```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns


df = pd.read_csv('Social_Network_Ads.csv')


df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()
```

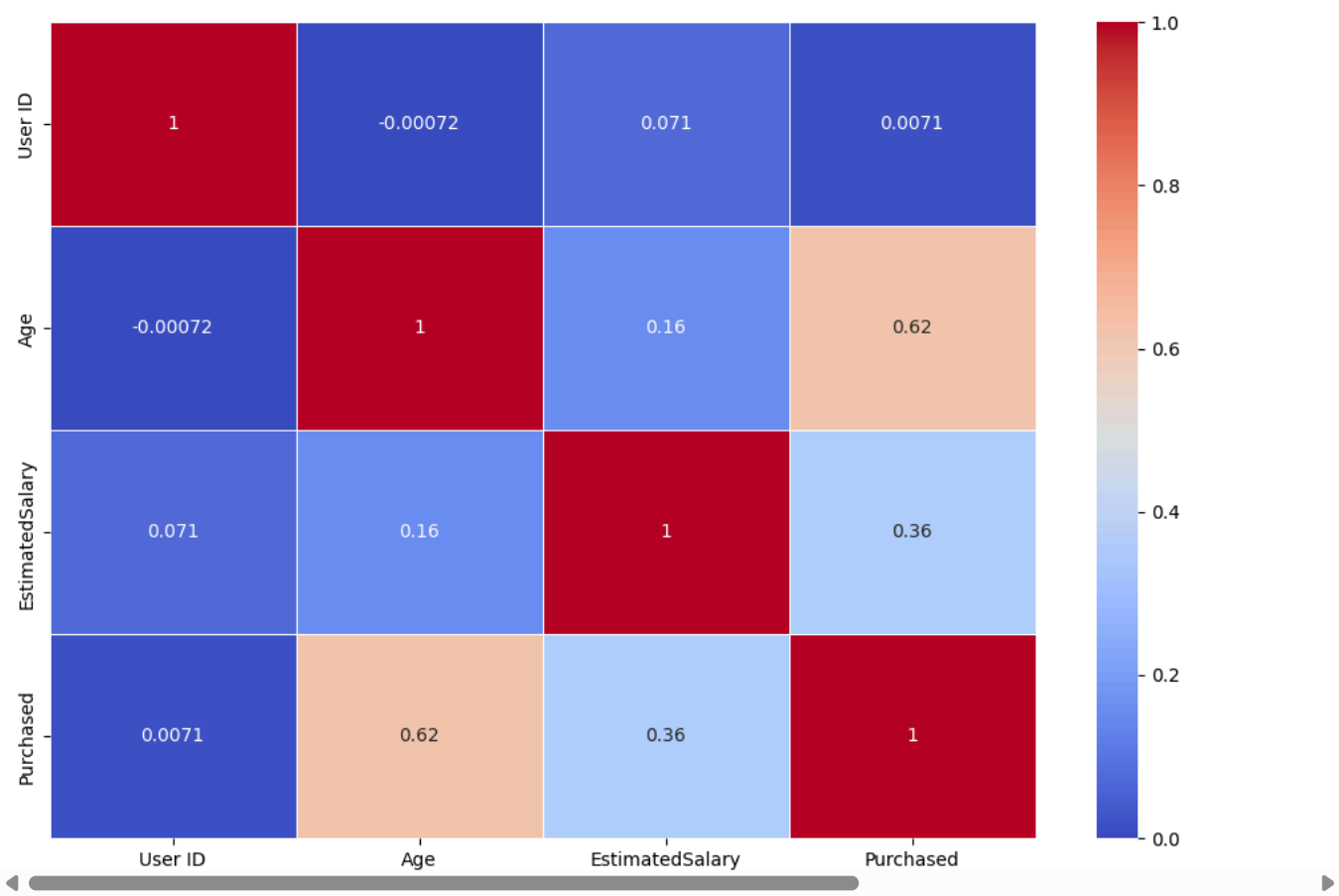
	0
User ID	0
Gender	0
Age	0
EstimatedSalary	0
Purchased	0

dtype: int64

```
corr = df.select_dtypes(include=np.number).corr()
corr.shape
```

```
(4, 4)
```

```
corr_matrix = df.select_dtypes(include=np.number).corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.show()
```




```
x = df[['Age','EstimatedSalary']]
y = df['Purchased']
```

```
x
```

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
...	...	...
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

400 rows × 2 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size =0.3,random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)
```

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

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

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)
print(cm)
```

```
[[74  5]
 [11 30]]
```

```
TN = cm[0, 0]
FP = cm[0, 1]
FN = cm[1, 0]
TP = cm[1, 1]
```

```
accuracy = (TP + TN) / (TP + TN + FP + FN)
error_rate = (FP + FN) / (TP + TN + FP + FN)
precision = TP / (TP + FP)
recall = TP / (TP + FN)
```

```
print("Accuracy:", accuracy)
```