If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.model_selection import train_test_split

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
df=pd.read_csv('/content/drive/MyDrive/Churn_Modelling.csv')
df=df.iloc[:,3:]
df
```

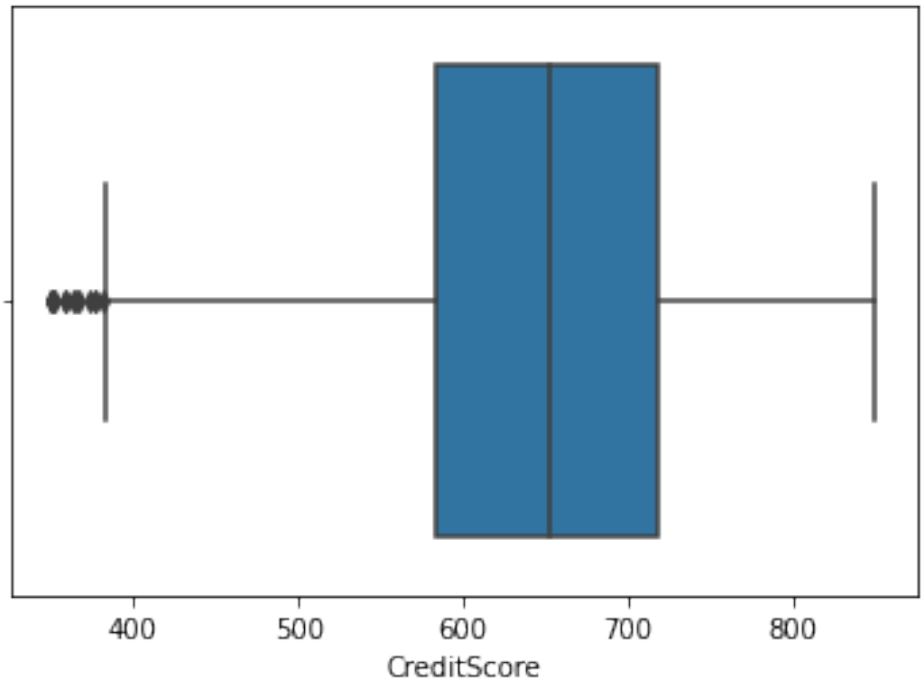
```
      CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0             619    France  Female   42       2       0.00
1
1             608     Spain  Female   41       1   83807.86
1
2             502    France  Female   42       8  159660.80
3
3             699    France  Female   39       1       0.00
2
4             850     Spain  Female   43       2  125510.82
```

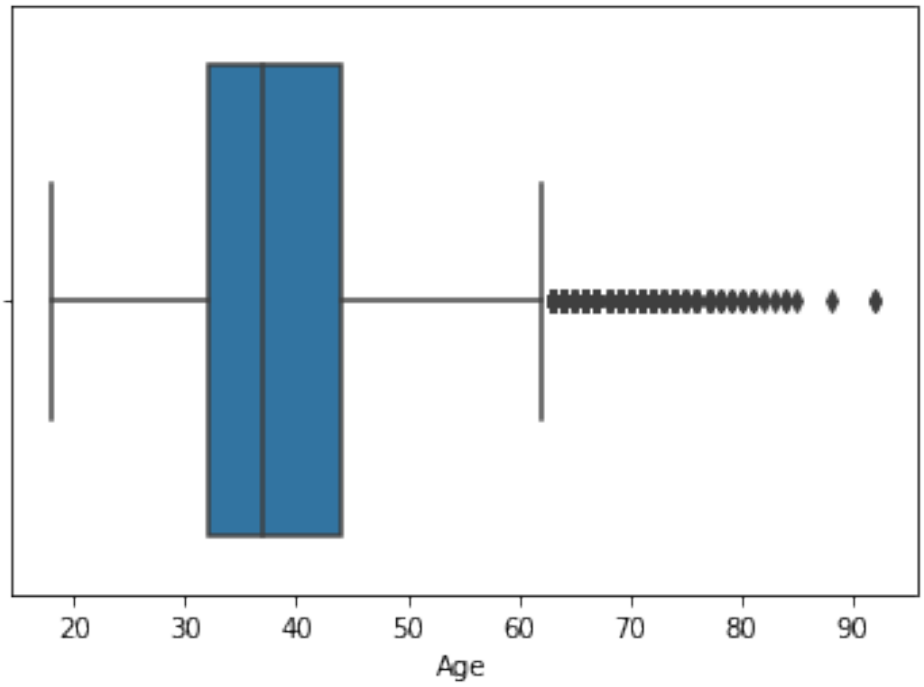
```
1
...           ...       ...     ...  ...     ...        ...
...
9995          771    France    Male   39       5       0.00
2
9996          516    France    Male   35      10   57369.61
1
9997          709    France  Female   36       7       0.00
1
9998          772   Germany    Male   42       3   75075.31
2
9999          792    France  Female   28       4  130142.79
1

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

[10000 rows x 11 columns]
```

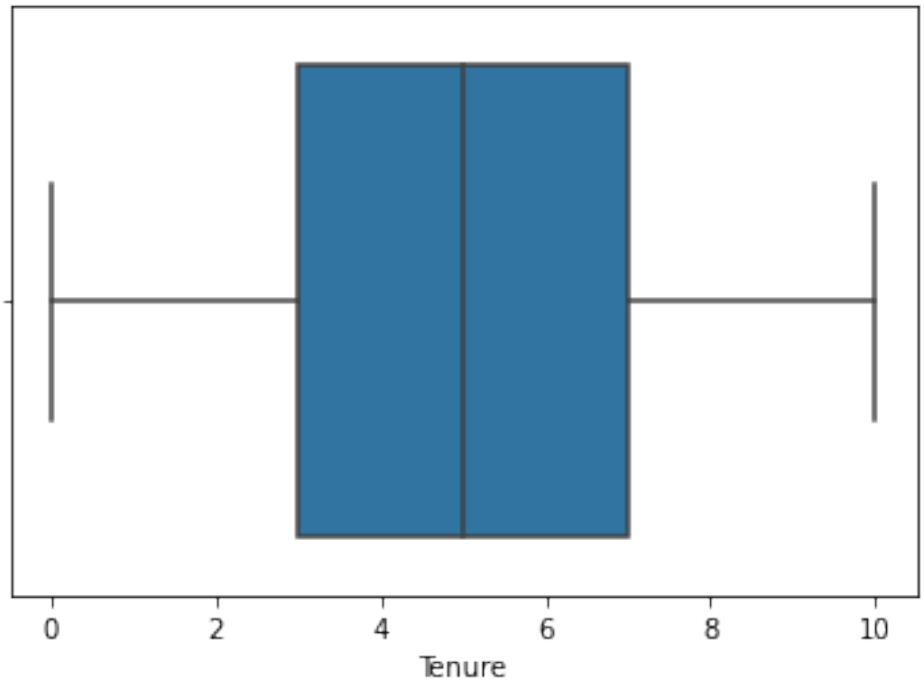
To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

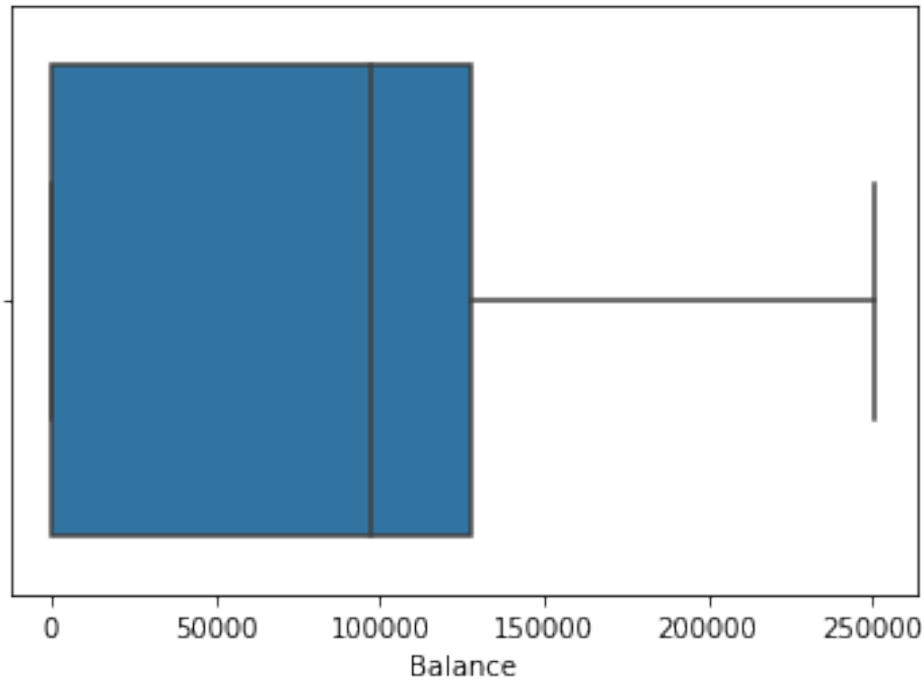
Variables that you define in one cell can later be used in other cells:

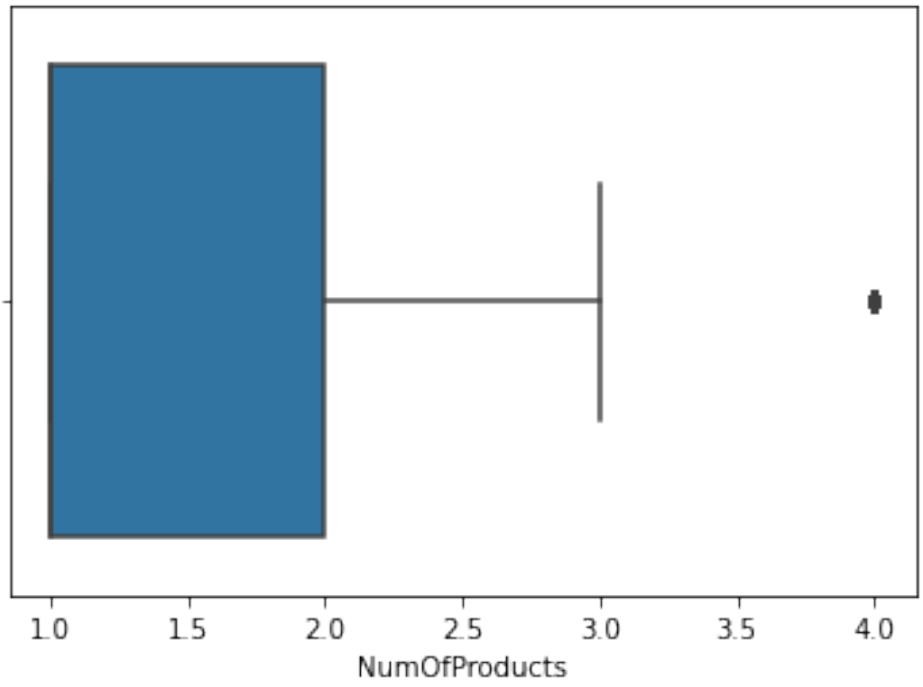
```
for col in df.columns:
  if(df.dtypes[col]=='int64' or df.dtypes[col]=='float64'):
    sns.boxplot(x=df[col]).set(xlabel=col)
    plt.show()
```

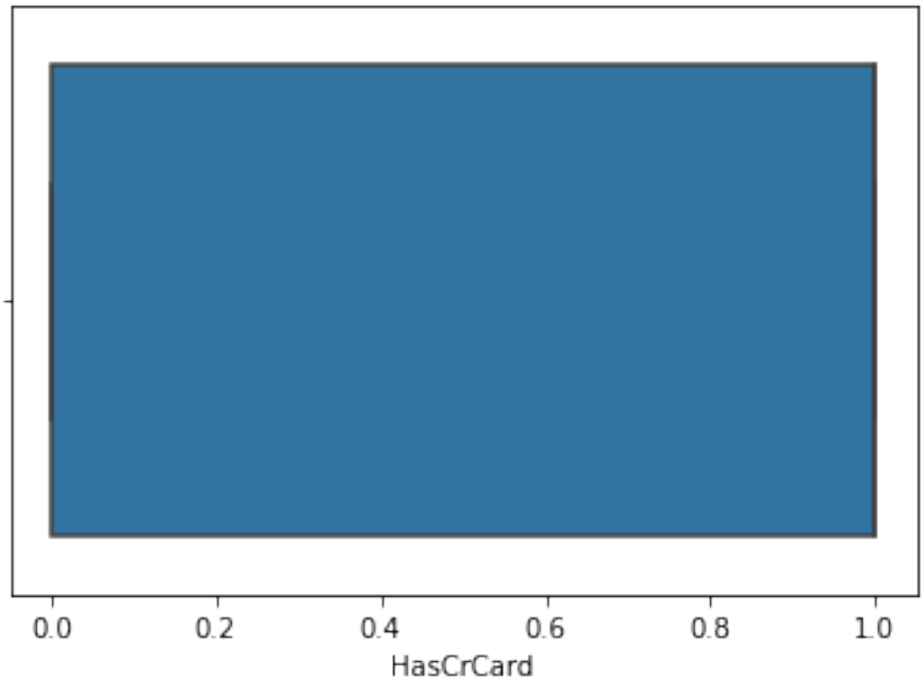

400
500
600
700
800
CreditScore

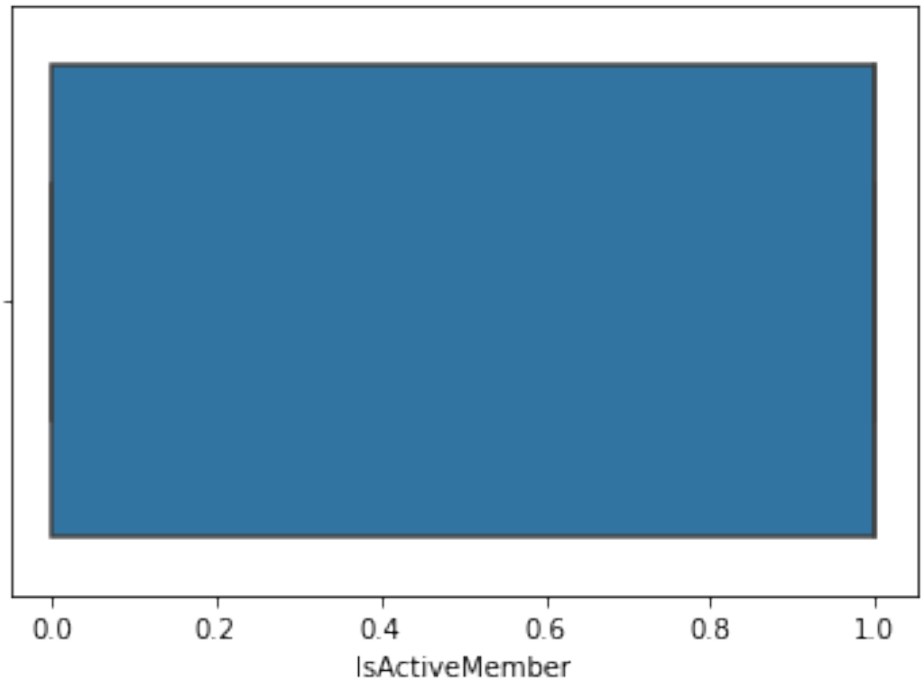

20
30
40
50
60
70
80
90
Age

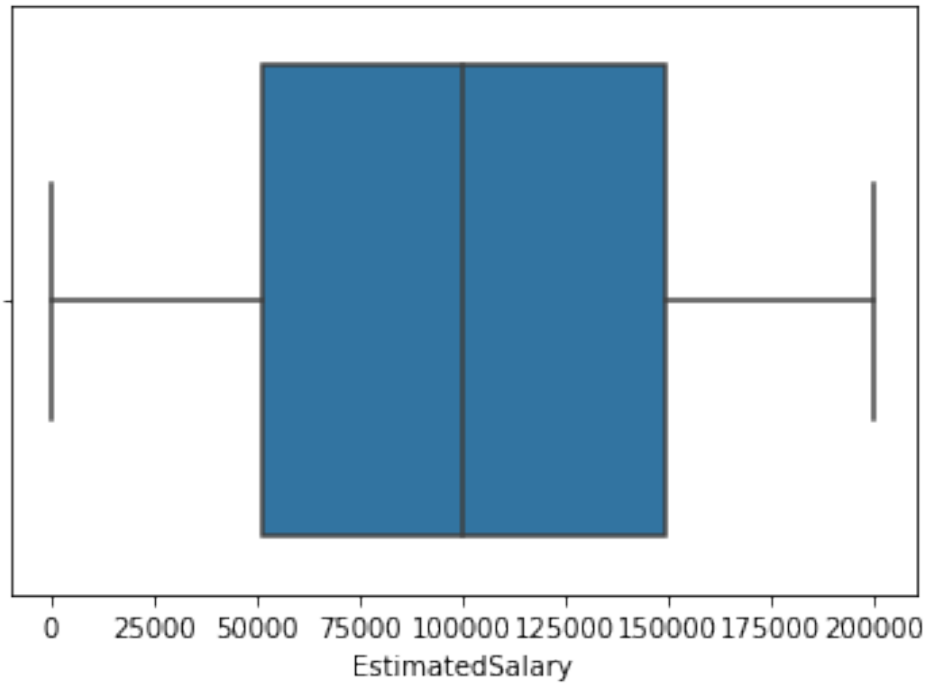

0
2
4
6
8
10
Tenure

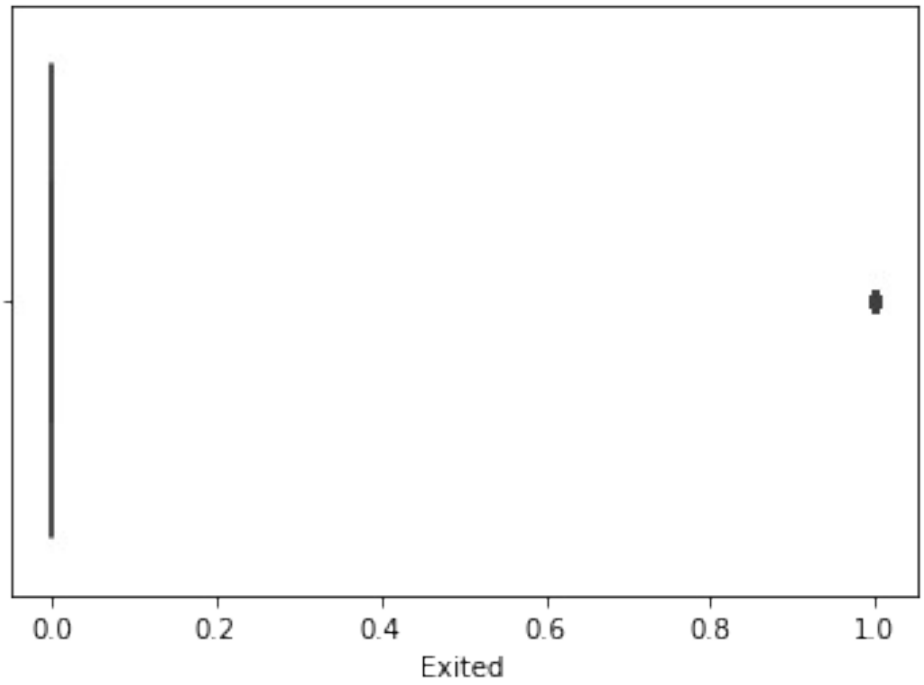

0
50000
100000
150000
200000
250000
Balance


1.0
1.5
2.0
2.5
3.0
3.5
4.0
NumOfProducts

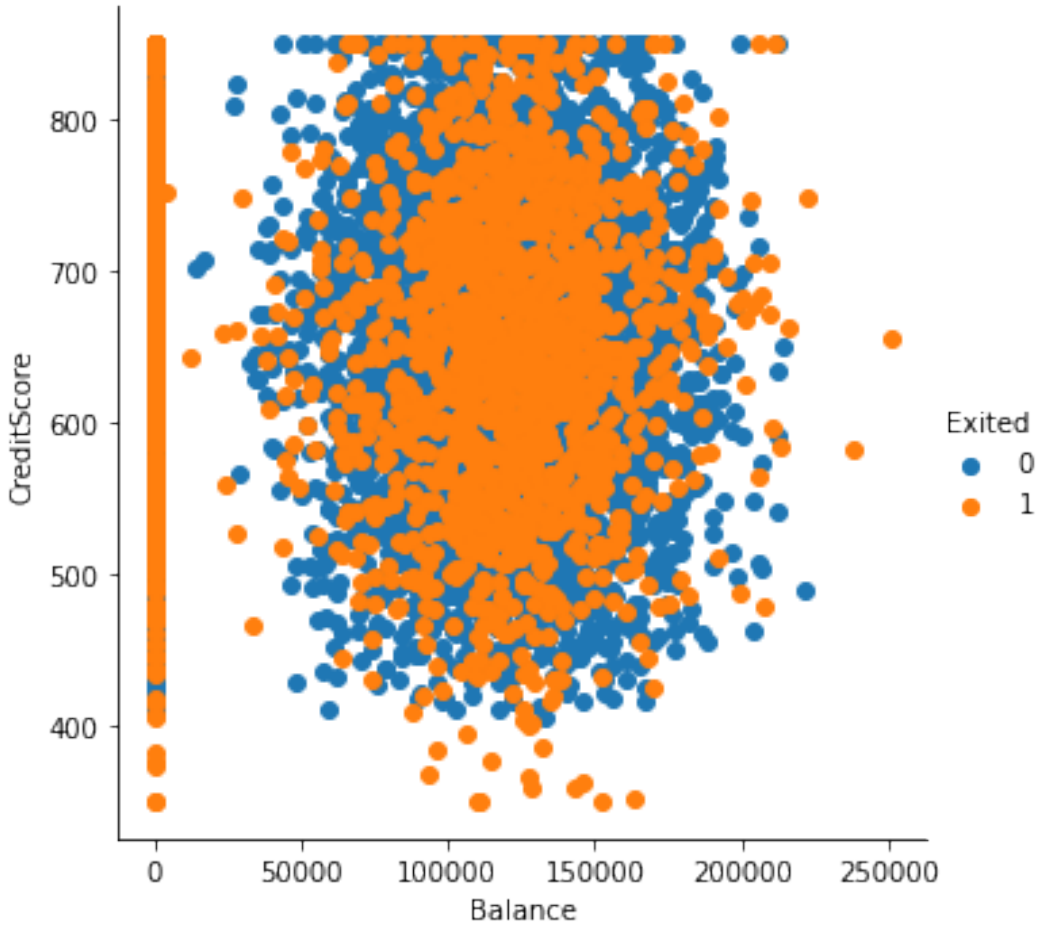

0.0
0.2
0.4
0.6
0.8
1.0
HasCrCard


0.0
0.2
0.4
0.6
0.8
1.0
IsActiveMember


0
25000
50000
75000
100000
125000
150000
175000
200000
EstimatedSalary

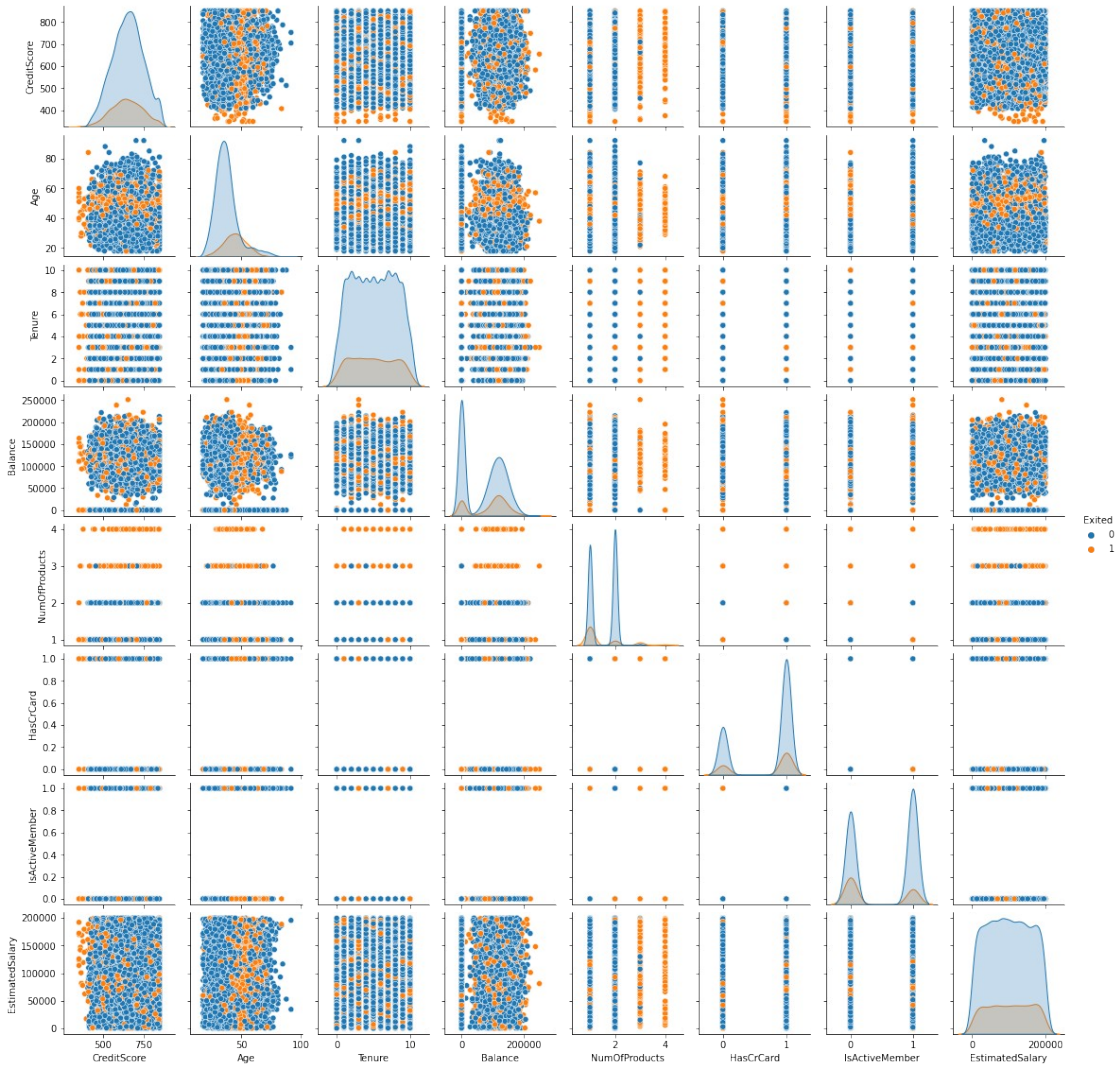



```
sns.FacetGrid(df,hue='Exited',height=5).map(plt.scatter,"Balance","Cre
ditScore").add_legend()
plt.show()
```




```
sns.pairplot(df, hue='Exited', height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7f31ed64f2d0>
```




```
df.describe()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200
std	96.653299	10.487806	2.892174	62397.405202	0.581654
min	350.000000	18.000000	0.000000	0.000000	1.000000
25%	584.000000	32.000000	3.000000	0.000000	1.000000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000

```
max      850.000000     92.000000     10.000000  250898.090000
4.000000

         HasCrCard  IsActiveMember  EstimatedSalary        Exited
count  10000.00000    10000.000000     10000.000000  10000.000000
mean       0.70550        0.515100    100090.239881      0.203700
std        0.45584        0.499797     57510.492818      0.402769
min        0.00000        0.000000        11.580000      0.000000
25%        0.00000        0.000000     51002.110000      0.000000
50%        1.00000        1.000000    100193.915000      0.000000
75%        1.00000        1.000000    149388.247500      0.000000
max        1.00000        1.000000    199992.480000      1.000000
```

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

```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
CreditsMedian=df.loc[df['CreditScore']<400,'CreditScore'].median()
ProdMedian=df.loc[df['NumOfProducts']>=3.5,'NumOfProducts'].median()
df.loc[df.CreditScore<400,'CreditScore']=np.nan
df.fillna(CreditsMedian,inplace=True)
df.loc[df.NumOfProducts>3,'NumOfProducts']=np.nan
df.fillna(ProdMedian,inplace=True)

labelencoder=LabelEncoder()
df['Geography']=labelencoder.fit_transform(df['Geography'])
df['Gender']=labelencoder.fit_transform(df['Gender'])

ind=df.iloc[:,:-1]
dep=df.iloc[:,-1:]

nm=MinMaxScaler()
N_ind=nm.fit_transform(ind)

xtrain,xtest,ytrain,ytest=train_test_split(N_ind,dep,test_size=0.3)
print(xtrain,xtest,ytrain,ytest)
```

```
[[0.44536082 1.         0.         ... 1.         0.
0.52135209]
 [0.53402062 1.         0.         ... 0.         1.
0.98337856]
```

```
 [0.60412371 1.         1.         ... 1.         0.
0.26065564]
 ...
 [0.41237113 0.5        1.         ... 1.         0.
0.52820469]
 [0.1443299  0.5        1.         ... 1.         0.
0.83732956]
 [0.62680412 0.         1.         ... 1.         1.
0.85790173]] [[0.67010309 1.         1.         ... 0.         1.
0.2550229 ]
 [0.96907216 0.         0.         ... 1.         1.
0.19906176]
 [0.38350515 0.5        0.         ... 1.         0.
0.72899417]
 ...
 [0.65360825 0.5        0.         ... 1.         1.
0.32336948]
 [0.58969072 0.5        1.         ... 0.         0.
0.96802425]
 [0.50927835 0.         1.         ... 1.         0.
0.82042755]]       Exited
3934       0
1072       0
2290       0
5344       0
2747       0
...      ...
6345       0
6207       0
3344       0
9071       0
5248       0

[7000 rows x 1 columns]       Exited
6202       0
9465       0
2735       0
6494       0
9481       0
...      ...
504        0
2119       0
9703       0
4923       0
8370       0

[3000 rows x 1 columns]
```

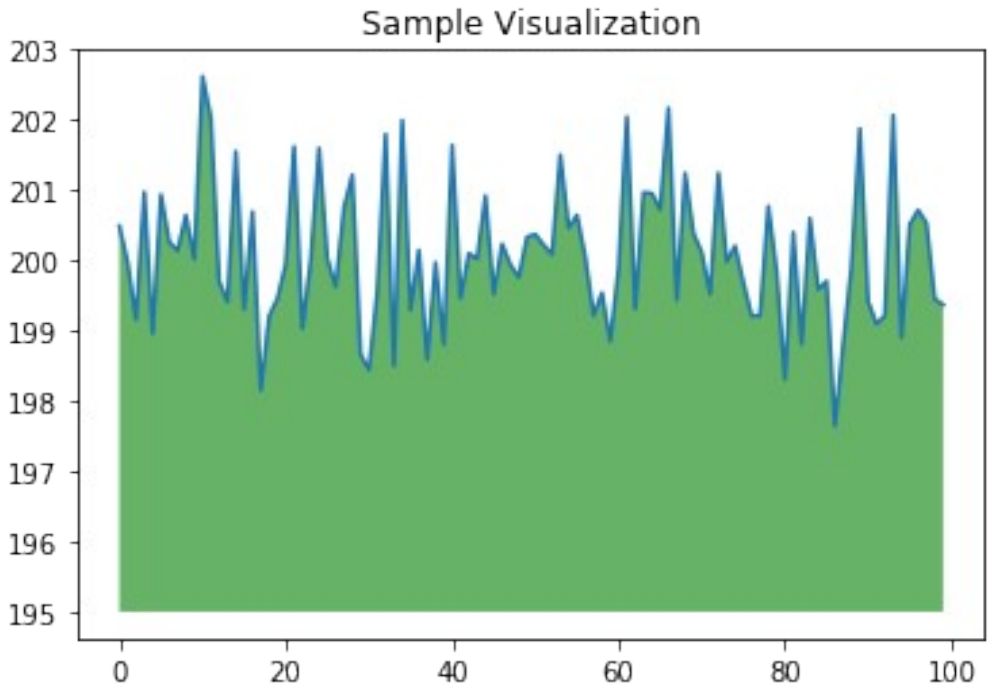
Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab

notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g',
alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```



You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn

more about importing data, and how Colab can be used for data science, see the links below under Working with Data.

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

- Overview of Colaboratory
- Guide to Markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets
-
- Loading data: Drive, Sheets, and Google Cloud Storage
- Charts: visualizing data
- Getting started with BigQuery

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Linear regression with tf.keras using synthetic data
- TensorFlow with GPUs
- TensorFlow with TPUs
- NeMo Voice Swap: Use Nvidia's NeMo conversational AI Toolkit to swap a voice in an audio fragment with a computer generated one.
- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.

- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.