In [3]:
```
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.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, classification_report, accuracy_score, r2_s
```

In [4]:
```
df = pd.read_csv('Social_Network_Ads.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [6]:
```
df.shape
```

Out[6]: (400, 5)

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [11]:
```
print("\n=== Statistical Information ===")
df.describe()
```

```
=== Statistical Information ===
```

Out[11]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

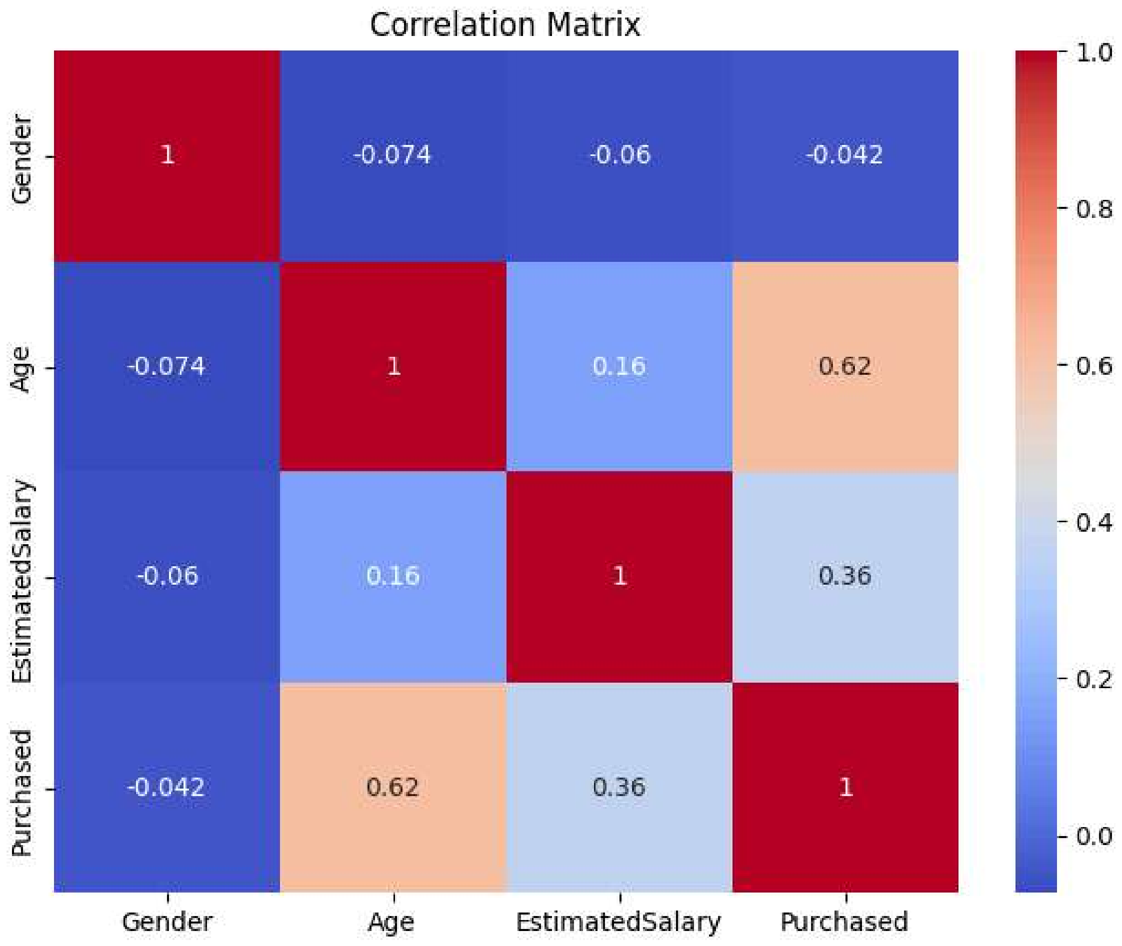
In [10]:
```
df.isnull().sum()
```

Out[10]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [12]:
```
# Drop User ID as it's not relevant
df = df.drop('User ID', axis=1)
```

In [13]:
```
# Convert Gender to numerical values (0 for Female, 1 for Male)
df['Gender'] = df['Gender'].map({'Female': 0, 'Male': 1})
```

In [14]:
```
# Display correlation matrix
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [15]:
```
# Prepare features and target
X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']

# Split data into training and testing sets (75% train, 25% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [16]:
```
# Build and train logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[16]:

```
▾ LogisticRegression
LogisticRegression()
```

In [17]:

```
# Make predictions
y_pred = logreg.predict(X_test)
y_prob = logreg.predict_proba(X_test)[:, 1]
```

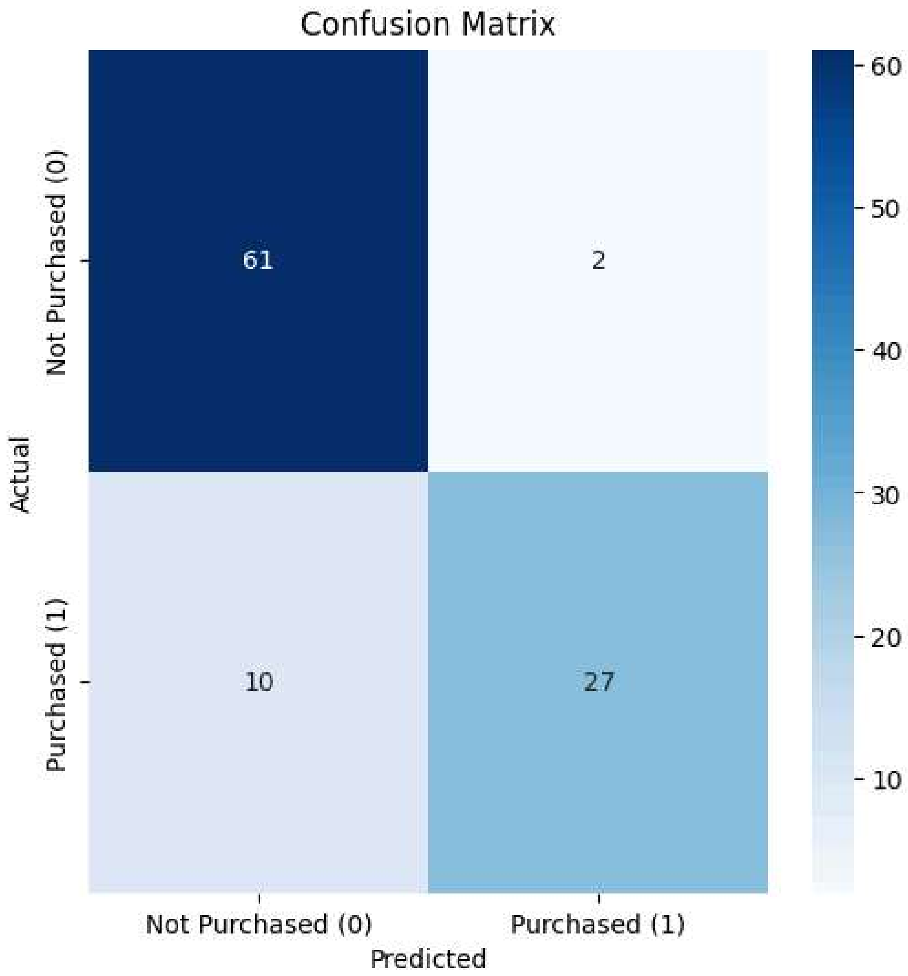
In [18]:

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

```
=== Confusion Matrix ===
[[61  2]
 [10 27]]
```

In [19]:

```
# Visualize confusion matrix
plt.figure(figsize=(6,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Purchased (0)', 'Purchased (1)'],
            yticklabels=['Not Purchased (0)', 'Purchased (1)'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [22]:

```
# Calculate metrics
tn, fp, fn, tp = cm.ravel()
accuracy = (tp + tn) / (tp + tn + fp + fn)
error_rate = 1 - accuracy
precision = tp / (tp + fp)
```

```
recall = tp / (tp + fn)
r2 = r2_score(y_test, y_prob)
```

In [23]:
```
# Display classification report and metrics
print("\n=== Classification Report ===")
print(classification_report(y_test, y_pred))
print("\n=== Performance Metrics ===")
print(f"True Positives (TP): {tp}")
print(f"True Negatives (TN): {tn}")
print(f"False Positives (FP): {fp}")
print(f"False Negatives (FN): {fn}")
print(f"Accuracy: {accuracy:.4f}")
print(f"Error Rate: {error_rate:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"R² Score (using probabilities): {r2:.4f}")
```

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.86      0.97      0.91        63
           1       0.93      0.73      0.82        37

    accuracy                           0.88       100
   macro avg       0.90      0.85      0.86       100
weighted avg       0.89      0.88      0.88       100


=== Performance Metrics ===
True Positives (TP): 27
True Negatives (TN): 61
False Positives (FP): 2
False Negatives (FN): 10
Accuracy: 0.8800
Error Rate: 0.1200
Precision: 0.9310
Recall: 0.7297
R² Score (using probabilities): 0.6407
```