Assignment -1
Python Programming

Assignment Date	8 September 2022
Student Name	Mr. Lavan R P
Student Roll Number	910619104042
Maximum Marks	2 Marks

In [6]:

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

In [7]:

```
#Dataset loaded
df = pd.read_csv('Churn_Modelling.csv')
df.shape
```

Out[7]:

```
(10000, 14)
```

In [8]:

```
df.head()
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3
3	4	15701354	Boni	699	France	Female	39	1	0.00	2
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1

In [9]:

```
df.dtypes
```

Out[9]:

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
```

```
Exited               int64
dtype: object
```

In [10]:

```
df.columns
```

Out[10]:

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

Univariate analysis

In [34]:

```
df['EstimatedSalary'].mean()
```

Out[34]:

```
100090.2398809998
```

In [35]:

```
df['EstimatedSalary'].std()
```

Out[35]:

```
57510.49281769822
```

In [36]:

```
df['EstimatedSalary'].value_counts()
```

Out[36]:

```
24924.92     2
101348.88    1
55313.44     1
72500.68     1
182692.80    1
            ..
120893.07    1
188377.21    1
55902.93     1
4523.74      1
38190.78     1
Name: EstimatedSalary, Length: 9999, dtype: int64
```

In [37]:

```
df.hist(column='EstimatedSalary',grid=False,edgecolor="red")
```

Out[37]:

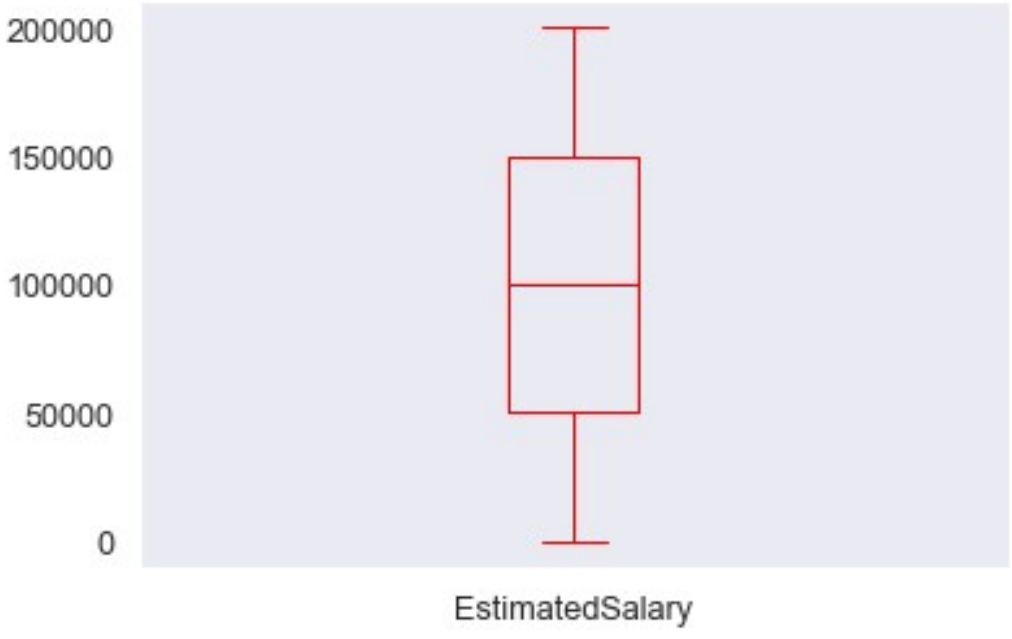
```
array([[<AxesSubplot:title={'center':'EstimatedSalary'}>]], dtype=object)
```

In [38]:

```
df.boxplot(column=['EstimatedSalary'], grid=False, color='red')
```

Out[38]:

```
<AxesSubplot:>
```

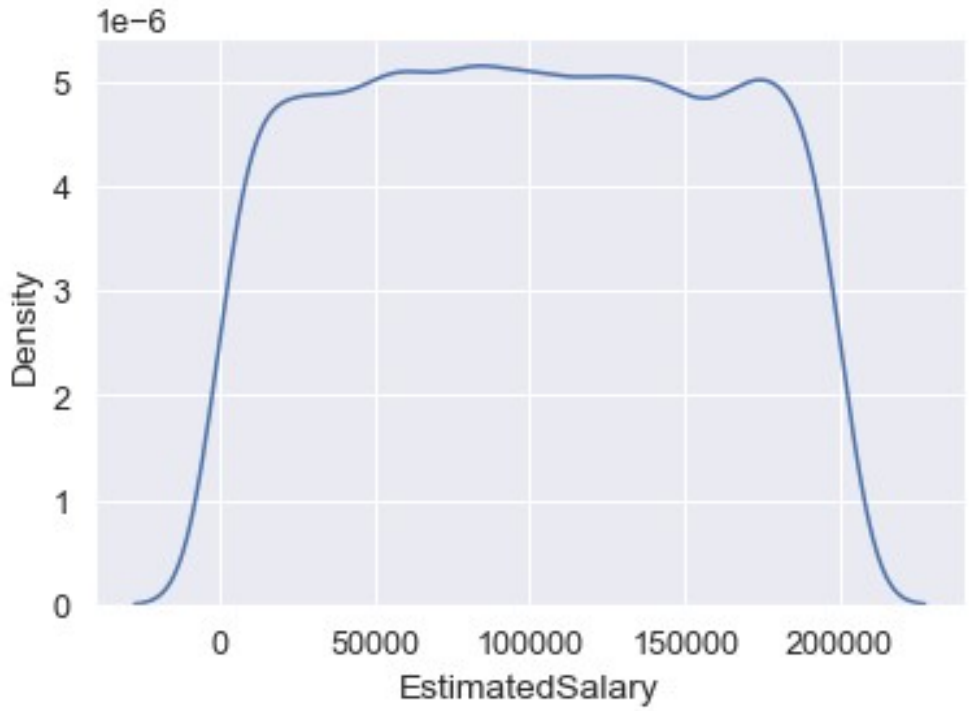


In [39]:

```
sns.kdeplot(df['EstimatedSalary'])
```

Out[39]:

```
<AxesSubplot:xlabel='EstimatedSalary', ylabel='Density'>
```



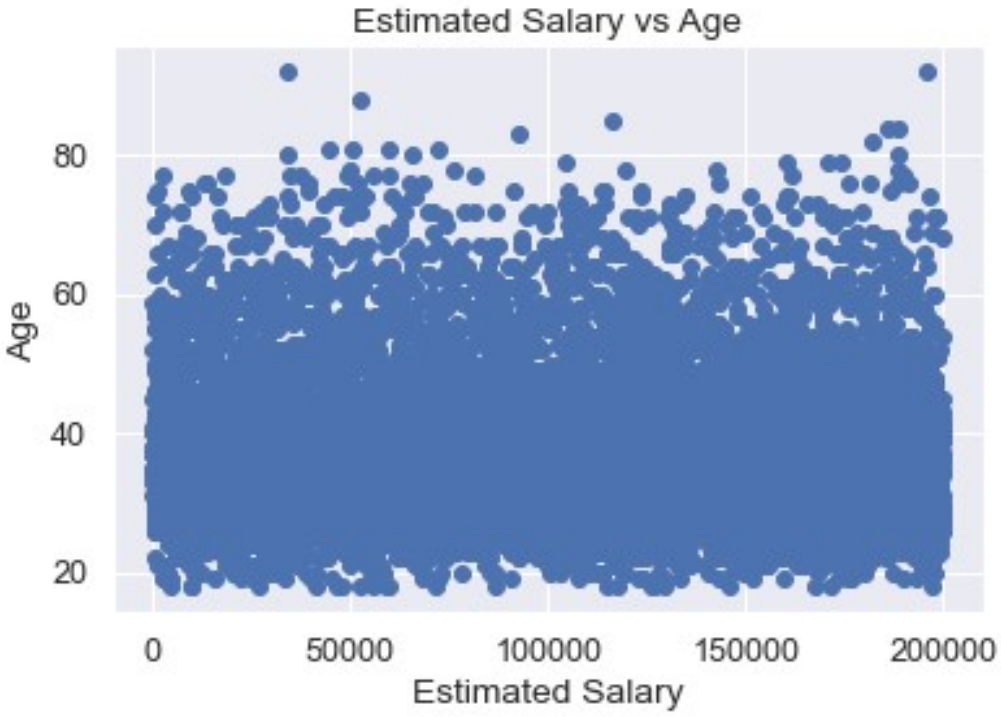
Bivariate analysis

In [41]:

```
# Scatterplots
plt.scatter(df.EstimatedSalary, df.Age)
plt.title('Estimated Salary vs Age')
plt.xlabel('Estimated Salary')
plt.ylabel('Age')
```

Out[41]:

```
Text(0, 0.5, 'Age')
```



In [42]:

```
# Correlation Coefficients
df.corr()
```

Out[42]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCar
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866
EstimatedSalary	-0.005988	0.015271	-0.001384	-	0.007784	0.012797	0.014204	-0.009933

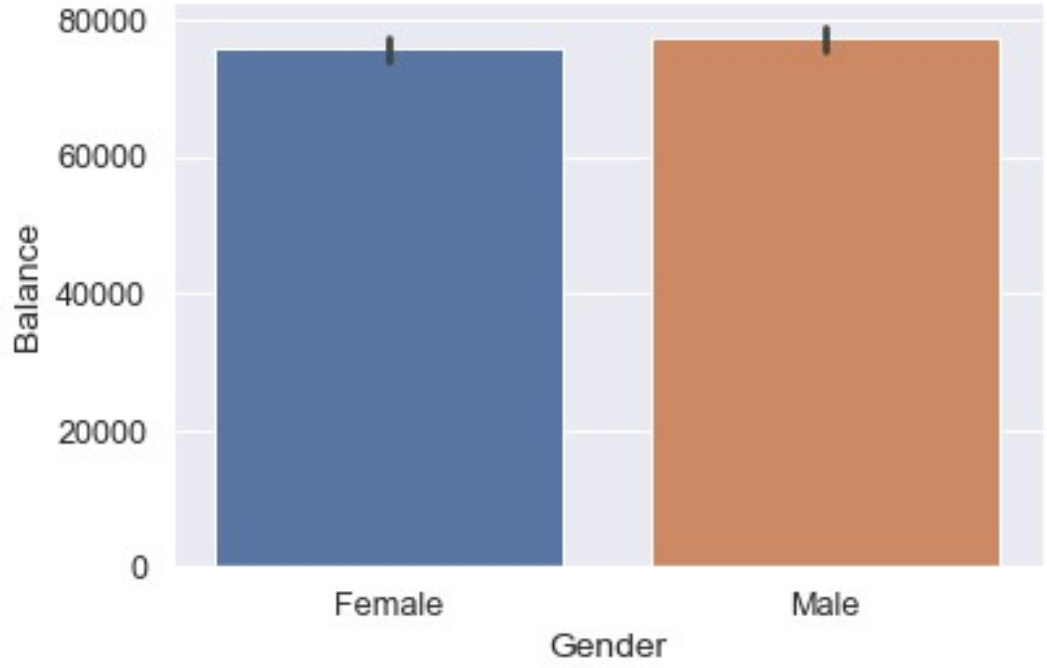
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCar
				0.007201				
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138

In [43]:

```
# Simple Linear Regression
sns.barplot(x='Gender',y='Balance',data=df)
```

Out[43]:

```
<AxesSubplot:xlabel='Gender', ylabel='Balance'>
```

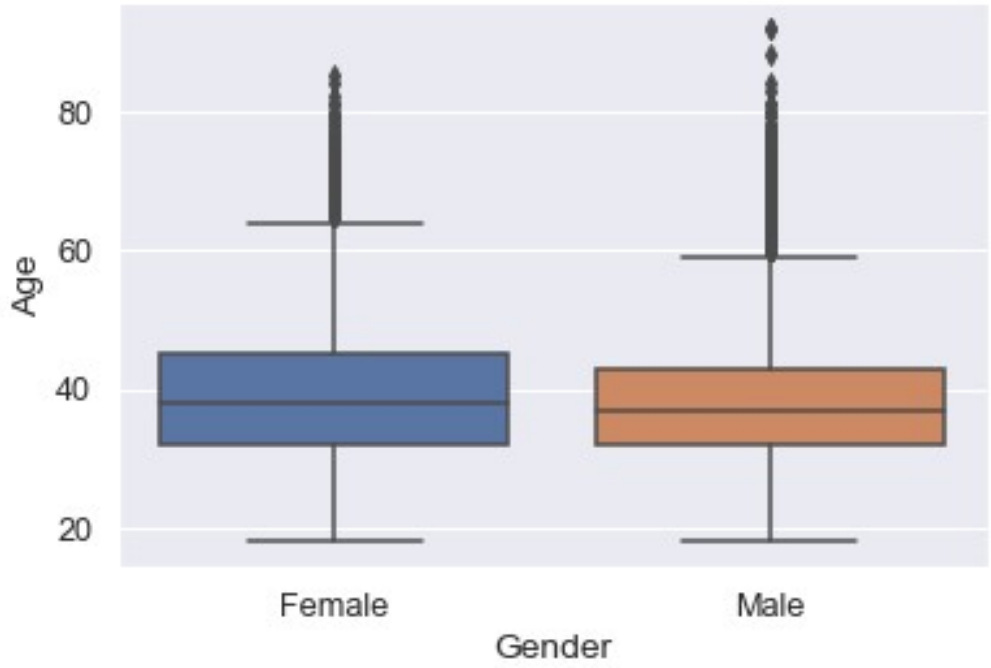


In [44]:

```
sns.boxplot(x='Gender',y='Age',data=df)
```

Out[44]:

```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```



In [45]:

```
sns.violinplot(x='Gender',y='Age',data=df, palette='coolwarm_r')
```

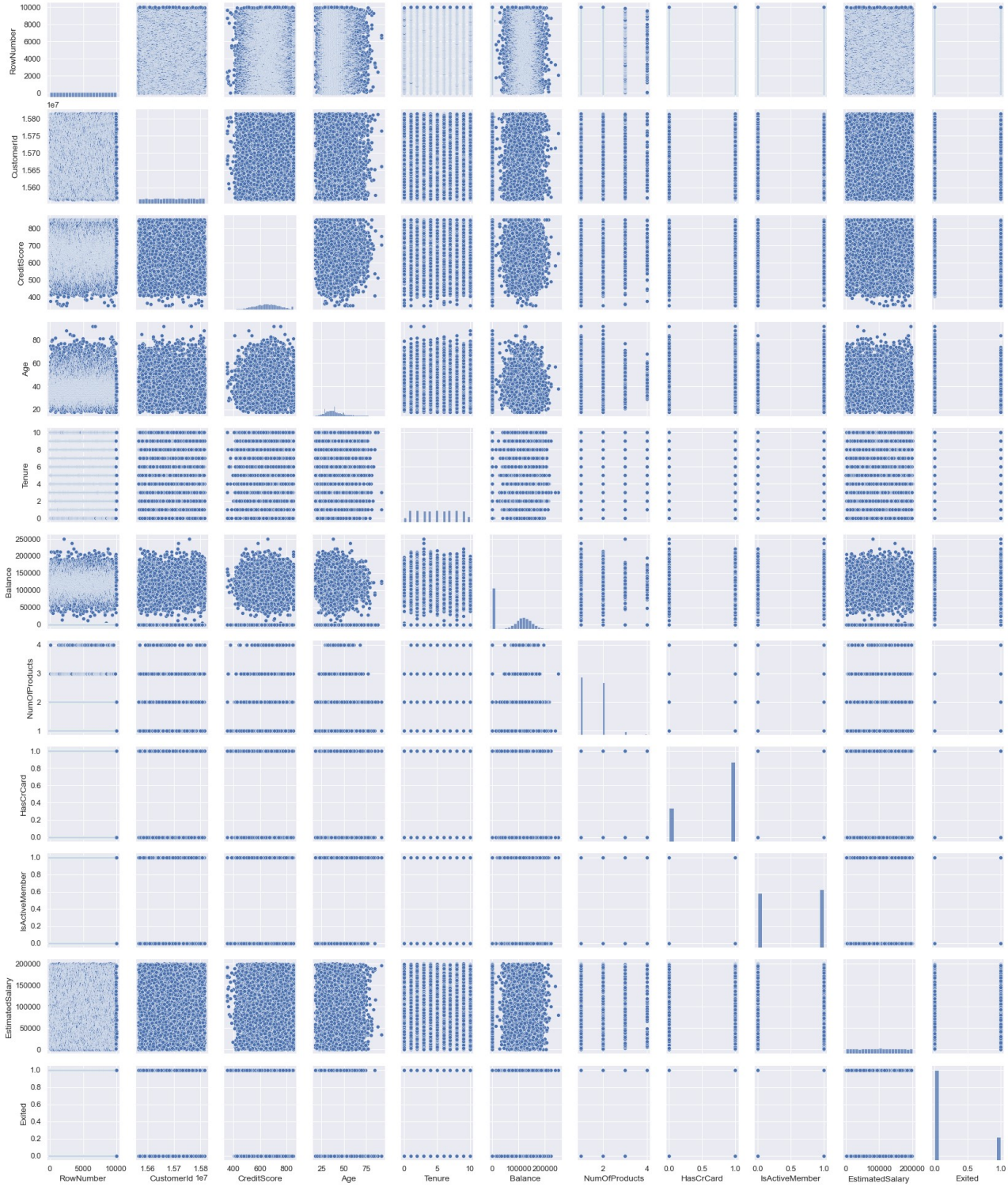
Out[45]:

```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```

Multivariate Analysis

In [46]:

```
# Pair Plot
sns.pairplot(data=df,aspect=.85);
```



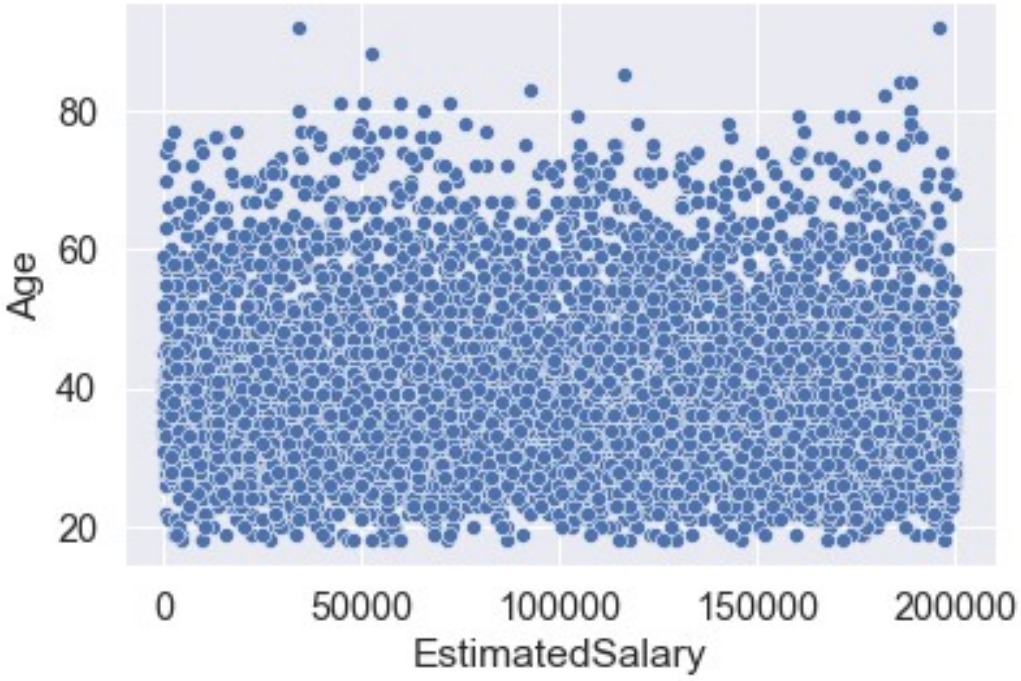
In [47]:

```
# Scatter Plot
sns.set(font_scale=1.3)
```

```
sns.scatterplot(x='EstimatedSalary',y='Age',data=df)
plt.xlabel('EstimatedSalary')
plt.ylabel('Age')
```

Out[47]:

```
Text(0, 0.5, 'Age')
```



In [49]:

```
# Heat Map
sns.set(font_scale=1.15)
plt.figure(figsize=(30,4))
sns.heatmap(df.corr(),cmap='RdBu_r',annot=True,vmin=-1, vmax=1);
```

In [51]:

```
#Missing values in the dataset
df.isnull().sum().sum()
```

Out[51]:

```
0
```

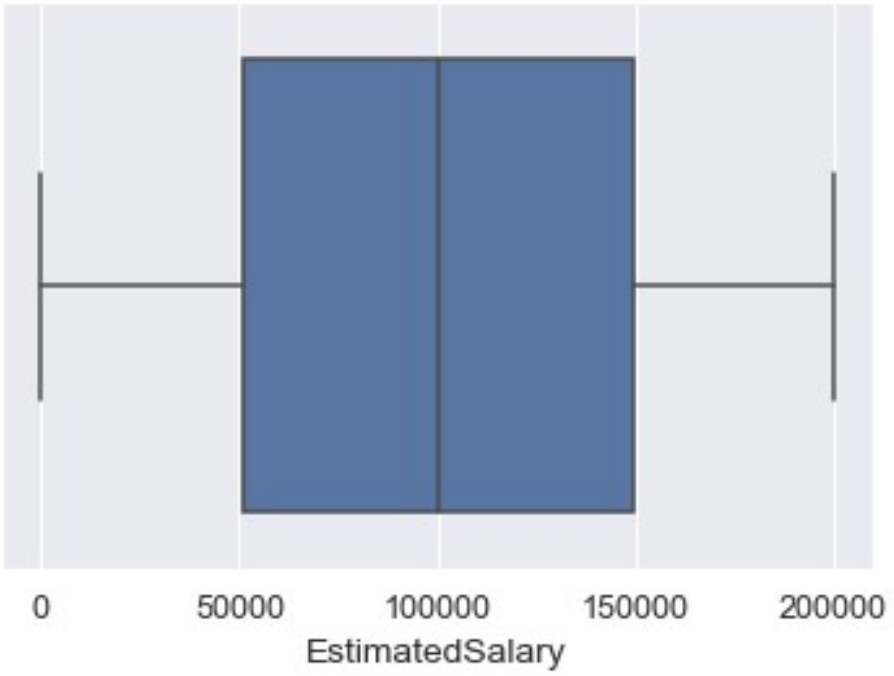
Outliers Identification

In [58]:

```
#Visualization
sns.boxplot(df['EstimatedSalary'],data=df)
C:\Users\Ilyas\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

Out[58]:

```
<AxesSubplot:xlabel='EstimatedSalary'>
```



In [59]:

```
#Skewness
df['EstimatedSalary'].skew()
```

Out[59]:

```
0.0020853576615585162
```

In [60]:

```
#Interquartile Range
Q1=df['EstimatedSalary'].quantile(0.25)
Q3=df['EstimatedSalary'].quantile(0.75)
IQR=Q3-Q1
print(IQR)
98386.1375
```

Outliers Treatment

In [66]:

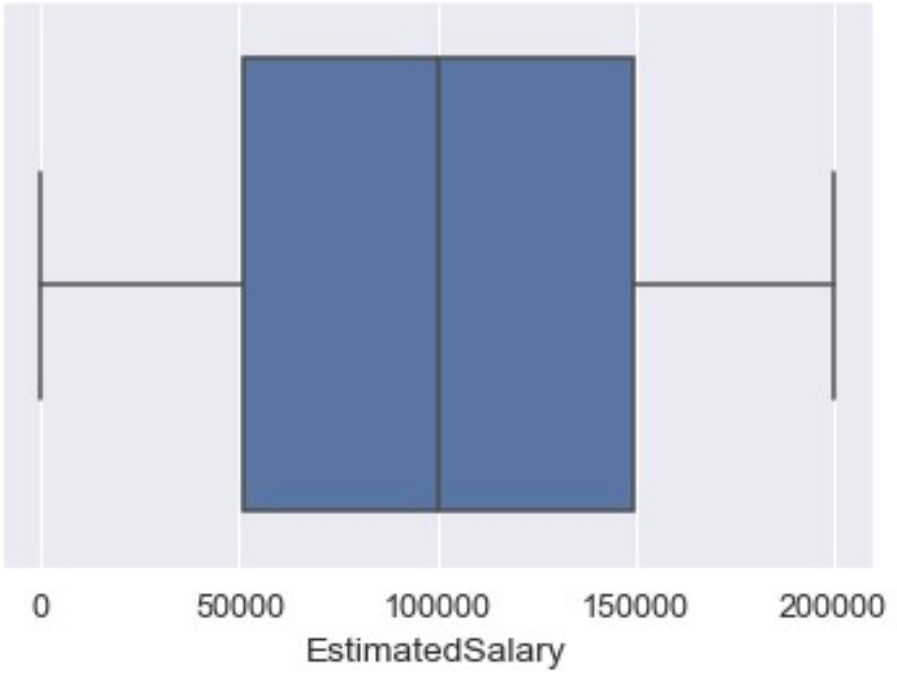
```
# 1.Flooring and Capping.
# 2.Trimming.
Q1=df['EstimatedSalary'].quantile(0.25)
Q3=df['EstimatedSalary'].quantile(0.75)
IQR=Q3-Q1
whisker_width = 1.5
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
df['EstimatedSalary']=np.where((df['EstimatedSalary'])>upper_whisker,upper_wh
isker,np.where(df['EstimatedSalary']<lower_whisker,lower_whisker,df['Estimate
dSalary']))
```

In [67]:

```
sns.boxplot(df['EstimatedSalary'],data=df)
C:\Users\Ilyas\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

Out[67]:

```
<AxesSubplot:xlabel='EstimatedSalary'>
```



Check for Categorical Columns

In [69]:

```
data_numeric = df[['Age','Balance', 'NumOfProducts','HasCrCard',
'IsActiveMember',      'EstimatedSalary',     'Exited']]
data_categorical = df[['Surname', 'Geography', 'Gender']]
```

In [70]:

```
data_numeric.head()
```

Out[70]:

	Age	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	42	0.00	1	1	1	101348.88	1
1	41	83807.86	1	0	1	112542.58	0
2	42	159660.80	3	1	0	113931.57	1
3	39	0.00	2	0	0	93826.63	0
4	43	125510.82	1	1	1	79084.10	0

In [71]:

```
data_categorical.head()
```

Out[71]:

	Surname	Geography	Gender
0	Hargrave	France	Female
1	Hill	Spain	Female
2	Onio	France	Female
3	Boni	France	Female
4	Mitchell	Spain	Female

In [72]:

```
print(df['Surname'].unique())
print(df['Geography'].unique())
print(df['Gender'].unique())
['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']
['France' 'Spain' 'Germany']
['Female' 'Male']
```

Perform Encoding

In [78]:

```
from sklearn.preprocessing import OneHotEncoder
Surname_encoder = OneHotEncoder()
Surname_reshaped = np.array(data_categorical['Surname']).reshape(-1, 1)
Surname_values = Surname_encoder.fit_transform(Surname_reshaped)
print(data_categorical['Surname'][:5])
print(Surname_values.toarray()[:5])
print(Surname_encoder.inverse_transform(Surname_values)[:5])
0    Hargrave
1        Hill
2        Onio
3        Boni
```

```
4    Mitchell
Name: Surname, dtype: object
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[['Hargrave']
 ['Hill']
 ['Onio']
 ['Boni']
 ['Mitchell']]
```

One Hot Encoding

In [79]:

```
Gender_encoder = OneHotEncoder()
Gender_reshaped = np.array(data_categorical['Gender']).reshape(-1, 1)
Gender_values = Gender_encoder.fit_transform(Gender_reshaped)
print(data_categorical['Gender'][:5])
print(Gender_values.toarray()[:5])
print(Gender_encoder.inverse_transform(Gender_values)[:5])
```

```
0    Female
1    Female
2    Female
3    Female
4    Female
Name: Gender, dtype: object
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]]
[['Female']
 ['Female']
 ['Female']
 ['Female']
 ['Female']]
```

Label Encoding

In [80]:

```
from sklearn.preprocessing import LabelEncoder
Gender_encoder = LabelEncoder()
```

In [81]:

```
Gender_encoder = LabelEncoder()
```

```
Gender_values = Gender_encoder.fit_transform(data_categorical['Gender'])

print("Before Encoding:", list(data_categorical['Gender'][:5]))
print("After Encoding:", Gender_values[:5])
print("Result after inverse:",
Gender_encoder.inverse_transform(Gender_values[:5]))
Before Encoding: ['Female', 'Female', 'Female', 'Female', 'Female']
After Encoding: [0 0 0 0 0]
Result after inverse: ['Female' 'Female' 'Female' 'Female' 'Female']
```

Split the data into Dependent and Independent variables.

In [83]:

```
a=df.iloc[:, :-1].values
print(a)
[[1 15634602 'Hargrave' ... 1 1 101348.88]
 [2 15647311 'Hill' ... 0 1 112542.58]
 [3 15619304 'Onio' ... 1 0 113931.57]
 ...
 [9998 15584532 'Liu' ... 0 1 42085.58]
 [9999 15682355 'Sabbatini' ... 1 0 92888.52]
 [10000 15628319 'Walker' ... 1 0 38190.78]]
```

In [85]:

```
b= df.iloc[:, -1].values
print(b)
[1 0 1 ... 1 1 0]
```

Scale the Independent variables

In [88]:

```
from sklearn.preprocessing import MinMaxScaler
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
a = df[['Age', 'NumOfProducts']]
scaleda = scale.fit_transform(a)
print(scaleda)
[[ 0.29351742 -0.91158349]
 [ 0.19816383 -0.91158349]
 [ 0.29351742  2.52705662]
 ...
 [-0.27860412 -0.91158349]
 [ 0.29351742  0.80773656]
 [-1.04143285 -0.91158349]]
```

Split the data into Training and Testing

In [95]:

```
#LinearRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
x = df.iloc[:, :-1]
y= df.iloc[:, -1]
a_train, a_test, b_train, b_test = train_test_split(
    a, y, test_size=0.05, random_state=0)
```

In [96]:

```
a_train
```

Out[96]:

	Age	NumOfProducts
799	42	1
1069	40	1
8410	46	1
9436	38	3
5099	45	1
...	...	...
9225	32	2
4859	22	1
3264	35	2
9845	38	2
2732	48	1

9500 rows × 2 columns

In [97]:

```
b_train
```

Out[97]:

```
799     0
1069    1
8410    0
9436    0
5099    1
       ..
9225    0
4859    0
3264    0
9845    0
2732    1
Name: Exited, Length: 9500, dtype: int64
```

In [98]:

```
a_test
```

Out[98]:

	Age	NumOfProducts
9394	35	1
898	40	1
2398	42	1
5906	32	1
2343	38	2
...	...	...
8938	47	1
9291	36	2
491	41	1
2021	18	1
4299	30	2

500 rows × 2 columns

In [99]:

```
b_test
```

Out[99]:

```
9394    0
898     1
2398    0
5906    0
2343    0
       ..
8938    0
9291    0
491     0
2021    0
4299    0
Name: Exited, Length: 500, dtype: int64
```

In [ ]:

In [ ]: