This Part, we will analyze data from a Banking Sector, The data shows details of Bank Members such as Gender, LoanAmount, Education, ApplicantIncome, LoanAmount among other details. We will analyze this data to get some insights form the Bank. The Currency is in USD.

**What is a Dataset?**

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. i.e Bank data members.

**Exploratory data analysis (EDA)** is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

In Below data , Gender, Marital Status, Education, Self_Employed,m Property Area and Loan Status are Categorical Variables.
Coapplicant Income, Loan Amount, Applicant Income are Continous Variable.

**Data Descriptions:**

Loan_ID: This is a Unique Loan ID for each member.(Unique)

Gender: This shows the Gender of Members. Male or Female(Categorical - Nominal)

Married: Shows marital Status of the Bank Members. Married Yes/No- (Categorical - Nominal)

Self_Employed: Show Self-Employement status of members. Self Employed: Yes/No - (Categorical - Nominal)

ApplicantIncome: This is the Income the member earns - Monthly (Continous)

CoapplicantIncome: Amount coapplicants Earns Monthly(Continous)

LoanAmount: Loan Amount taken by the member.(Continous)

Property_Area: Are Where members plan to invest in a given property. (Categorical - Nominal)

Loan_Status: Status of Loan : Paid Yes- Y or Not Paid No - N. (Categorical - Ordinal)

##### What is a Dataframe?

A dataframe is a data structure constructed with rows and columns, similar to a database tables or Excel spreadsheet.

**What is Pandas?**
pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,built on top of the Python programming language.
Pandas Link: https://pandas.pydata.org/
In below code snipet, we read a dataset using pandas and store all data in dataframe named data.

**read_csv** is a pandas function to read a CSV File.

```
import pandas
# data is our dataframe.
data = pandas.read_csv("https://coding.co.ke/datasets/bank.csv")
data
```

| | Loan_ID | Gender | Married | Education | Self_Employed | ApplicantIncome | Co |
|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | LP002978 | Female | No | Graduate | No | 2900 | |
| **610** | LP002979 | Male | Yes | Graduate | No | 4106 | |
| **611** | LP002983 | Male | Yes | Graduate | No | 8072 | |
| **612** | LP002984 | Male | Yes | Graduate | No | 7583 | |
| **613** | LP002990 | Female | No | Graduate | Yes | 4583 | |

Below we can access columns in our dataframe by taking the columns names as keys in data frame - data.

```
# access our data frame and provide column name as KEY.
data['Gender']
# We can do the same for other columns
```

```
0        Male
1        Male
2        Male
3        Male
4        Male
        ...
609    Female
610      Male
611      Male
612      Male
613    Female
Name: Gender, Length: 614, dtype: object
```

Below, before proceeding we check if we have empty records in our dataset. We can access our data in our dataframe - data

```
data.isnull().sum()
```

```
Loan_ID               0
Gender                0
Married               0
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Property_Area         0
Loan_Status           0
dtype: int64
```

We can see from above output we have some missing records, example Self_Employed is missing 32 records and LoanAmount is mising 22 records. In Data Science, if we have missing records, We have to do a process called data cleaning.

**Data cleaning,** also referred to as data cleansing and data scrubbing, is one of the most important steps for your organization if you want to create a culture around quality data in decision-making.

We will clean up the empty records as follows.

- For Self_Employed currently the data we have Yes or No, this is a categorical variable, For the empty data we can put a Categorical variable to fill the empty spots, we will put Unknown or Neutral, Since we do not know the Self Employed Status for the missing Records. See below

```
data['Self_Employed'].fillna('Unknown',inplace=True)
```

**fillna** is used to fill empties and inplace=True is used to update the empty record with 'Unknown'. Below we print the data again to see if the empty in 'Self_Employed' is Updated to 'Unknown'.

```
data.head(50)
```

| | | | | | | |
|---|---|---|---|---|---|---|
| **18** | LP001038 | Male | Yes | Graduate | No | 4887 |
| **19** | LP001041 | Male | Yes | Graduate | Unknown | 2600 |
| **20** | LP001043 | Male | Yes | Not Graduate | No | 7660 |
| **21** | LP001046 | Male | Yes | Graduate | No | 5955 |
| **22** | LP001047 | Male | Yes | Not Graduate | No | 2600 |
| **23** | LP001050 | Female | Yes | Not Graduate | No | 3365 |
| **24** | LP001052 | Male | Yes | Graduate | Unknown | 3717 |
| **25** | LP001066 | Male | Yes | Graduate | Yes | 9560 |
| **26** | LP001068 | Male | Yes | Graduate | No | 2799 |
| **27** | LP001073 | Male | Yes | Not Graduate | No | 4226 |
| **28** | LP001086 | Male | No | Not Graduate | No | 1442 |
| **29** | LP001087 | Female | No | Graduate | Unknown | 3750 |
| **30** | LP001091 | Male | Yes | Graduate | Unknown | 4166 |
| **31** | LP001095 | Male | No | Graduate | No | 3167 |
| **32** | LP001097 | Male | No | Graduate | Yes | 4692 |
| **33** | LP001098 | Male | Yes | Graduate | No | 3500 |
| **34** | LP001100 | Male | No | Graduate | No | 12500 |
| **35** | LP001106 | Male | Yes | Graduate | No | 2275 |
| **36** | LP001109 | Male | Yes | Graduate | No | 1828 |
| **37** | LP001112 | Female | Yes | Graduate | No | 3667 |
| **38** | LP001114 | Male | No | Graduate | No | 4166 |
| **39** | LP001116 | Male | No | Not Graduate | No | 3748 |
| **40** | LP001119 | Male | No | Graduate | No | 3600 |
| **41** | LP001120 | Male | No | Graduate | No | 1800 |
| **42** | LP001123 | Male | Yes | Graduate | No | 2400 |
| **43** | LP001131 | Male | Yes | Graduate | No | 3941 |
| **44** | LP001136 | Male | Yes | Not Graduate | Yes | 4695 |
| **45** | LP001137 | Female | No | Graduate | No | 3410 |

Above we print the first 50 records using head(50), by looking on Self_Employed we see its updated with unknown. Next we fill the LoanAmount, The LoanAmount is a Continous variable, hence we must fill it with a continous variable, For LoanAnount we fill the empty records with the average/mean of LoanAmount, Find the Loan Amount mean with below code.

```
# Access Loan Amount in our data frame.
# You can also change mean() to median(), mode(), std() etc
mean = data['LoanAmount'].mean()
mean
```

```
    146.41216216216216
```

Above find mean() of LoanAmount and store it in a variable mean, the mean is 146.4, Now fill the Loan Amount empties with that mean. Below is the code

```
data['LoanAmount'].fillna(mean, inplace=True)
```

Having filled Empty for Self Employed and LoanAmount, Check if we have any other empty in our data.

```
data.isnull().sum()
```

```
    Loan_ID             0
    Gender              0
    Married             0
    Education           0
    Self_Employed       0
    ApplicantIncome     0
    CoapplicantIncome   0
    LoanAmount          0
    Property_Area       0
    Loan_Status         0
    dtype: int64
```

**Great!, Above we do not have any empty record.**

Lets describe our data, we get basic analysis in our data.Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics. Lets try Answer the following questions?

1. How many record do we have (count?)
2. What is the average ApplicantIncome?
3. What is the minimum ApplicantIncome?
4. What is the Highest Loan Amount Taken?
5. What are data 25%, 50%, 75% Percentiles?

```
data.describe()
```

| | ApplicantIncome | CoapplicantIncome | LoanAmount |
|---|---|---|---|
| **count** | 614.000000 | 614.000000 | 614.000000 |
| **mean** | 5403.459283 | 1621.245798 | 146.412162 |
| **std** | 6109.041673 | 2926.248369 | 84.037468 |
| **min** | 150.000000 | 0.000000 | 9.000000 |
| **25%** | 2877.500000 | 0.000000 | 100.250000 |
| **50%** | 3812.500000 | 1188.500000 | 129.000000 |
| **75%** | 5795.000000 | 2297.250000 | 164.750000 |
| **max** | 81000.000000 | 41667.000000 | 700.000000 |

Correlate data, this helps you see how each variable correates with each either Positive or Negative.

We can now answer our questions from above output.

1. How many record do we have (count?)

   From the count we have 614 Records

2. What is the average ApplicantIncome?

   The Average Applicant Income is 5403.45

3. What is the minimum ApplicantIncome?

   The minumum Applicant Income is 150.00

4. What is the Highest Loan Amount Taken?

   The highest Loan Amount take in this Bank is 700 USD

5. What are data 25%, 50%, 75% Percentiles?

   Loking at Loan Amount, the 25th Pecntile is 100, The 50th is 129 and the 75th is 164, The 100th Percentile is the maximum Loan Amount which 700 USD

Next, we can see the data correlations with corr() function A negative correlation is a relationship between two variables that move in opposite directions. In other words, when variable A increases, variable B decreases.

A positive correlation is a relationship between two variables that move in tandem—that is, in the same direction. One goes up, the other goes up. Lets see for example correlation between Loan Amount and Applicant Income.

```
data.corr()
```

```
<ipython-input-10-c44ded798807>:1: FutureWarning: The default value of nume
  data.corr()
```

| | ApplicantIncome | CoapplicantIncome | LoanAmount |
|---|---|---|---|
| **ApplicantIncome** | 1.000000 | -0.116605 | 0.565620 |
| **CoapplicantIncome** | -0.116605 | 1.000000 | 0.187828 |
| **LoanAmount** | 0.565620 | 0.187828 | 1.000000 |

Above we see the Correlation between Loan Amount and Applicant Income is 0.5. What does this mean?

Pos0.1 - Pos1.0 means Positive Correlation. The Closer to 1, The Stronger Positive it is,

Neg-0.1 - Neg-1.0 means Negative Correlation, The Closer to -1, The Stronger Negative it is

0.0 - Means No Correlation.

So how about 0.5? Its a Strong Positive Correlation. This means when Loan Amount goes up , Applicant Income also goes Up. 0.5 means only 50% of Records. Also to Note, If correlation was 0.8, we could say its Stronger Positive Coreelation meaning 80% of Records are Positvely Correlated.

Lets check out columns/dimensions data types.

```
data.dtypes
```

```
Loan_ID              object
Gender               object
Married              object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome    float64
LoanAmount           float64
Property_Area        object
Loan_Status          object
dtype: object
```

Lets find average LoanAmount, You can change to

- median() for median.
- std() for standard deviation
- mode() For mode Below we use mean() - median.

```
data['LoanAmount'].mean()
```

```
146.41216216216216
```

To find the proportionality of Categorical variables i.e How many Male vs Female.
We can use the groupby() function.

```
# Access the Gender column, group by and show sizes.
data.groupby('Gender').size()
```

```
Gender
Female    125
Male      489
dtype: int64
```

We can conclude that we 125 Female and 489 Male, Male are 2 times more than female.

```
# Access the Gender and Married column, group by and show sizes.
data.groupby(['Gender', 'Married']).size()
```

```
Gender  Married
Female  No          83
        Yes         42
Male    No         130
        Yes        359
dtype: int64
```

Above, its clear that Male that were Married have are the Majority totaling to 359 compared to Female that were Married which is 42.

```
# Access the Gender and Married column, group by and show sizes.
data.groupby(['Gender', 'Married'])['LoanAmount'].mean()
```

```
Gender  Married
Female  No         118.455715
        Yes        169.724099
Male    No         136.485083
        Yes        153.743093
Name: LoanAmount, dtype: float64
```

In Above Output, we see that Female who were Married have the Highest LoanAmount on Average, Observe 169.72, Followed by Male who were Married at 153.74 Loan Amount on Average.

Now we can create graphs, we use seaborn and matplotlib.

**What is matplotlib?**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations. https://matplotlib.org/

**What is seaborn?**

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.
https://seaborn.pydata.org/

import seaborn as sns

import matplotlib.pyplot as plt

First Install seaborn and matplotlib use below commands

pip3 install seaborn

pip3 install matplotlib

After installation Check matplotlib themes available

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

Use one from the List

```
plt.style.use("seaborn")
```

```
<ipython-input-17-4b92c22464bb>:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated sir
  plt.style.use("seaborn")
```

1. Creating Graphs. We will create graphs using Seaborn. Below we create a Count Plot To show the propoprtion of Gender. Count plot show the counts of observations in each categorical bin using bars
   A Count Plot, We can see below that the Male are 3 Time more than Female members. A Count plot is a univariate Plot - meaning we have only one Variable(Gender) used in the Plot.

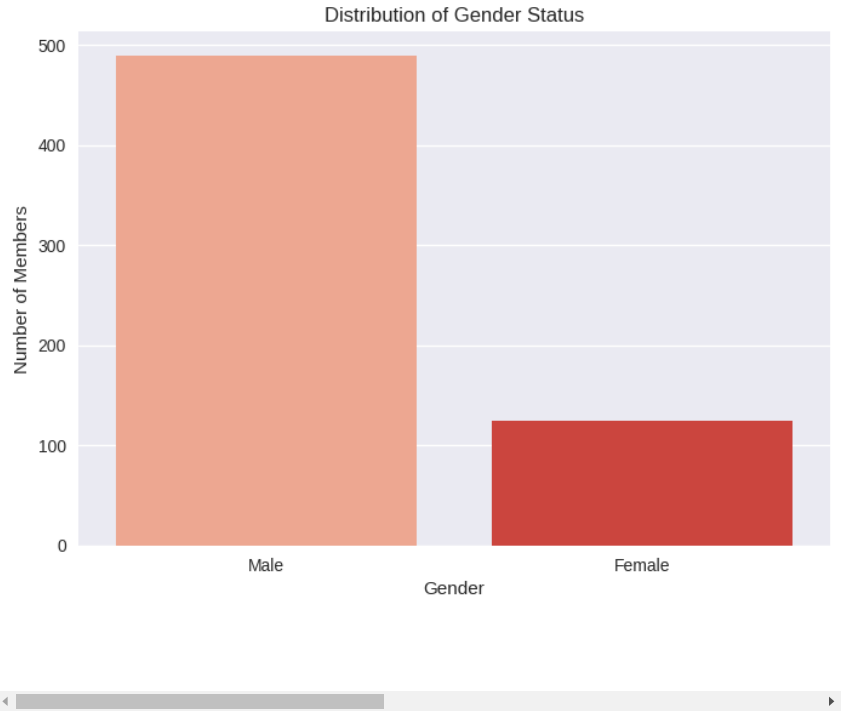
Pallete can be Oranges, Blues, Reds, magma, flare, crest

```
sns.countplot(x = data['Gender'], palette = 'Reds')
plt.title("Distribution of Gender Status")
plt.xlabel('Gender')
plt.ylabel('Number of Members')
```

```
<ipython-input-18-12bb48d71724>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removec

  sns.countplot(x = data['Gender'], palette = 'Reds')
Text(0, 0.5, 'Number of Members')
```

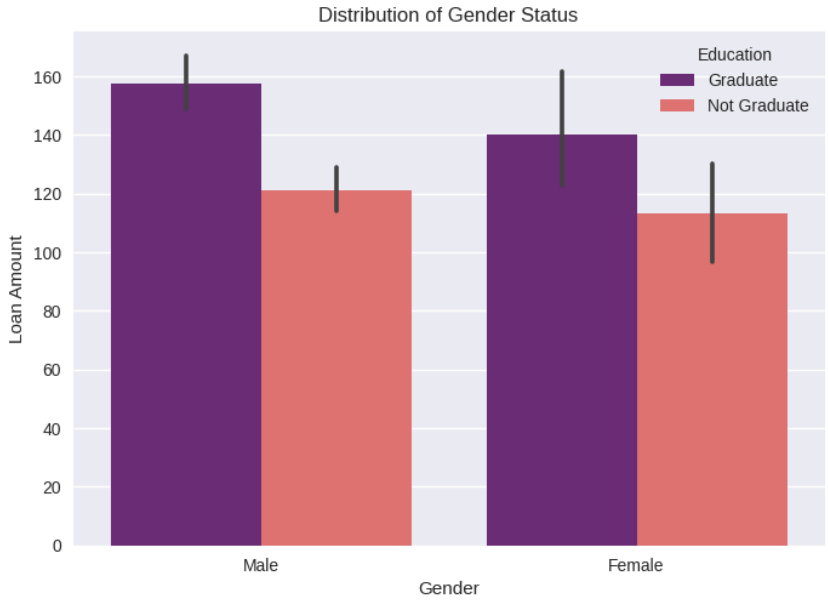



2. We create a Bar Plot. It shows the relationship between a numeric and a categoric variable. Each entity of the categoric variable is represented as a bar.
   Below we see that Male who are Graduates account for more Loan Amount of Average, Followed By Graduate Female. A Bar plot is a bivariate since we can have 2 variables, it can also be a multivariate plot when we have 3 or more variables.

```
import numpy
sns.barplot(x = 'Gender', y = 'LoanAmount', data = data, estimator=numpy.mean,
            palette = 'magma', hue='Education')

plt.title("Distribution of Gender Status")
plt.xlabel('Gender')
plt.ylabel('Loan Amount')
```

```
Text(0, 0.5, 'Loan Amount')
```




3. We create a Swarm Plot. A swarmplot is a type of categorical scatter plot used to visualize the distribution of data points in a dataset.
   Below we see that most applicant income for both Male and Female is below 20000 USD.
   A Bar plot is a bivariate since we can have 2 variables, it can also be a multivariate plot when we have 3 or more variables.

```
# Swarm
colors = ['#3949AB', '#2E7D32', '#546E7A']
sns.swarmplot(x ='Gender', y ='ApplicantIncome', data = data, palette= 'magma',
              hue ='Self_Employed')
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWa
  warnings.warn(msg, UserWarning)
<Axes: xlabel='Gender', ylabel='ApplicantIncome'>
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWa
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWa
  warnings.warn(msg, UserWarning)
```