In [6]:
```
#Step 1: Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, confusion_matrix, ConfusionMatrixDisplay,
    roc_curve, roc_auc_score
)
```

In [12]:
```
#Step 2: Load Titanic dataset



df = sns.load_dataset('titanic')
df.head()
```

Out[12]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [14]:
```
#Step 3: Select useful columns
df = df[['survived', 'pclass', 'sex', 'age', 'fare', 'embarked']]
```

In [18]:
```
#Step 4: Handle missing values
df['age'] = df['age'].fillna(df['age'].median())
df['embarked'] = df['embarked'].fillna(df['embarked'].mode()[0])
```

In [20]:
```
#Step 5: Encode categorical variables
df = pd.get_dummies(df, drop_first=True)
```

In [22]:
```
#Step 6: Split features & target
X = df.drop('survived', axis=1)
y = df['survived']
```

In [24]:
```
#Step 7: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [26]:
```
#Step 8: Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [28]:
```
#Step 9: Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[28]:

▾ LogisticRegression

```
LogisticRegression()
```

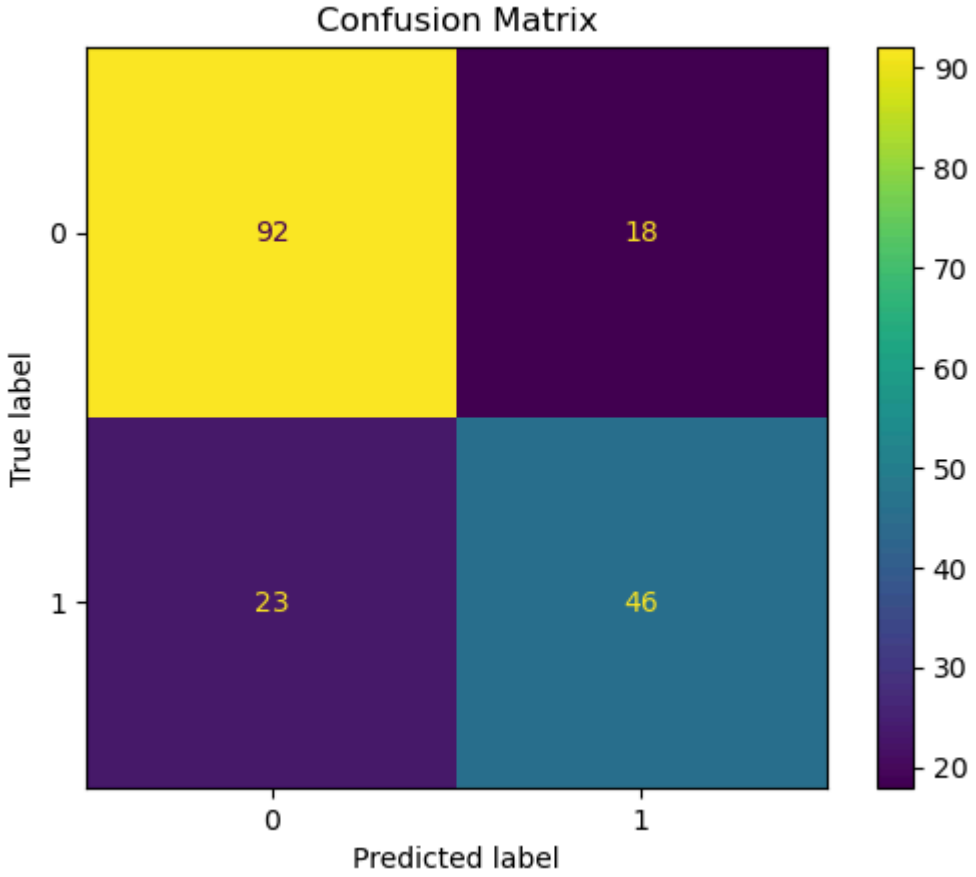
In [30]:
```
#Step 10: Predictions
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]
```

In [32]:
```
#Step 11: Evaluation metrics
print("Accuracy :", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
```

```
print("Recall   :", recall_score(y_test, y_pred))
print("F1-score :", f1_score(y_test, y_pred))
```

```
Accuracy : 0.770949720670391
Precision: 0.71875
Recall   : 0.6666666666666666
F1-score : 0.6917293233082706
```

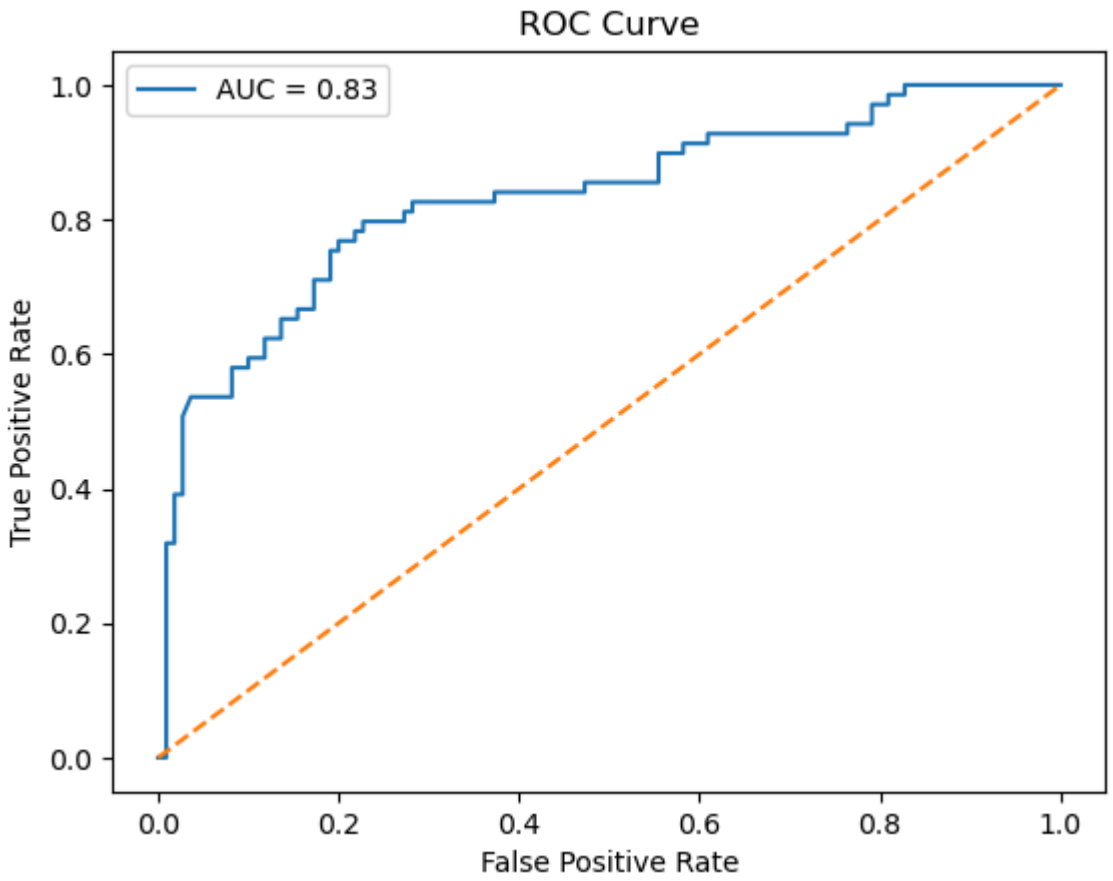
In [34]:
```
#Step 12: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title("Confusion Matrix")
plt.show()
```



In [36]:
```
#Step 13: ROC Curve & AUC (SAVE THIS IMAGE)
fpr, tpr, _ = roc_curve(y_test, y_prob)
auc_score = roc_auc_score(y_test, y_prob)

plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

```
print("AUC Score:", auc_score)
```




```
AUC Score: 0.8325428194993412
```

In [ ]: