Churn Assignment Part B - Data Visualisation

You are tasked with doing some exploratory data analysis, which is the first step in building a model to predict churn. Since this process is usually very large, we will look at a subset of the total plots you would need to complete this.

1. First you should look at the differences in churn rates, split by the different categorical variables. Produce the appropriate visualisation to compare the average churn rate, split by:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.express as px
%matplotlib inline
```

In [2]:
```
table = pd.read_csv('churn_data.csv', sep=',')
```

In [3]:
```
table.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

In [4]:
```
table.drop(['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
```

In [5]:
```
# i. Geography
group_geo = table.groupby('Geography').mean().reset_index()
```

In [6]:
```
group_geo.head()
```

Out[6]:

	Geography	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	France	649.668329	38.511767	5.004587	62092.636516	1.530913	0.706621	0.516753	99899.180814	0.161548
1	Germany	651.453567	39.771622	5.009964	119730.116134	1.519729	0.713830	0.497409	101113.435102	0.324432
2	Spain	651.333872	38.890997	5.032297	61818.147763	1.539362	0.694792	0.529673	99440.572281	0.166734

In [7]:
```
# ii. Gender

group_gender = table.groupby('Gender').mean().reset_index()
```

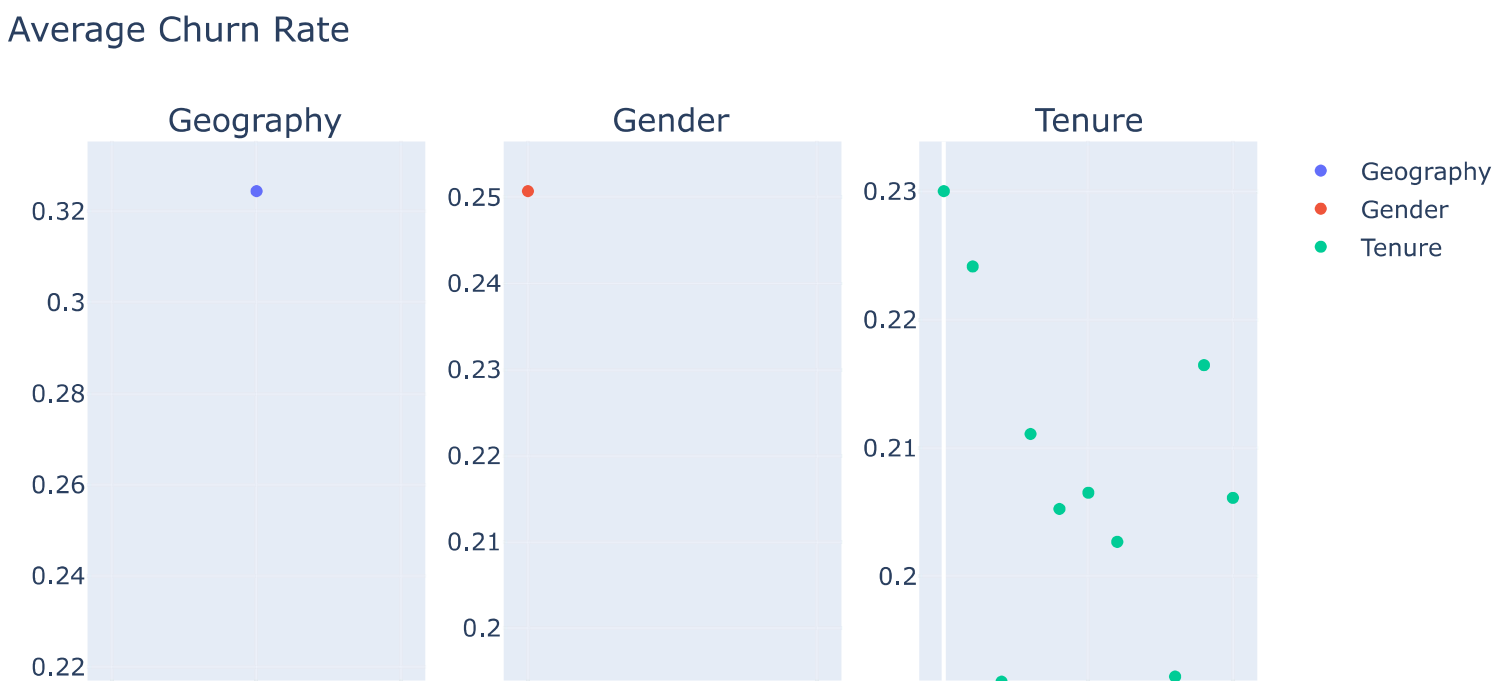
In [8]:
```
group_gender.head()
```

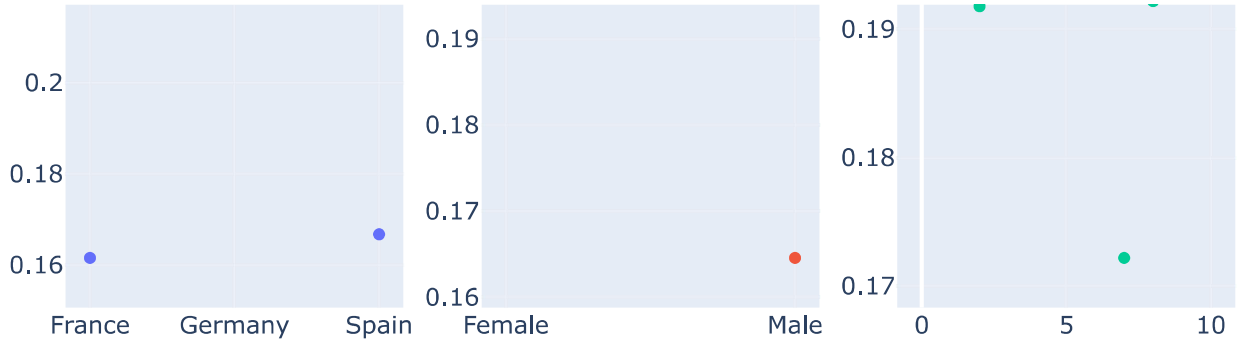
Out[8]:

	Gender	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	Female	650.831389	39.238389	4.966102	75659.369139	1.544134	0.702619	0.502751	100601.541382	0.250715
1	Male	650.276892	38.658237	5.051677	77173.974506	1.518600	0.707898	0.525380	99664.576931	0.164559

In [9]:
```
# iii. Tenure

group_tenure = table.groupby('Tenure').mean().reset_index()
```

In [10]:
```
fig = make_subplots(rows=1, cols=3, subplot_titles=('Geography',  'Gender', 'Tenure'))


fig.add_trace(
    go.Scatter(name='Geography',x=group_geo['Geography'], y = group_geo['Exited'], mode = 'markers' ),
    row=1, col=1
)

fig.add_trace(
    go.Scatter(name='Gender',x=group_gender['Gender'], y= group_gender['Exited'], mode = 'markers' ),
    row=1, col=2
)

fig.add_trace(
    go.Scatter(name='Tenure',x=group_tenure['Tenure'],y= group_tenure['Exited'], mode = 'markers' ),
    row=1, col=3
)

fig.update_layout(height=600, width=800, title_text="Average Churn Rate")
fig.show()
```

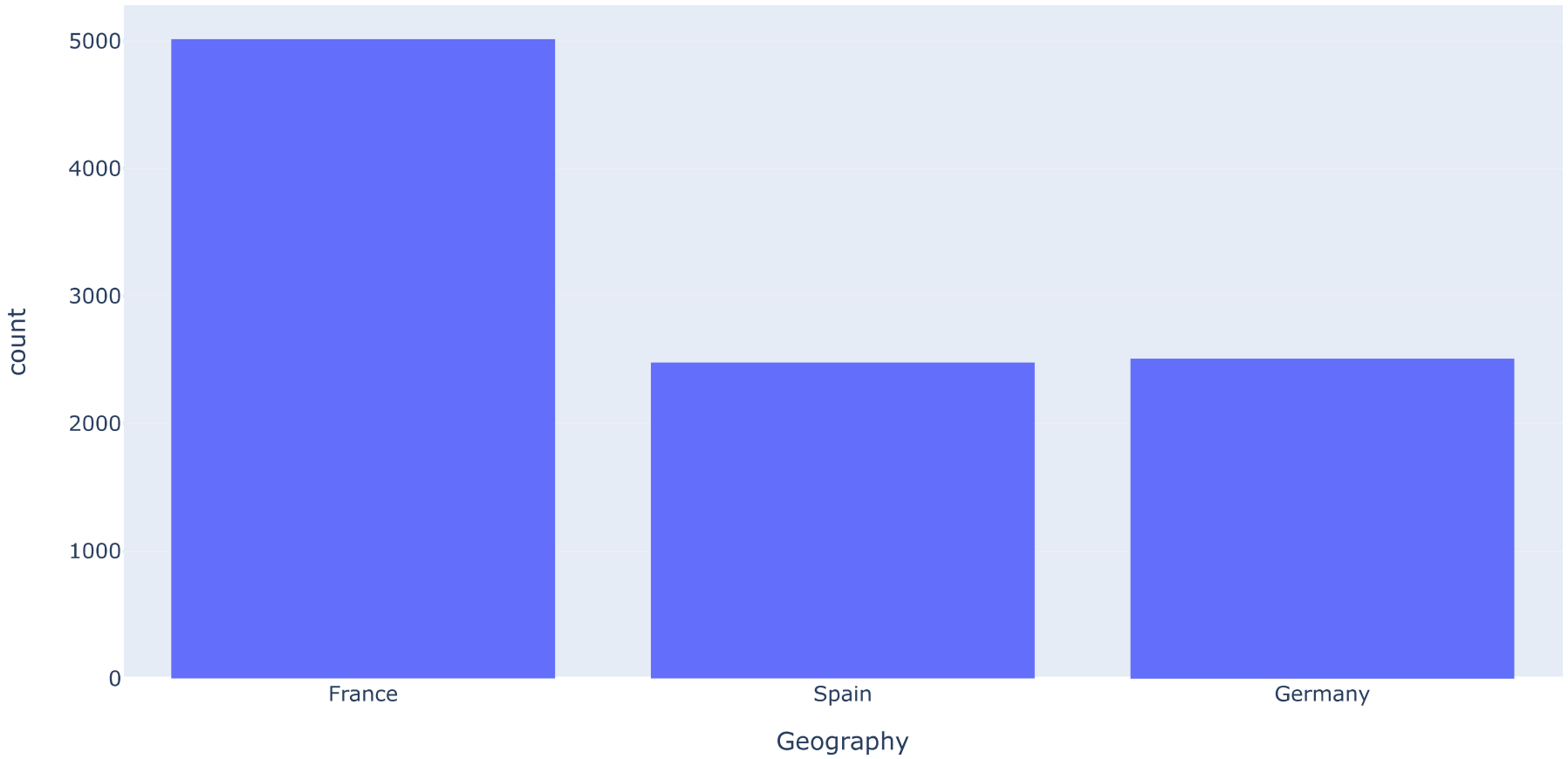




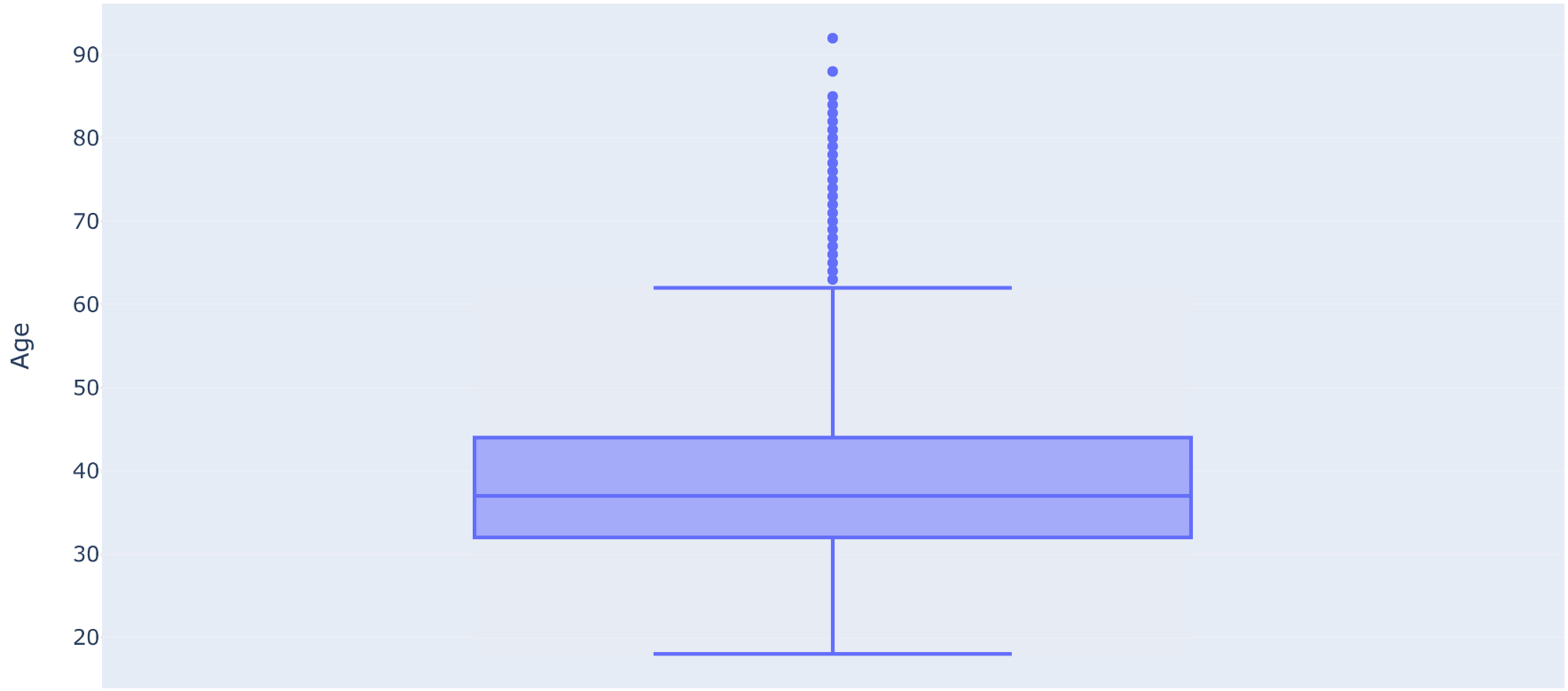
For maximum marks, make sure plots are correctly labelled.

2. We would also like to know how the data is distributed. Some models require features to be normally distributed, and highly skewed variables can affect summary statistics if left unchecked. Produce the appropriate visualisation for the distribution of:

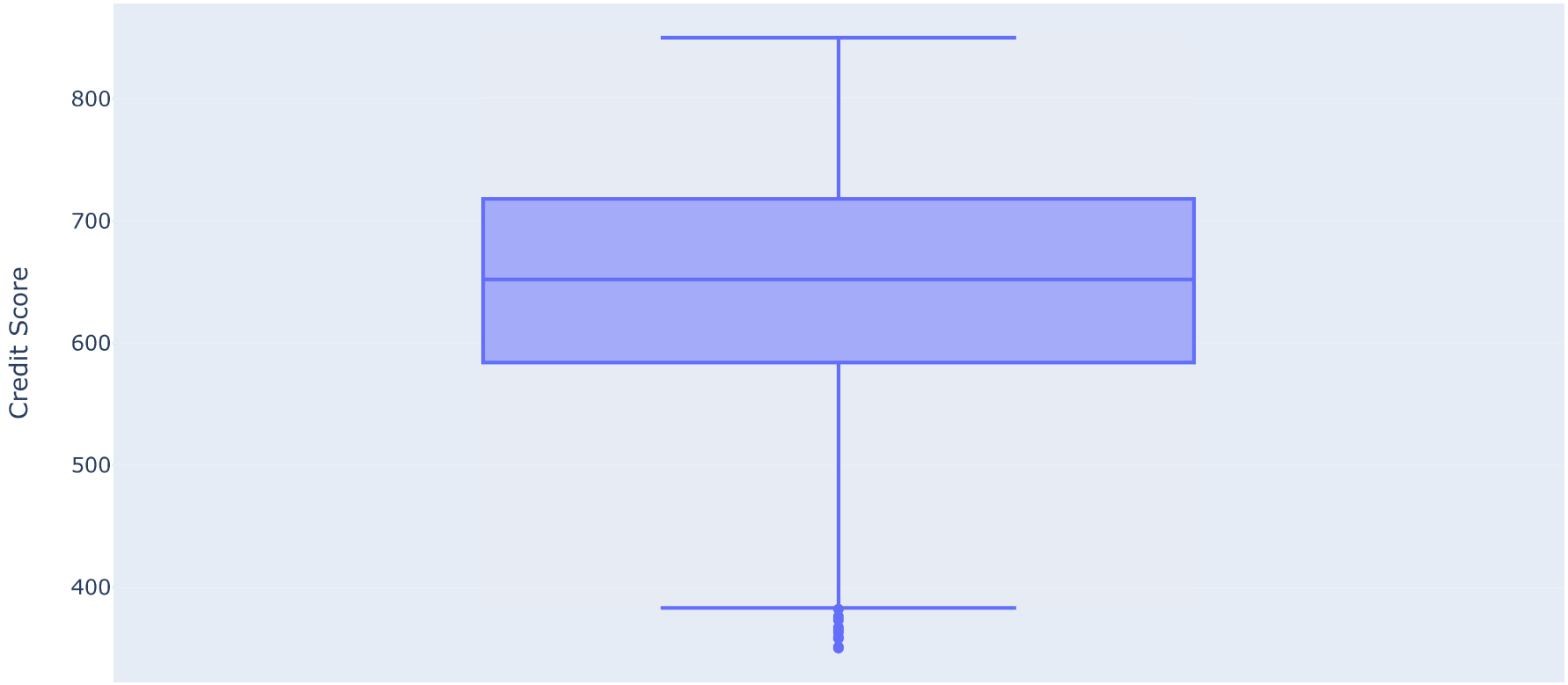
In [11]:
```
# i. Geography
fig = px.histogram(table, x='Geography')
fig.show()
```



In [12]:
```
# ii. Age
fig = px.box(table, y='Age')
fig.show()
```

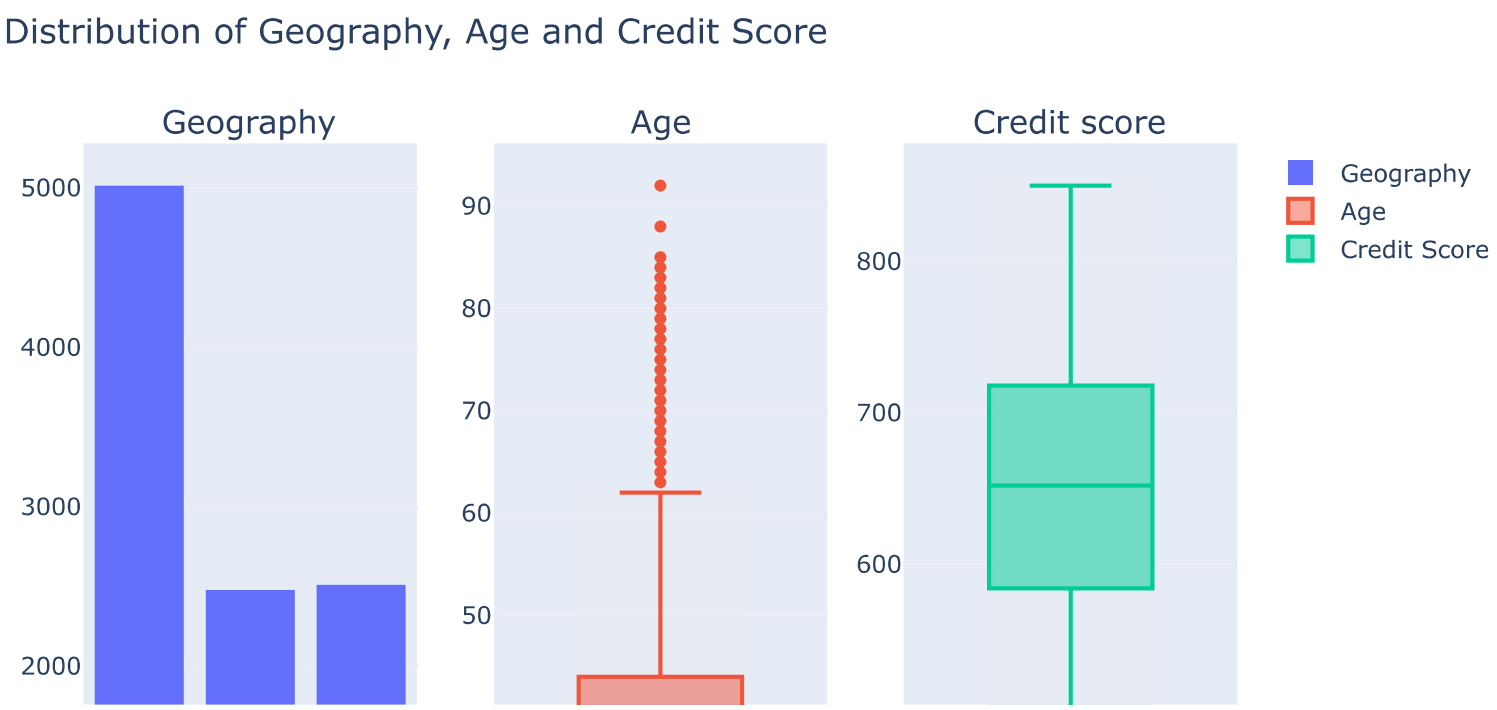


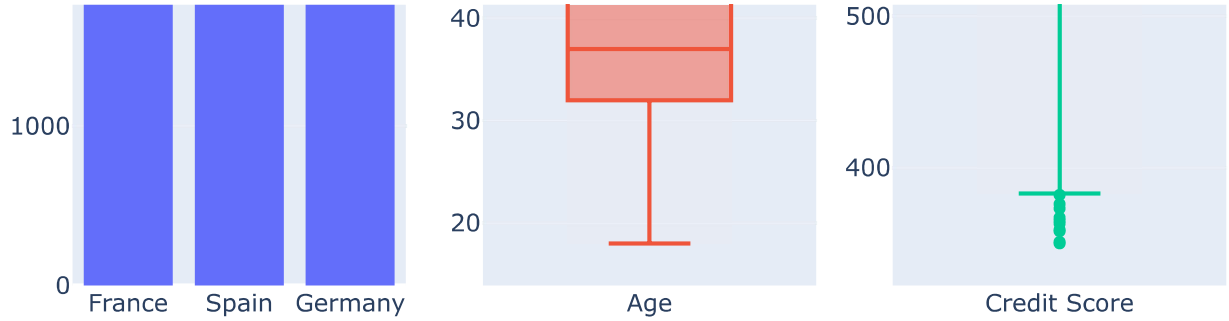
In [13]:
```
# iii. Credit Score
fig = px.box(table, y='CreditScore',labels={'CreditScore':'Credit Score'})
fig.show()
```



3. Combine all of the above visualisations into a subplot (hint: Subplot takes figures created in graph objects, so you may need to recreate some visualisations). For maximum marks, make sure that you correctly label each figure in the subplot.

In [14]:
```
fig = make_subplots(rows=1, cols=3, subplot_titles=('Geography',  'Age', 'Credit score'))


fig.add_trace(
    go.Histogram(name='Geography',x=table['Geography'] ),
    row=1, col=1
)

fig.add_trace(
    go.Box(name='Age',y=table['Age'] ),
    row=1, col=2
)

fig.add_trace(
    go.Box(name='Credit Score',y=table['CreditScore'] ),
    row=1, col=3
)

fig.update_layout(height=600, width=800, title_text="Distribution of Geography, Age and Credit Score")
fig.show()
```





4. You can get the correlation between all columns using df.corr(). Create a bar plot that shows the correlation of each feature with the target. (Make sure to add a title and axis labels)

In [15]:
```
correlation = table.corr()
```

In [16]:
```
correlation.drop('Exited', axis=0,inplace=True)
```

In [17]:
```
correlation = correlation.reset_index()
```

In [18]:
```
correlation
```

Out[18]:

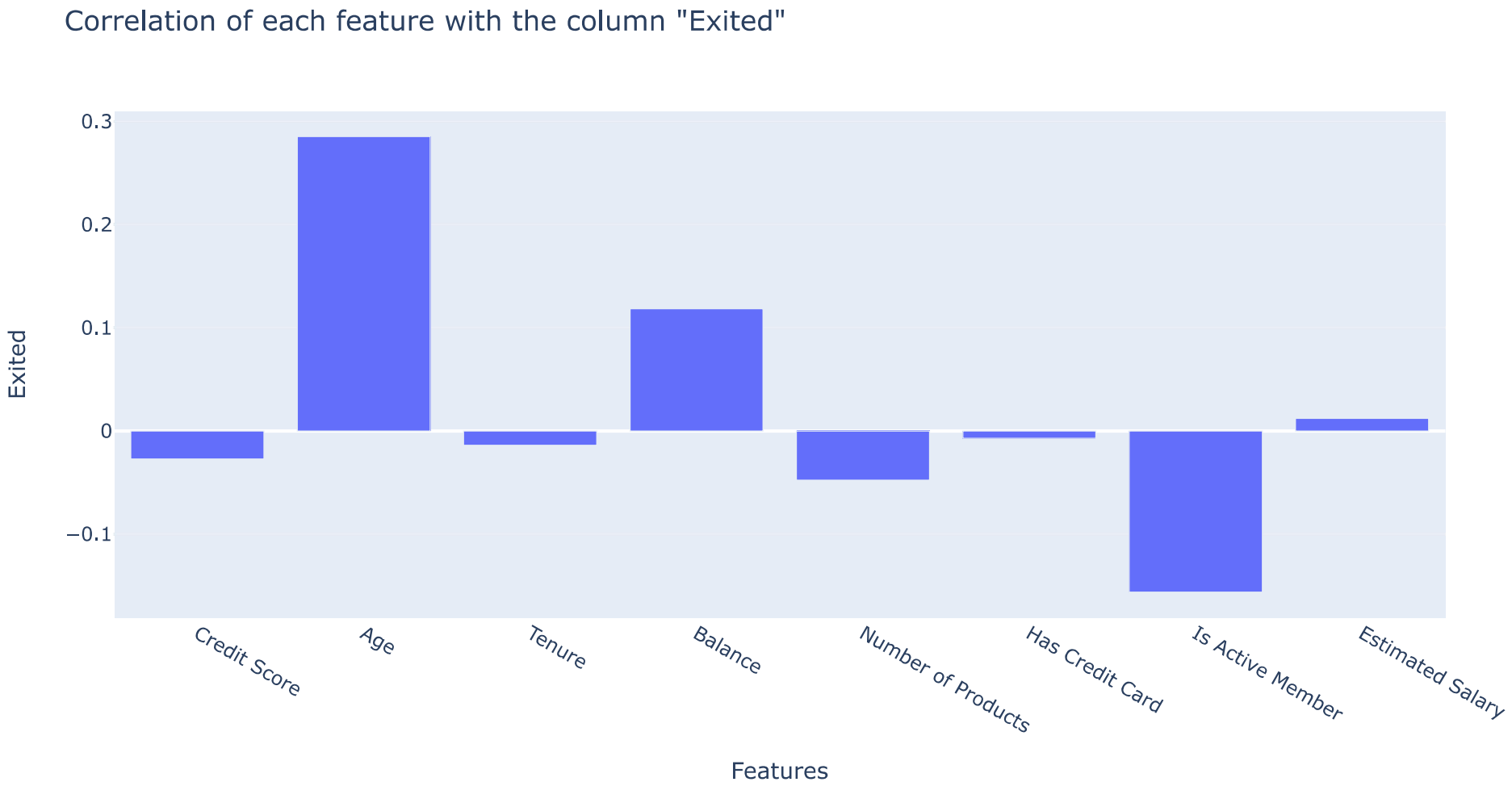
	index	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
1	Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
2	Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
3	Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
4	NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
5	HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
6	IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
7	EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097

In [19]:
```
correlation['index'][0]= 'Credit Score'
correlation['index'][4]= 'Number of Products'
correlation['index'][5]= 'Has Credit Card'
correlation['index'][6]= 'Is Active Member'
correlation['index'][7]= 'Estimated Salary'
```

```
C:\Users\resil\AppData\Local\Temp/ipykernel_12596/3336410515.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\resil\AppData\Local\Temp/ipykernel_12596/3336410515.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\resil\AppData\Local\Temp/ipykernel_12596/3336410515.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\resil\AppData\Local\Temp/ipykernel_12596/3336410515.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

C:\Users\resil\AppData\Local\Temp/ipykernel_12596/3336410515.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame
```

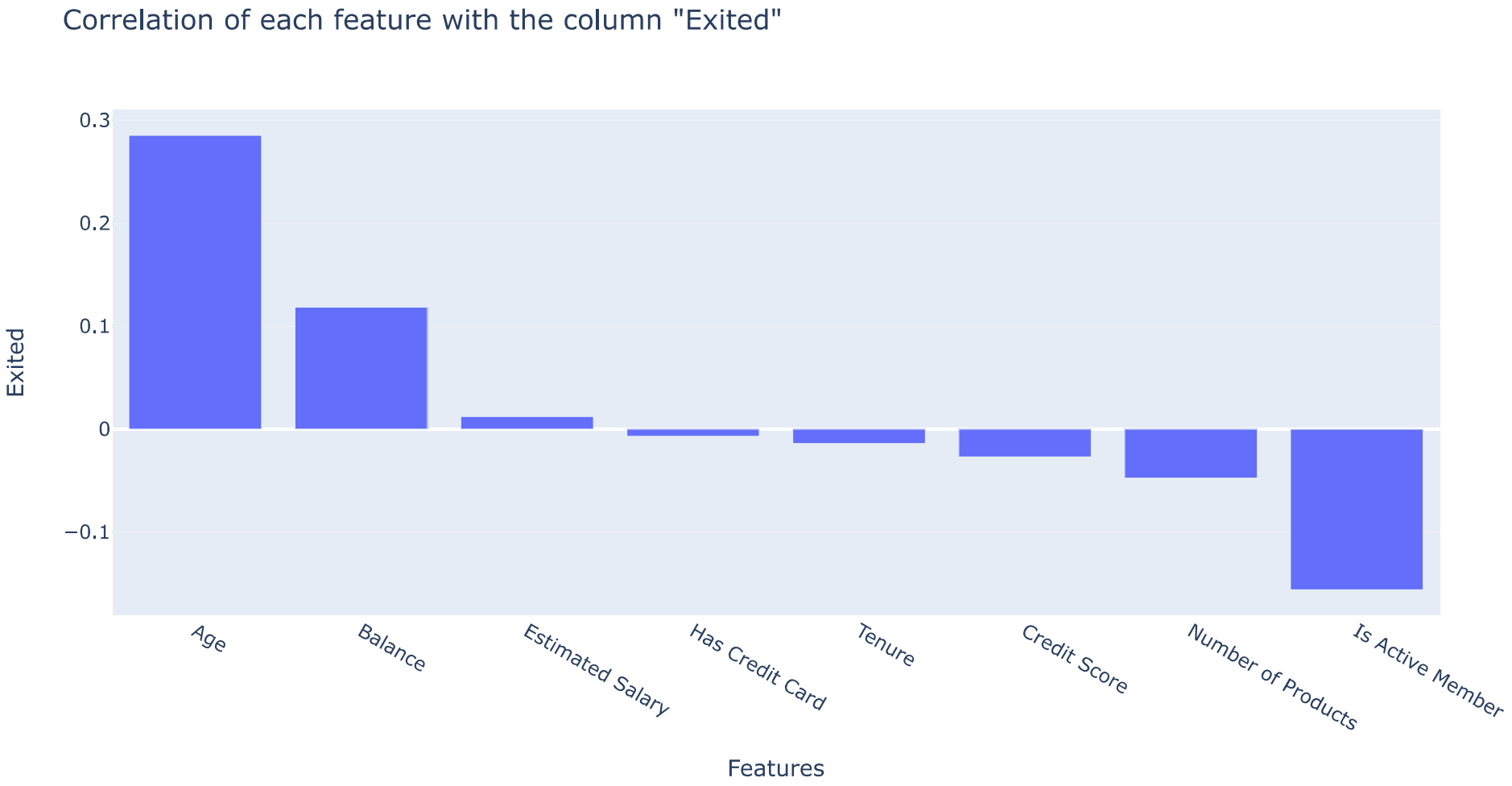
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

In [20]:
```
fig = px.bar(data_frame = correlation,x='index', y= 'Exited',title='Correlation of each feature with the column "Exited"', labels

fig.show()
```



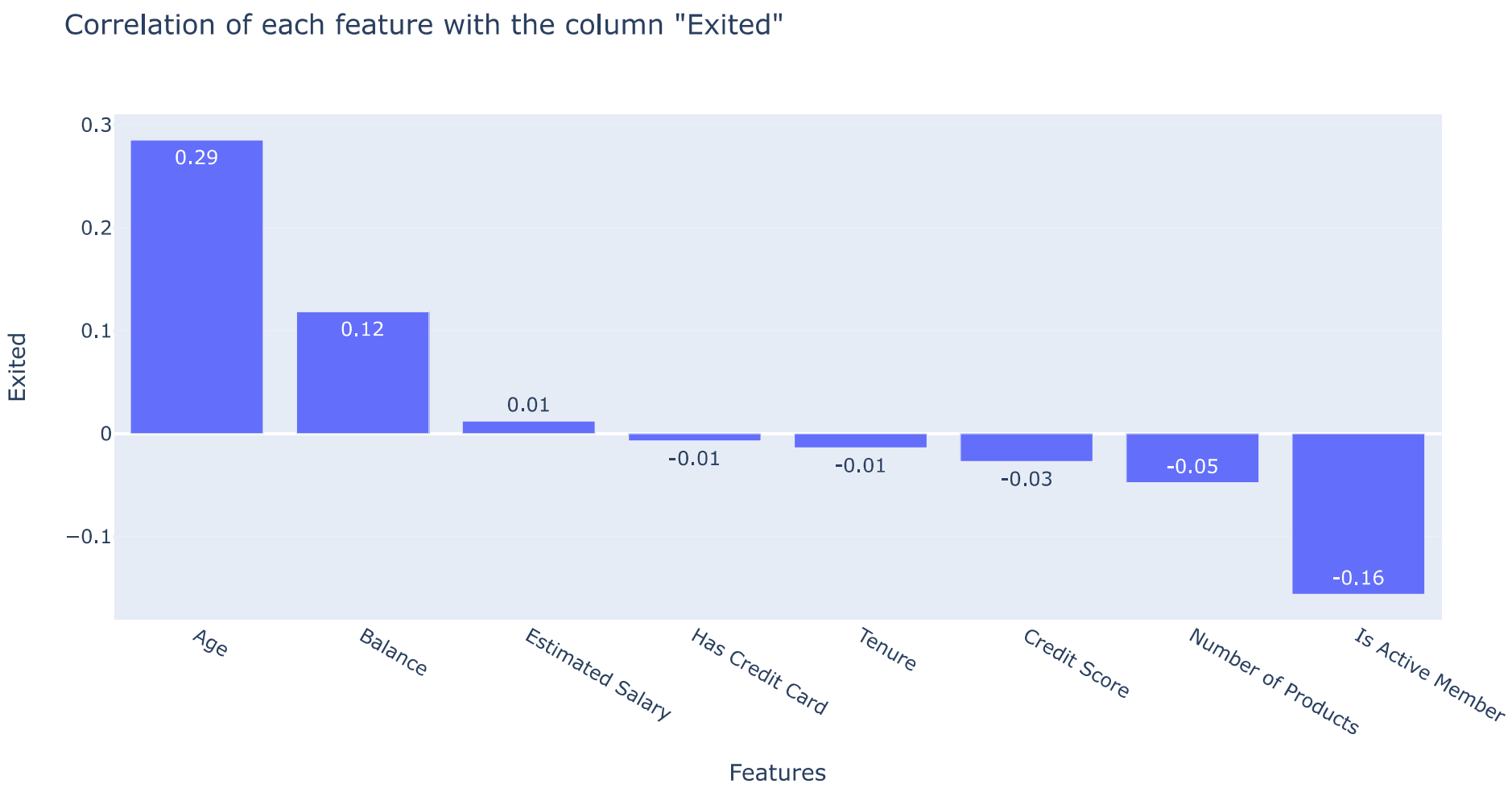
4.1. Order the bars so that the feature with the highest correlation is the first bar.

In [21]:
```
fig = px.bar(data_frame = correlation,x='index', y= 'Exited',title='Correlation of each feature with the column "Exited"', labels

fig.show()
```



4.2. Add the correlation value to the top of each bar

In [22]:
```
fig = px.bar(data_frame = correlation,x='index', y= 'Exited',title='Correlation of each feature with the column "Exited"', labels

fig.show()
```



4.3. Add a line to the figure which shows the average correlation (hint: This will require adding an extra trace).

In [23]:
```
average=round(correlation['Exited'].mean(),2)
```

In [24]:
```
average
```

Out[24]:
```
0.02
```

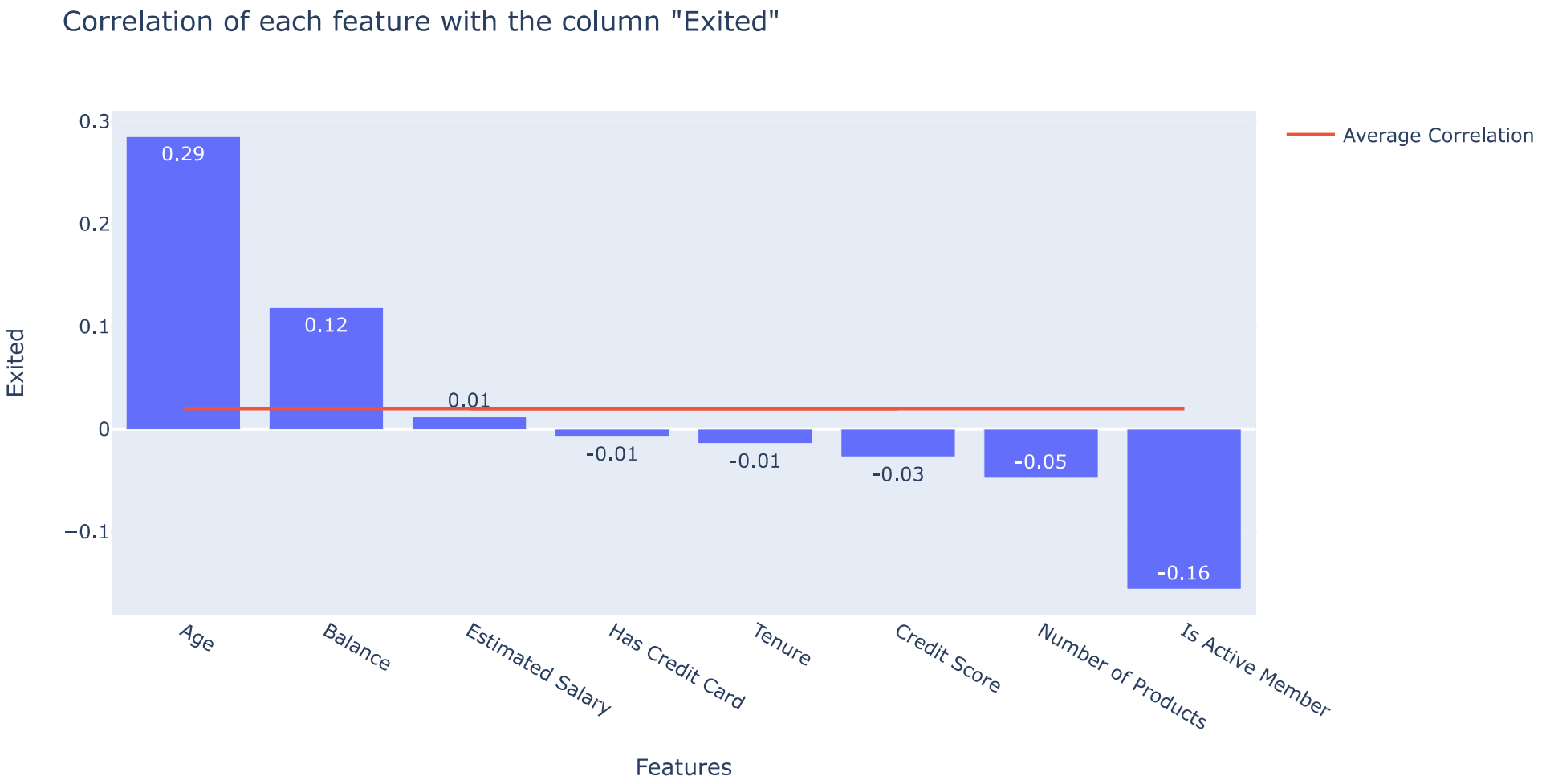
In [25]:
```
correlation['Average'] = average
```

In [26]:
```
correlation
```

Out[26]:

	index	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Average
0	Credit Score	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094	0.02
1	Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323	0.02
2	Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001	0.02
3	Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533	0.02
4	Number of Products	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820	0.02
5	Has Credit Card	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138	0.02
6	Is Active Member	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128	0.02
7	Estimated Salary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097	0.02

In [27]:
```
fig = px.bar(data_frame = correlation,x='index', y= 'Exited',title='Correlation of each feature with the column "Exited"', labels
fig.add_trace(go.Scatter(name='Average Correlation',x=correlation['index'],y=correlation['Average'],mode='lines'))

fig.show()
```



In [ ]:

Please save this notebook as a PDF containing your finished plots and submit them on the website by 24th November.

To save as a PDF, click on the '...' symbol on the same bar as '+ Markdown' and 'Run All' and click 'Export', then 'pdf'.

In [ ]:

In [ ]: