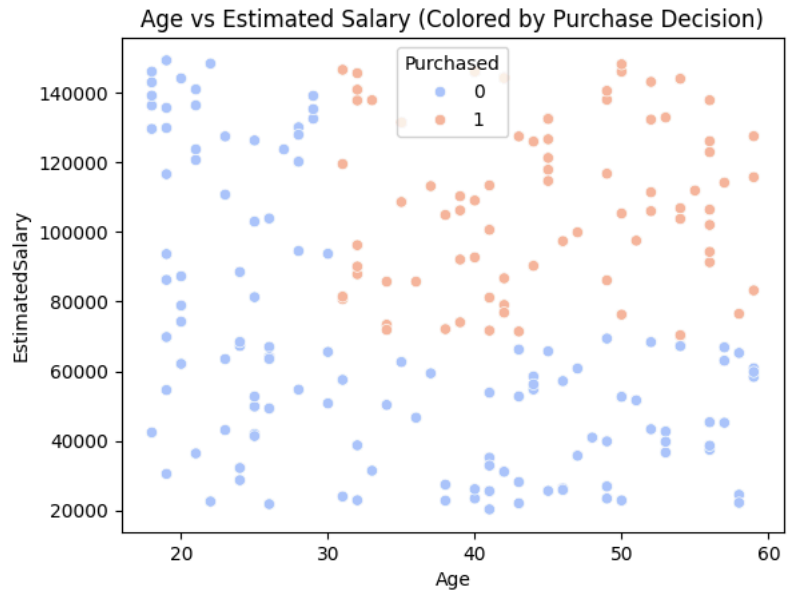
```
#STEP 1: Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
```

```
# STEP 2: Create or load dataset
# Generating synthetic data for demonstration
np.random.seed(42)
age = np.random.randint(18, 60, 200)
salary = np.random.randint(20000, 150000, 200)
purchased = (age > 30) & (salary > 70000)
purchased = purchased.astype(int)

df = pd.DataFrame({
    'Age': age,
    'EstimatedSalary': salary,
    'Purchased': purchased
})

print("✅ Dataset Created Successfully!")
print(df.head())
```

```
✅ Dataset Created Successfully!
   Age  EstimatedSalary  Purchased
0   56            45342          0
1   46            57157          0
2   32            87863          1
3   25           126308          0
4   38            72083          1
```

```
# STEP 3: Visualize data distribution
sns.scatterplot(x='Age', y='EstimatedSalary', hue='Purchased', data=df, palette='coolwarm')
plt.title("Age vs Estimated Salary (Colored by Purchase Decision)")
plt.show()
```



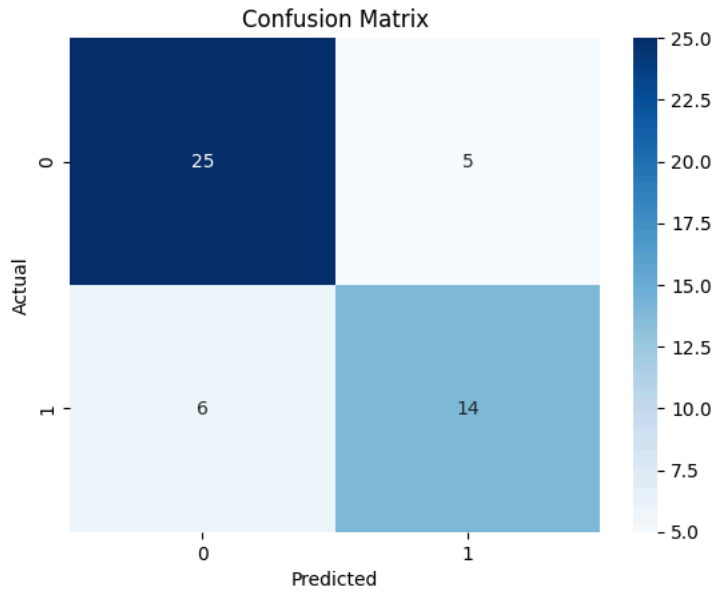

```
# STEP 4: Split dataset into training and testing sets
X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# STEP 5: Feature scaling (important for logistic regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# STEP 6: Train Logistic Regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

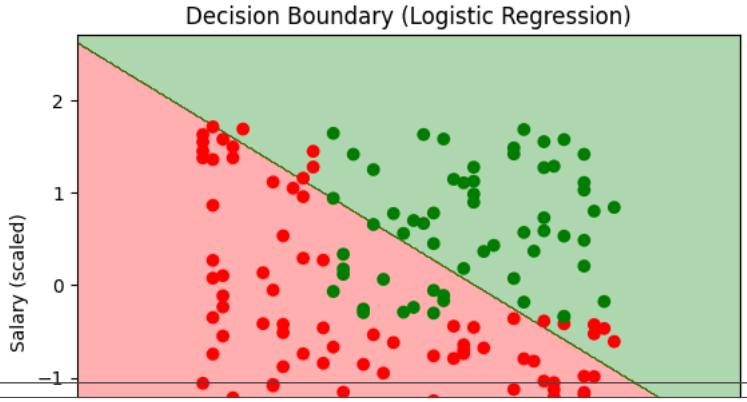
```
print("\n✅ Model Trained Successfully!")

# STEP 7: Predictions and evaluation
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:, 1]

print("\n📋 Classification Report:")
print(classification_report(y_test, y_pred))
```

```
✅ Model Trained Successfully!

📋 Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.83      0.82        30
           1       0.74      0.70      0.72        20

    accuracy                           0.78        50
   macro avg       0.77      0.77      0.77        50
weighted avg       0.78      0.78      0.78        50
```

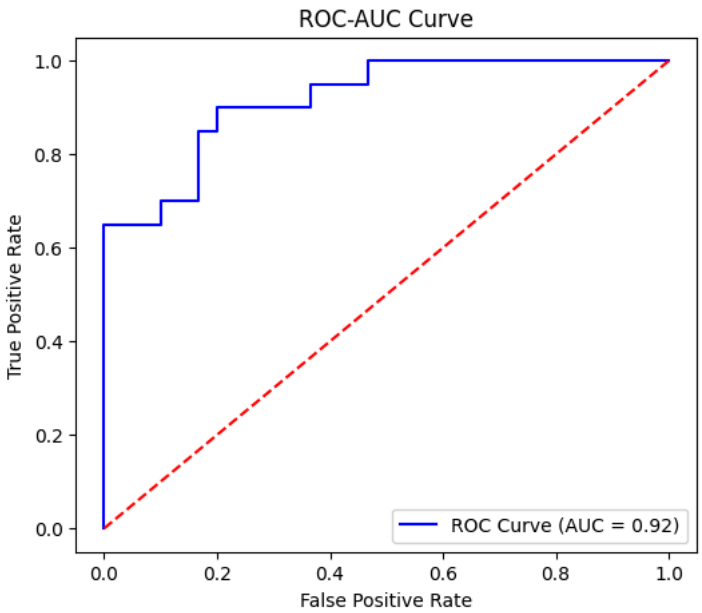
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```




```
# STEP 8: Plot decision boundary
from matplotlib.colors import ListedColormap

X_set, y_set = X_train_scaled, y_train
X1, X2 = np.meshgrid(
    np.arange(X_set[:, 0].min() - 1, X_set[:, 0].max() + 1, 0.01),
    np.arange(X_set[:, 1].min() - 1, X_set[:, 1].max() + 1, 0.01)
)
Z = model.predict(np.array([X1.ravel(), X2.ravel()]).T)
Z = Z.reshape(X1.shape)

plt.contourf(X1, X2, Z, alpha=0.3, cmap=ListedColormap(('red', 'green')))
plt.scatter(X_set[:, 0], X_set[:, 1], c=y_set, cmap=ListedColormap(('red', 'green')))
plt.title("Decision Boundary (Logistic Regression)")
plt.xlabel("Age (scaled)")
plt.ylabel("Salary (scaled)")
plt.show()
```




```
# STEP 9: ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.title('ROC-AUC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

print(f"✅ ROC-AUC Score: {auc:.3f}")
```




```
✅ ROC-AUC Score: 0.918
```