# Take Away Assignment

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
df= pd.read_excel("DS_Python_Assignment.xlsx")
df.shape
```

Out[3]:
```
(10000, 117)
```

**Write a function/package in python which can create the below in the form of a html/pdf/word once you input the given data:**

1. **List down all the columns with missing values.**
2. **Categotize the columns based upon their data type and print, for ex: print all the numeric variables and other data types as well.**
3. **List the columns with duplicates a) Remove them b) Print before and after.**
4. **List the constant columns a) Remove them b) Print before and after.**
5. **Create box plot to visualise the outliers of all the numeric columns.**
6. **Create charts for any 6 columns and show their distribution.**

In [4]:
```
def generate_html_report(df):
# 1. List down all the columns with missing values.
    cols_with_null_values= []
    null_count= []
    for cols in df.columns:
        if df[cols].isnull().sum()!=0:
            cols_with_null_values.append(cols)
            null_count.append(df[cols].isnull().sum())


    null_cols= pd.DataFrame({"Column Names": cols_with_null_values, 'Null Value Counts': null_count})
    html1= null_cols.to_html(index=True, border=2)
```

```
# 2. Categotize the columns based upon their data type and print,
    #    for ex: print all the numeric variables and other data types as well
    int_type= []
    float_type= []
    object_type= []
    for cols in df.columns:
        if df[cols].dtypes == 'int64':
            int_type.append(cols)
        if df[cols].dtypes == 'float64':
            float_type.append(cols)
        if df[cols].dtypes == 'O':
            object_type.append(cols)
    cols_data_types= pd.DataFrame({"Data_Types": ['Integer', 'Float', 'Object'],
                                   "Column_Names": [' ,'.join(int_type), ' ,'.join(float_type), ' ,'.join(object_type)]})
    html2= cols_data_types.to_html(index=True, border=2)

# 3. List the columns with duplicates a) Remove them b) Print before and after
    ##Created a copy of dataset df as df_copy1 to for performing cleaning based on duplicate values.
    df_copy1= df.copy()
    duplicated_cols= []
    duplicated_values= []
    for cols in df_copy1.columns:
        if df_copy1[cols].duplicated().sum() != 0:
            duplicated_cols.append(cols)
            duplicated_values.append(df_copy1[cols].duplicated().sum())
    ##All the columns with Duplicate values
    df_duplicated = pd.DataFrame({"columns_names":duplicated_cols, "duplicated_values": duplicated_values})
    html3_1= df_duplicated.to_html(border=2)

    ##Removing duplicate values from each column
    for cols in df_copy1.columns:
        df_copy1.drop_duplicates(cols, inplace=True)
    columns_name= []
    duplicate_counts= []
    for cols in df_copy1.columns:
        columns_name.append(cols)
        duplicate_counts.append(df_copy1[cols].duplicated().sum())
    df_dup = pd.DataFrame({"columns_names": columns_name, "duplicated_values": duplicate_counts})
    html3_2= df_dup.to_html(border=2)

# 4. List the constant columns a) Remove them b) Print before and after
    constant_cols= []
    constant_vals= []
    for cols in df.columns:
        if df[cols].nunique()==1:
```

```
            constant_cols.append(cols)
            constant_vals.append(df[cols].nunique())
    df_constant= pd.DataFrame({'Columns with Single value/ Constant columns': constant_cols,
                               "Number of Unique Values": constant_vals})
    html4_1= df_constant.to_html(border=2)


# 5. Create box plot to visualise the outliers of all the numeric columns
    import os
    if not os.path.exists("charts"):
        os.mkdir("charts")
    chart_files= []
    for cols in df.iloc[:, :30].select_dtypes(include=['number']):
        plt.figure(figsize=(12,4))
        sns.boxplot(x=df[cols])
        plt.title(f'Boxplot for {cols}')
        plt.tight_layout()
        file_name = f"charts/boxplot_{cols}.png"
        plt.savefig(file_name)
        chart_files.append(file_name)
        plt.close()

# 6. Create charts for any 6 columns and show their distribution
    plt.figure(figsize=(12,5))
    sns.histplot(df['TENURE_IN_MONTHS'], bins=100, kde=True)
    plt.title(f'Distibution of TENURE_IN_MONTHS using Histogram')
    plt.tight_layout()
    chart1 = f"Histogram.png"
    plt.savefig(chart1)
    plt.close()
## Got Stuck in the 6th question.

    # Joining Tables
    html_content = f"""
    <html>
    <head><title>Data Report</title></head>
    <body>
        <h1>Data Report</h1>
        <h2>Table 1: Columns with Null Values</h2>
        {html1}
        <h2>Table 2: Data Types</h2>
        {html2}
        <h2>Table 3.1: Columns with Duplicate Values</h2>
        {html3_1}
        <h2>Table 3.2: Columns without Duplicate Values</h2>
        {html3_2}
```

```
        <h2>Table 4.1: Constant Columns</h2>
        {html4_1}
        <h2>Boxplots to check Outliers</h2>
    """
    for file_name in chart_files:
        html_content += f'<div><img src="{file_name}" alt="Boxplot" style="width:80%; margin-bottom:20px;"></div>'


    html_content += """
        <h2>Charts Showing Distribution of data</h2>
        <div>
        <img src="{chart1}" alt="Histogram of TENURE_IN_MONTHS" style="width:80%; margin-bottom:20px;">
        </div>
    </body>
    </html>
    """


    # Save the HTML to a file
    with open("report.html", "w") as f:
        f.write(f"<html><head><title>Data Report</title></head><body><h1>Data Report</h1>{html_content}</body></html>")

    print("HTML report generated: report.html")


generate_html_report(df)
```

```
HTML report generated: report.html
```