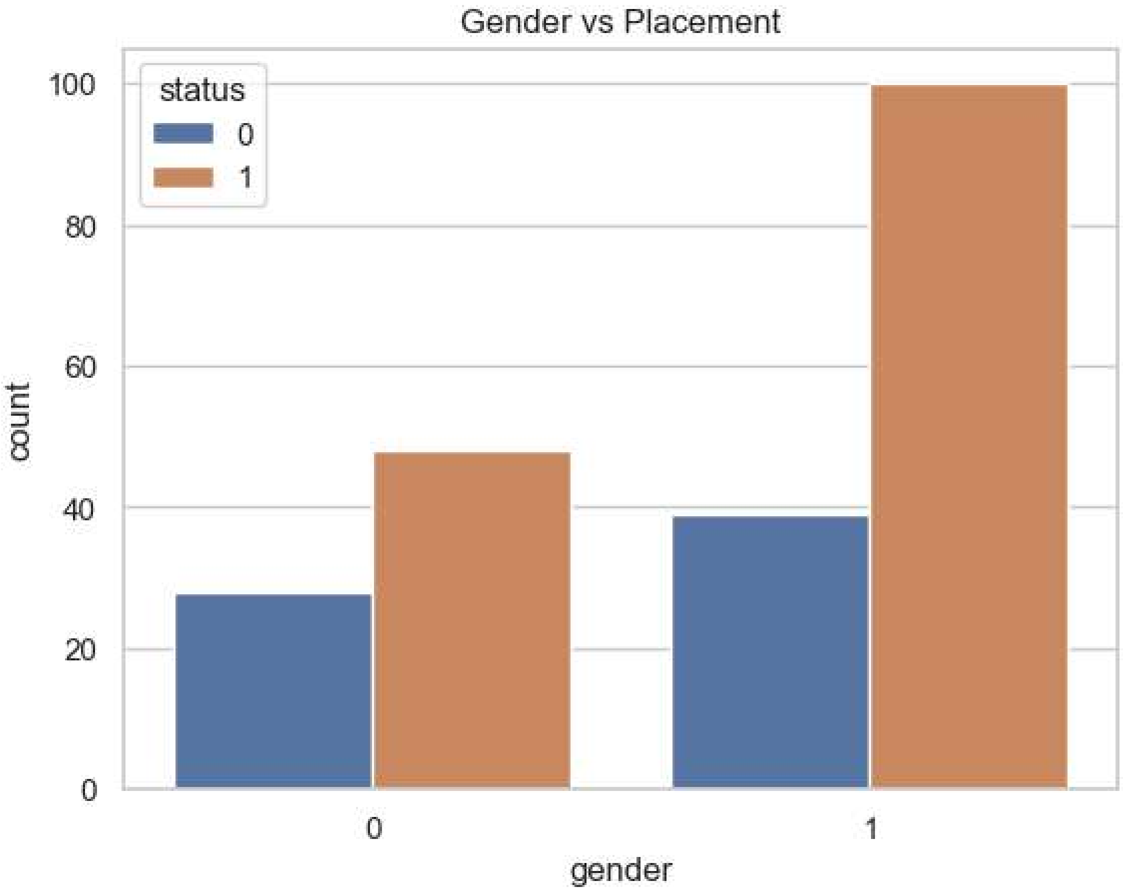


	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
gender	1	-0.069	0.019	-0.021	0.066	0.072	-0.17	0.061	0.085	0.084	-0.11	-0.3	0.091
ssc_p	-0.069	1	0.12	0.51	0.067	0.24	0.54	0.21	0.18	0.26	-0.17	0.39	0.61
ssc_b	0.019	0.12	1	-0.14	0.61	0.051	0.038	0.1	-0.041	-0.019	-0.052	0.083	0.037
hsc_p	-0.021	0.51	-0.14	1	-0.02	-0.16	0.43	-0.086	0.14	0.25	-0.24	0.35	0.49
hsc_b	0.066	0.067	0.61	-0.02	1	0.15	0.067	0.058	0.038	0.039	0.0022	0.09	0.017
hsc_s	0.072	0.24	0.051	-0.16	0.15	1	0.14	0.6	0.0079	0.076	0.17	0.039	0.033
degree_p	-0.17	0.54	0.038	0.43	0.067	0.14	1	0.079	0.12	0.22	-0.22	0.4	0.48
degree_t	0.061	0.21	0.1	-0.086	0.058	0.6	0.079	1	0.11	0.012	0.084	0.12	-0.02
workex	0.085	0.18	-0.041	0.14	0.038	0.0079	0.12	0.11	1	0.057	-0.19	0.17	0.28
etest_p	0.084	0.26	-0.019	0.25	0.039	0.076	0.22	0.012	0.057	1	-0.24	0.22	0.13
specialisation	-0.11	-0.17	-0.052	-0.24	0.0022	0.17	-0.22	0.084	-0.19	-0.24	1	-0.11	-0.25
mba_p	-0.3	0.39	0.083	0.35	0.09	0.039	0.4	0.12	0.17	0.22	-0.11	1	0.077
status	0.091	0.61	0.037	0.49	0.017	0.033	0.48	-0.02	0.28	0.13	-0.25	0.077	1

In [12]:

```
#Placement by specialisation
sns.countplot(x='specialisation',hue='status',data=df)
plt.title('Specialisation vs Placement')
plt.show()

sns.countplot(x='gender',hue='status',data=df)
plt.title('Gender vs Placement')
plt.show()
```


Specialisation vs Placement
status
0
1
count
80
60
40
20
0
0
1
specialisation


Gender vs Placement
status
0
1
count
100
80
60
40
20
0
0
1
gender

In [18]:

```
df.to_csv("cleaned_data.csv", index=False)
```

In [5]:

```
#Preprocess before modelling
import pandas as pd
df = pd.read_csv("cleaned_data.csv")
from sklearn.model_selection import train_test_split
X = df.drop("status", axis=1)   # Features
y = df["status"]                # Target (0 = Not Placed, 1 = Placed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [6]:

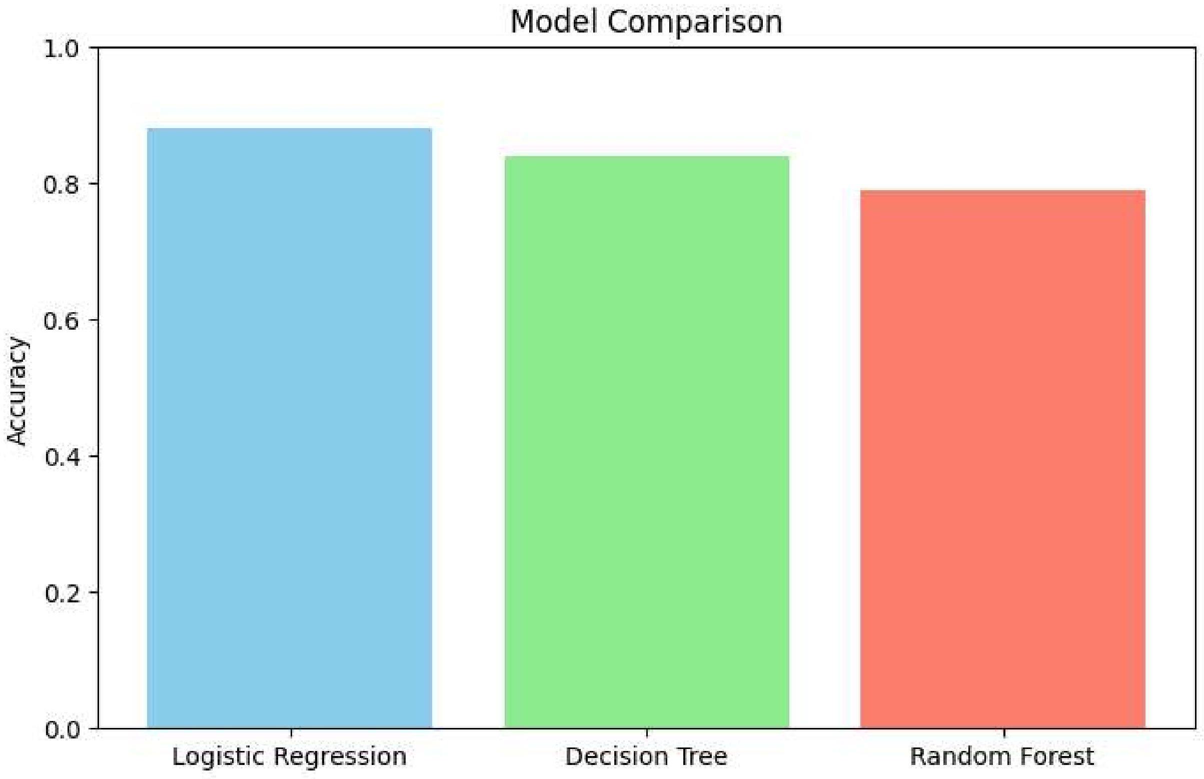
```
#Training multiple ML models

#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
logreg = LogisticRegression(max_iter=500, solver='lbfgs')
logreg.fit(X_train_scaled, y_train)
y_pred = logreg.predict(X_test_scaled)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))


#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

#Random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
Logistic Regression Accuracy: 0.8837209302325582
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        12
           1       0.91      0.94      0.92        31

    accuracy                           0.88        43
   macro avg       0.86      0.84      0.85        43
weighted avg       0.88      0.88      0.88        43

Decision Tree Accuracy: 0.8372093023255814
Random Forest Accuracy: 0.7906976744186046
```

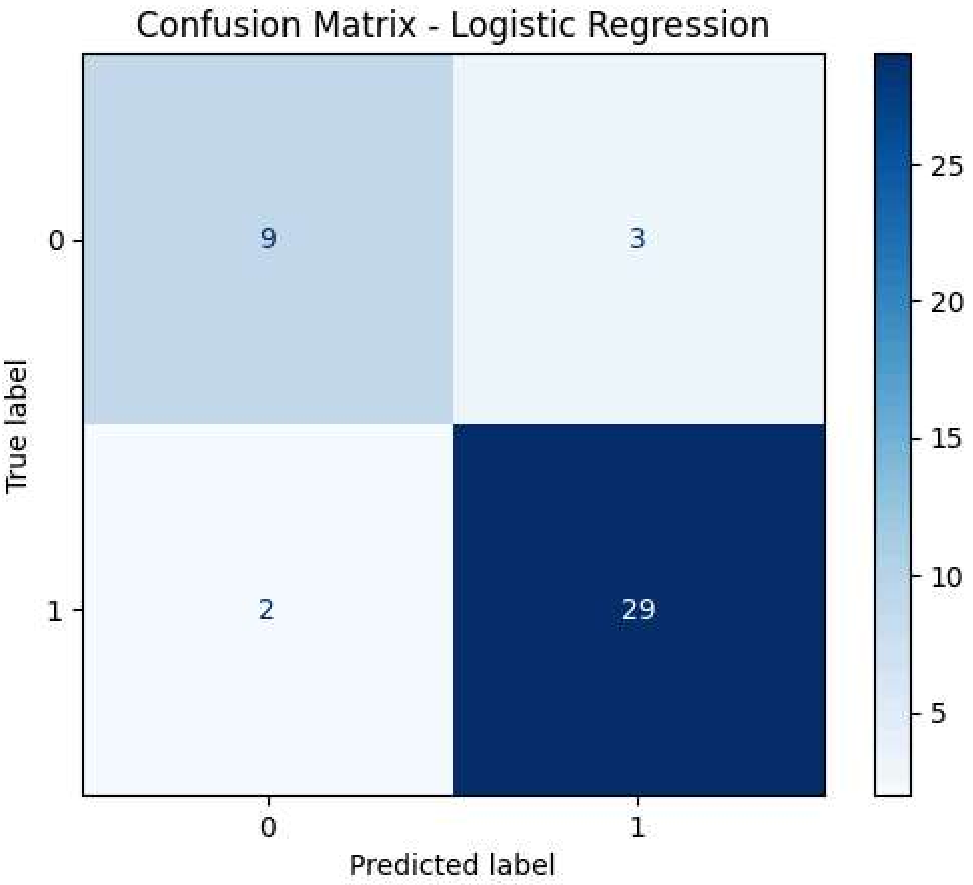
In [7]:

```
import matplotlib.pyplot as plt
```

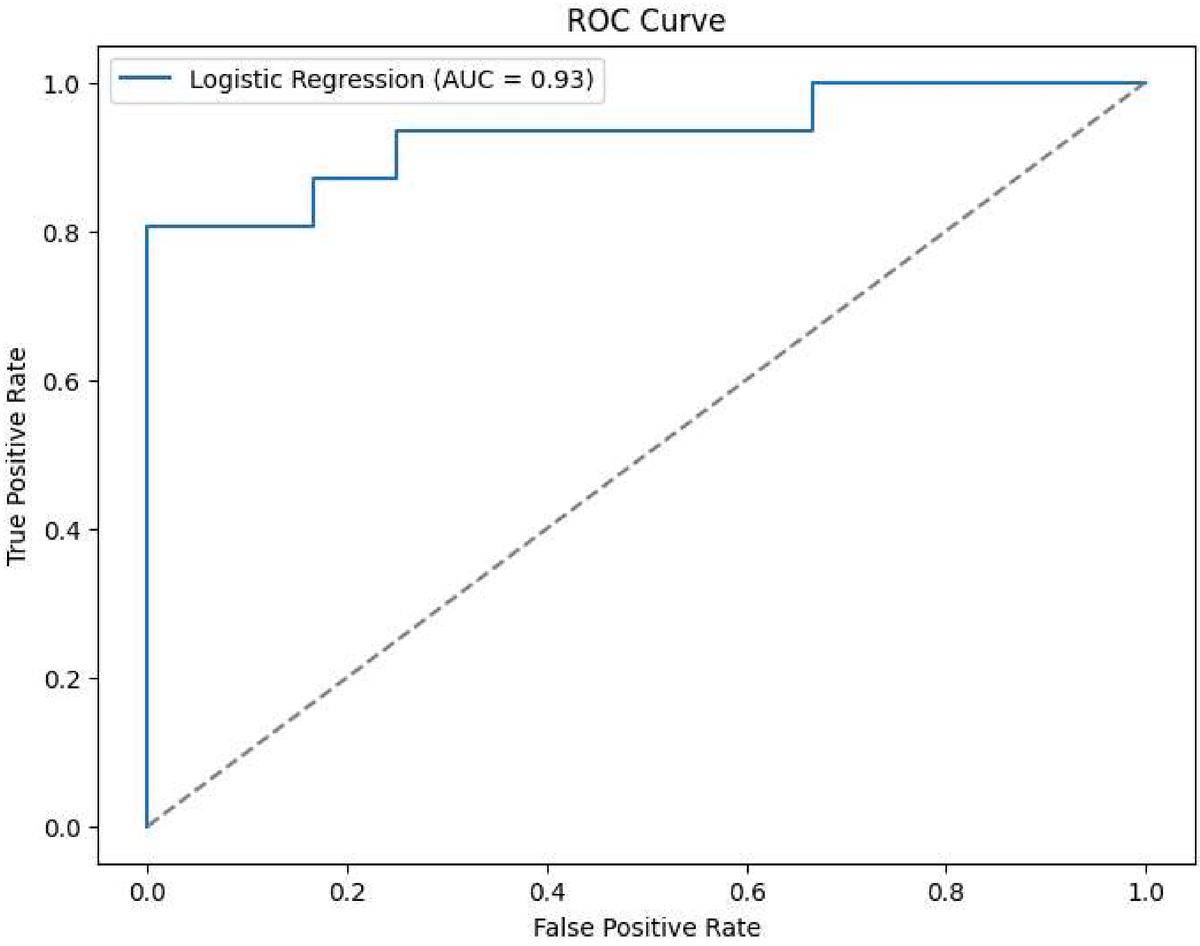
```
accuracies = {
    'Logistic Regression': 0.88,
    'Decision Tree': 0.84,
    'Random Forest': 0.79
}

# Bar chart
plt.figure(figsize=(8,5))
plt.bar(accuracies.keys(), accuracies.values(), color=['skyblue', 'lightgreen', 'sa
plt.ylabel('Accuracy')
plt.title('Model Comparison')
plt.ylim(0, 1)
plt.show()
```



In [8]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```



In [9]:
```
from sklearn.metrics import roc_curve, auc

y_prob = logreg.predict_proba(X_test_scaled)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0,1], [0,1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```


ROC Curve
Logistic Regression (AUC = 0.93)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

```
In [ ]:
```