# Data Cleaning using Pandas

Sure, let's go through each line of code with explanations:

**Importing Necessary Libraries**

import pandas as pd

Here, we're importing the Pandas library and aliasing it as `pd`.

**Reading the Excel File**

df = pd.read_excel(r"Downloads\Customer Call List (1).xlsx")

This line reads an Excel file named "Customer Call List (1).xlsx" and stores its contents in a DataFrame named `df`.

**Dropping Duplicate Rows**

df = df.drop_duplicates()

This line removes duplicate rows from the DataFrame `df`.

**Dropping Unnecessary Columns**

df = df.drop(columns="Not_Useful_Column")

This line drops a column named "Not_Useful_Column" from the DataFrame `df`.

**Stripping Characters from a Column**

df["Last_Name"] = df["Last_Name"].str.strip("123._/")

This line removes leading and trailing occurrences of characters "123._/" from the values in the "Last_Name" column.

**Replacing Values in a Column**

df["Phone_Number"] = df["Phone_Number"].str.replace('nan--','')

df["Phone_Number"] = df["Phone_Number"].str.replace('Na--','')

These lines replace the substrings "nan--" and "Na--" in the "Phone_Number" column with an empty string, effectively removing them.

**Splitting a Column into Multiple Columns**

```
df[["Street_Address", "State", "Zip_Code"]] = df["Address"].str.split(',', n=2, expand=True)
```

This line splits the "Address" column by commas, creating three new columns: "Street_Address", "State", and "Zip_Code". The `n=2` parameter specifies that the splitting should occur only at the first two commas.

**Replacing Values in a Column (Continued)**

```
df["Do_Not_Contact"] = df["Do_Not_Contact"].str.replace('Yes','Y')

df["Do_Not_Contact"] = df["Do_Not_Contact"].str.replace('No','N')
```

These lines replace the values "Yes" and "No" in the "Do_Not_Contact" column with "Y" and "N", respectively.

**Filling NaN Values**

```
df = df.fillna('')
```

This line fills any missing (NaN) values in the DataFrame `df` with empty strings.

**Dropping Rows Based on a Condition**

```
for x in df.index:

    if df.loc[x, "Do_Not_Contact"] == 'Y':

        df.drop(x, inplace=True)
```

This loop iterates through the DataFrame's rows and drops any row where the value in the "Do_Not_Contact" column is "Y".

**Dropping Rows Based on a Condition (Continued)**

```
for x in df.index:

    if df.loc[x, "Phone_Number"] == '':

        df.drop(x, inplace=True)
```

Similar to the previous loop, this loop iterates through the DataFrame's rows and drops any row where the value in the "Phone_Number" column is an empty string.

**Resetting Index**

```
df = df.reset_index(drop=True)
```

This line resets the index of the DataFrame `df` after dropping rows, ensuring it's continuous without any gaps.

Each line of code performs specific data manipulation tasks, such as cleaning, transforming, or filtering the DataFrame. Let me know if you need further clarification on any part!