In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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, accuracy_score, precision_score, r
```

In [2]:

```
# Load the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
print(dataset.head())
```

```
   Age  EstimatedSalary  Purchased
0   19            19000          0
1   35            20000          0
2   26            43000          0
3   27            57000          0
4   19            76000          0
```

In [3]:

```
# Select features and target
X = dataset[['Age', 'EstimatedSalary']].values
y = dataset['Purchased'].values

# Split into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [4]:

```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[4]:

```
▾ LogisticRegression
LogisticRegression(random_state=0)
```

In [5]:

```
y_pred = classifier.predict(X_test)
```

In [6]:

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Extracting TP, TN, FP, FN
TN, FP, FN, TP = cm.ravel()
print(f"TP = {TP}, FP = {FP}, TN = {TN}, FN = {FN}")

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Error Rate
error_rate = 1 - accuracy
print(f"Error Rate: {error_rate:.2f}")
```

```
# Precision
precision = precision_score(y_test, y_pred)
print(f"Precision: {precision:.2f}")

# Recall
recall = recall_score(y_test, y_pred)
print(f"Recall: {recall:.2f}")
```

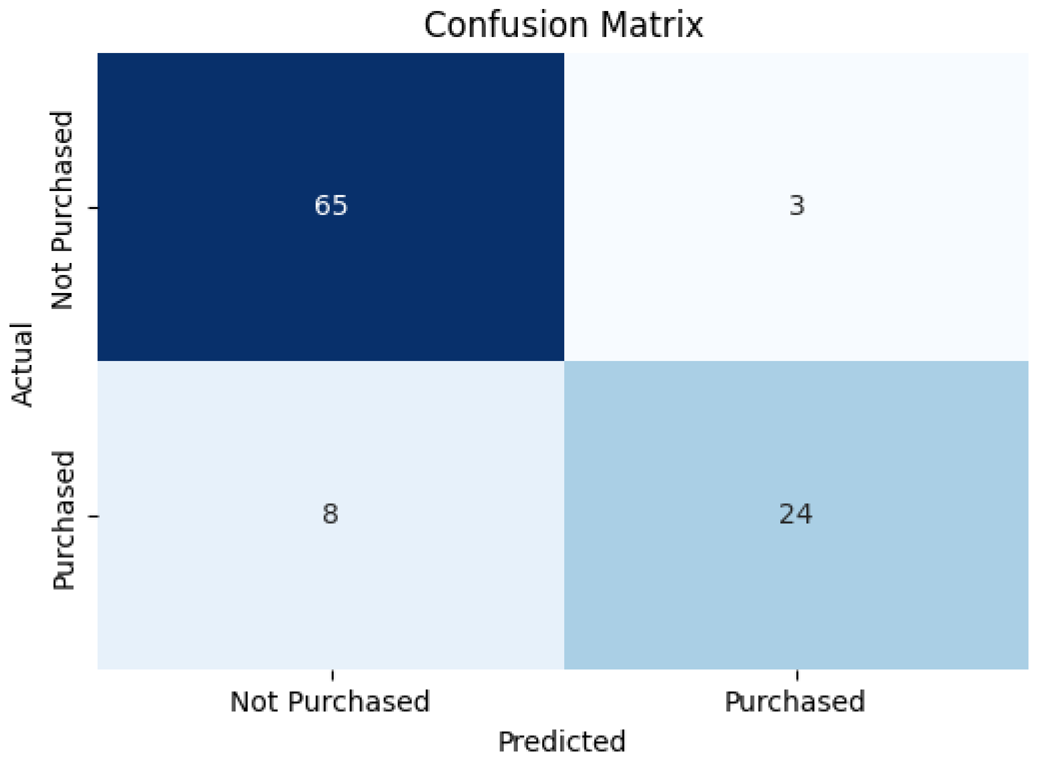
```
Confusion Matrix:
 [[65  3]
 [ 8 24]]
TP = 24, FP = 3, TN = 65, FN = 8
Accuracy: 0.89
Error Rate: 0.11
Precision: 0.89
Recall: 0.75
```

In [8]:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Compute confusion matrix again (just to be sure)
cm = confusion_matrix(y_test, y_pred)

# Draw the confusion matrix using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Not Purchased', 'Purchased'],
            yticklabels=['Not Purchased', 'Purchased'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


Confusion Matrix
Actual
Not Purchased
Purchased
Not Purchased
Purchased
Predicted