In [1]:

```
import pandas as pd
df=pd.read_csv("Churn_Modelling.csv") # import dataset
print(df)
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age  \
0             1    15634602   Hargrave          619    France  Female   42   
1             2    15647311       Hill          608     Spain  Female   41   
2             3    15619304       Onio          502    France  Female   42   
3             4    15701354       Boni          699    France  Female   39   
4             5    15737888   Mitchell          850     Spain  Female   43   
...         ...         ...        ...          ...       ...     ...  ...   
9995       9996    15606229   Obijiaku          771    France    Male   39   
9996       9997    15569892  Johnstone          516    France    Male   35   
9997       9998    15584532        Liu          709    France  Female   36   
9998       9999    15682355  Sabbatini          772   Germany    Male   42   
9999      10000    15628319     Walker          792    France  Female   28   

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1   
1          1   83807.86              1          0               1   
2          8  159660.80              3          1               0   
3          1       0.00              2          0               0   
4          2  125510.82              1          1               1   
...      ...        ...            ...        ...             ...   
9995       5       0.00              2          1               0   
9996      10   57369.61              1          1               1   
9997       7       0.00              1          0               1   
9998       3   75075.31              2          1               0   
9999       4  130142.79              1          1               0   

      EstimatedSalary  Exited  
0           101348.88       1  
1           112542.58       0  
2           113931.57       1  
3            93826.63       0  
4            79084.10       0  
...               ...     ...  
9995         96270.64       0  
9996        101699.77       0  
9997         42085.58       1  
9998         92888.52       1  
9999         38190.78       0  

[10000 rows x 14 columns]
```

In [30]:

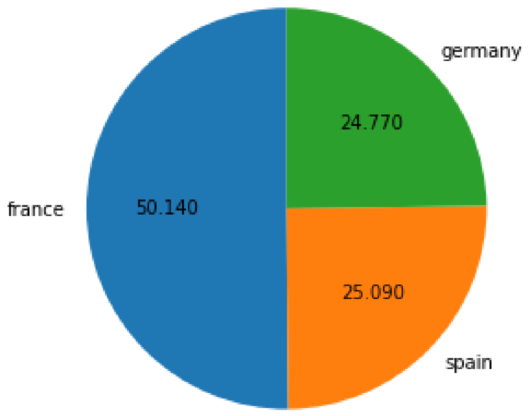
```
#@title 1. Univarient Analysis
```

In [32]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
plt.pie(df['Geography'].value_counts(),startangle=90,autopct='%.3f',labels=['france','sp
ain','germany'])
```

Out[32]:

```
([<matplotlib.patches.Wedge at 0x7faff9505490>,
  <matplotlib.patches.Wedge at 0x7faff9505bd0>,
  <matplotlib.patches.Wedge at 0x7faff950f490>],
 [Text(-1.0999893606763749, -0.004838015996287074, 'france'),
  Text(0.786805947043686, -0.7687238787085312, 'spain'),
  Text(0.7721769705773018, 0.7834173384027577, 'germany')],
 [Text(-0.599994196732568, -0.002638917816156585, '50.140'),
  Text(0.4291668802056469, -0.419303933841017, '25.090'),
  Text(0.42118743849671003, 0.427318548219686, '24.770')])
```



In [ ]:

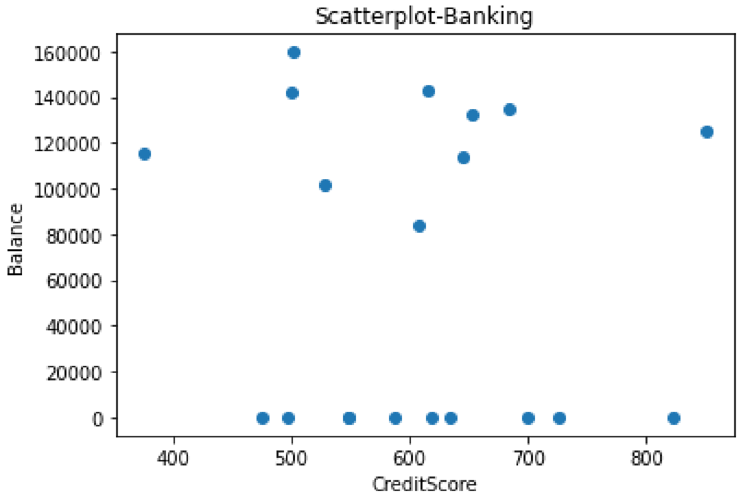
```
#@title 2. Bi - Variate Analysis
```

In [37]:

```
dfs1 = df.head(20)
plt.scatter(dfs1.CreditScore,dfs1.Balance)
plt.title('Scatterplot-Banking')
plt.xlabel("CreditScore")
plt.ylabel("Balance")
```

Out[37]:

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



In [10]:

```
import numpy as np
import seaborn as sns
```

In [28]:

```
#@title 3. Multi - Variate Analysis
```

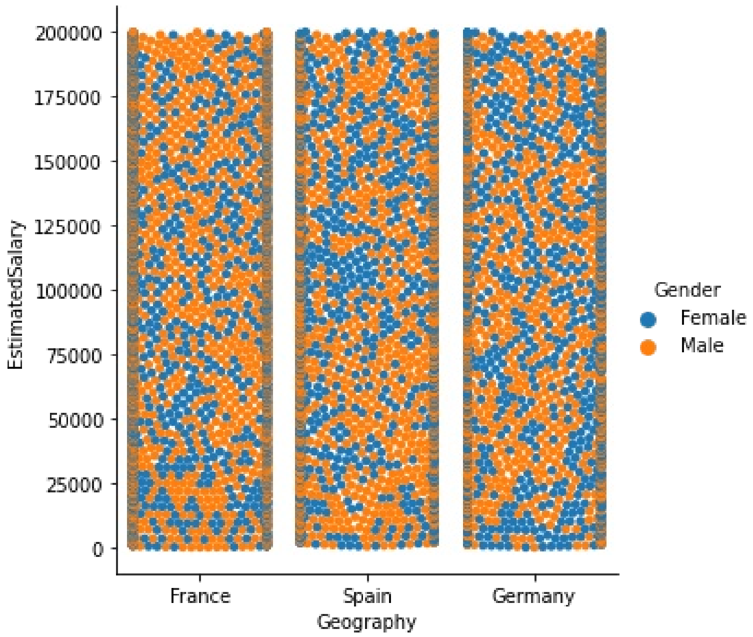
In [34]:

```
df=sns.catplot(x="Geography",y="EstimatedSalary",hue="Gender",kind="swarm",data=df)
print(df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 80.8% of
the points cannot be placed; you may want to decrease the size of the markers or use stri
pplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 62.1% of
the points cannot be placed; you may want to decrease the size of the markers or use stri
```

```
pplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 62.5% of
the points cannot be placed; you may want to decrease the size of the markers or use stri
pplot.
  warnings.warn(msg, UserWarning)
```

```
<seaborn.axisgrid.FacetGrid object at 0x7faff96fb610>
```



In [ ]:

```
#@title 4. Perform descriptive statistics on the dataset.
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [4]:

```
df.describe()
```

Out[4]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	

In [9]:

```
#@title 5. Handle the Missing values.
```

In [7]:

```
values={"RowNumber":0, "CustomerId":0, "CreditScore":0, "Age":0, "Tenure":0, "Balance":0
, "NumOfProducts":0, "HasCrCard":0, "IsActiveMember":0, "EstimatedSalary":0, "Exited":0}
df.fillna(value=values)
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasC
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	
...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	

10000 rows × 14 columns

In [ ]:

```
#@title 6. Find the outliers and replace the outliers
```

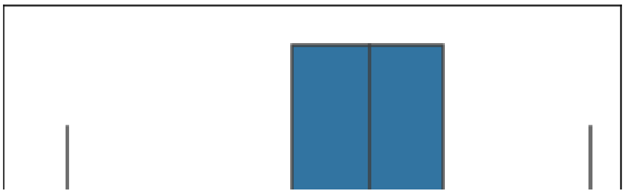
In [17]:

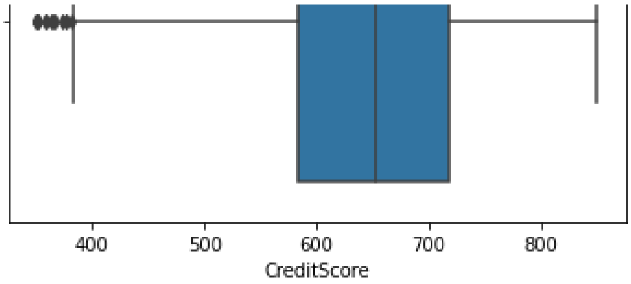
```
df= pd.read_csv("Churn_Modelling.csv")
sns.boxplot(df.CreditScore)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f518dc0c810>
```





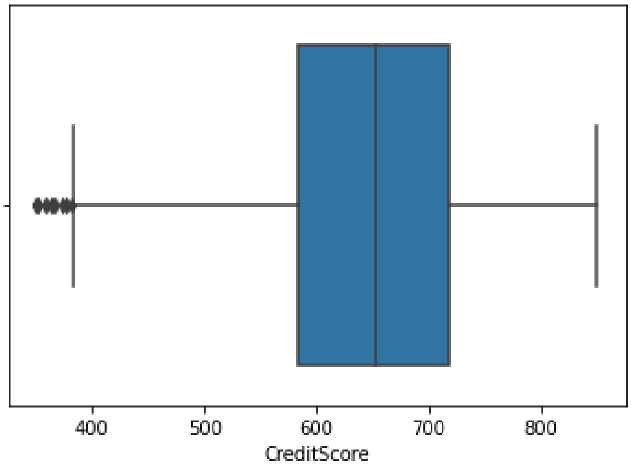
In [18]:

```
Q1= df.CreditScore.quantile(0.25)
Q3=df.CreditScore.quantile(0.75)
IQR=Q3-Q1
upper_limit =Q3 + 1.5*IQR
lower_limit =Q1 - 1.5*IQR
df['CreditScore'] = np.where(df['CreditScore']>upper_limit,30,df['CreditScore'])
sns.boxplot(df.CreditScore)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f518db93f10>
```



In [ ]:

```
#@title 7. Check for Categorical columns and perform encoding.
```

In [16]:

```
df.head(5)
```

Out[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCar
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 [22]:

```
#label encoding
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Gender= le.fit_transform(df.Gender)
df.head(20)
```

Out[22]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	1	15634602	Hargrave	619	France	0	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	0	41	1	83807.86	1	
2	3	15619304	Onio	502	France	0	42	8	159660.80	3	
3	4	15701354	Boni	699	France	0	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.82	1	
5	6	15574012	Chu	645	Spain	1	44	8	113755.78	2	
6	7	15592531	Bartlett	822	France	1	50	7	0.00	2	
7	8	15656148	Obinna	376	Germany	0	29	4	115046.74	4	
8	9	15792365	He	501	France	1	44	4	142051.07	2	
9	10	15592389	H?	684	France	1	27	2	134603.88	1	
10	11	15767821	Bearce	528	France	1	31	6	102016.72	2	
11	12	15737173	Andrews	497	Spain	1	24	3	0.00	2	
12	13	15632264	Kay	476	France	0	34	10	0.00	2	
13	14	15691483	Chin	549	France	0	25	5	0.00	2	
14	15	15600882	Scott	635	Spain	0	35	7	0.00	2	
15	16	15643966	Goforth	616	Germany	1	45	3	143129.41	2	
16	17	15737452	Romeo	653	Germany	1	58	1	132602.88	1	
17	18	15788218	Henderson	549	Spain	0	24	9	0.00	2	
18	19	15661507	Muldrow	587	Spain	1	45	6	0.00	1	
19	20	15568982	Hao	726	France	0	24	6	0.00	2	

In [24]:

```
#one hot encoding
df_main=pd.get_dummies(df,columns=['Geography'])
df_main.head()
```

Out[24]:

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

In [ ]:

```
#@title 8. Split the data into dependent and independent variables.
```

In [25]:

```
#Splitting the Dataset into the Independent Feature Matrix
df=pd.read_csv("Churn_Modelling.csv")
```

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

```
[[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 [26]:

```
#Extracting the Dataset to Get the Dependent Vector
Y = df.iloc[:, -1].values
print(Y)
```

```
[1 0 1 ... 1 1 0]
```

In [ ]:

```
#@title 9. Scale the independent variables
```

In [27]:

```
w = df.head()
q = w[['Age','Balance','EstimatedSalary']] #spliting the dataset into measureable values
q
```

Out[27]:

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

In [28]:

```
from sklearn.preprocessing import scale # library for scallling
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()

x_scaled = mm.fit_transform(q)
x_scaled
```

Out[28]:

```
array([[0.75      , 0.        , 0.63892099],
       [0.5       , 0.52491194, 0.96014087],
       [0.75      , 1.        , 1.        ],
       [0.        , 0.        , 0.42305883],
       [1.        , 0.78610918, 0.        ]])
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
sc =  StandardScaler()
x_ss = sc.fit_transform(q)
x_ss
```

Out[30]:

```
array([[ 0.44232587, -1.13763618,  0.09337626],
       [-0.29488391,  0.15434425,  0.96285595],
       [ 0.44232587,  1.32369179,  1.07074687],
       [-1.76930347, -1.13763618, -0.49092058],
       [ 1.17953565,  0.79723632, -1.6360585 ]])
```

In [32]:

```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame(scale(q),columns=q.columns)
X_scale=X_scaled.head()
X_scale
```

Out[32]:

	Age	Balance	EstimatedSalary
0	0.442326	-1.137636	0.093376
1	-0.294884	0.154344	0.962856
2	0.442326	1.323692	1.070747
3	-1.769303	-1.137636	-0.490921
4	1.179536	0.797236	-1.636059

In [ ]:

```
#@title 10. Split the data into training and testing
```

In [33]:

```
x= df[['Age','Balance','EstimatedSalary']]
x
```

Out[33]:

	Age	Balance	EstimatedSalary
0	42	0.00	101348.88
1	41	83807.86	112542.58
2	42	159660.80	113931.57
3	39	0.00	93826.63
4	43	125510.82	79084.10
...	...	...	...
9995	39	0.00	96270.64
9996	35	57369.61	101699.77
9997	36	0.00	42085.58
9998	42	75075.31	92888.52
9999	28	130142.79	38190.78

10000 rows × 3 columns

In [34]:

```
y = df['Balance']
y
```

Out[34]:

```
0            0.00
1        83807.86
2       159660.80
3            0.00
4       125510.82
          ...
9995         0.00
9996     57369.61
9997         0.00
9998     75075.31
9999    130142.79
```

```
9999    150112.79
Name: Balance, Length: 10000, dtype: float64
```

In [35]:

```
#scaling
from sklearn.preprocessing import StandardScaler, MinMaxScaler
sc = StandardScaler()
x_scaled1 = sc.fit_transform(x)
x_scaled1
```

Out[35]:

```
array([[ 0.29351742, -1.22584767,  0.02188649],
       [ 0.19816383,  0.11735002,  0.21653375],
       [ 0.29351742,  1.33305335,  0.2406869 ],
       ...,
       [-0.27860412, -1.22584767, -1.00864308],
       [ 0.29351742, -0.02260751, -0.12523071],
       [-1.04143285,  0.85996499, -1.07636976]])
```

In [36]:

```
#train and test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled1, y, test_size = 0.3, rando
m_state = 0)
```

In [37]:

```
x_train
```

Out[37]:

```
array([[-0.56466489,  1.11721307, -0.77021814],
       [ 0.00745665, -1.22584767, -1.39576675],
       [ 3.53553951,  1.35419118, -1.49965629],
       ...,
       [-0.37395771,  1.35890908,  1.41441489],
       [-0.08789694, -1.22584767,  0.84614739],
       [ 0.86563897,  0.50630343,  0.32630495]])
```

In [38]:

```
x_train.shape
```

Out[38]:

```
(7000, 3)
```

In [39]:

```
x_test
```

Out[39]:

```
array([[-0.37395771,  0.87532296,  1.61304597],
       [ 0.10281024,  0.42442221,  0.49753166],
       [ 0.29351742,  0.30292727, -0.4235611 ],
       ...,
       [ 0.10281024,  1.46672809,  1.17045451],
       [ 2.86806437,  1.25761599, -0.50846777],
       [ 0.96099256,  0.19777742, -1.15342685]])
```

In [40]:

```
x_test.shape
```

Out[40]:

```
(3000, 3)
```

In [41]:

```
y_train
```

Out[41]:

```
7681    146193.60
9031         0.00
3691    160979.68
202          0.00
5625    143262.04
          ...
9225    120074.97
4859    114440.24
3264    161274.05
9845         0.00
2732    108076.33
Name: Balance, Length: 7000, dtype: float64
```

In [42]:

```
y_test
```

Out[42]:

```
9394    131101.04
898     102967.41
2398     95386.82
5906    112079.58
2343    163034.82
          ...
4004         0.00
7375     80926.02
9307    168001.34
8394    154953.94
5233     88826.07
Name: Balance, Length: 3000, dtype: float64
```

In [ ]:

```

```