Here we make the class the main face of tkinter app

the libraries need to that pandas & tkinter

the first four libraries is the linked python scripts to tkinter app

In [ ]:
```
from win3_code import Win3Code
from win2_code import Win2Code
from win1_code import Win1Code
from self_code import SelfCode
import pandas as pd
from tkinter import filedialog,Tk,simpledialog,messagebox
from tkinter import *


class new(SelfCode,Win1Code,Win2Code,Win3Code):
    def __init__(self):
        super().__init__()
        self.button_select.config(command=self.upload)
        self.Button_finish.config(command=self.destroyy)
        self.combo['values']=['csv','json','xlsx','sql']
        self.combo.set('csv')
        self.butt_next.config(command=self.win1_new)
        self.k=0
        self.df=pd.DataFrame()
```

function to clsoe the app

function to upload file as aviliable extentions (xlsx, sql , csv, sql)

In [ ]:
```
def destroyy(self):
        self.destroy()

    def upload(self):
        lis=['second','third']
        # Initialize tkinter
        self.label1.config(text='Browser your '+lis[self.k]+' file')
        file_extension =self.combo.get() #extension_var.get()
        #Tk().withdraw()  # We don't want a full GUI, so keep the root window from appearing
        file_type = (file_extension, f'*.{file_extension}')
        file_location = filedialog.askopenfilename(filetypes=[file_type])
        # Show an "Open" dialog box and return the path to the selected file
```

```
        # Read the file into a pandas DataFrame
        if file_extension == 'csv':
            df = pd.read_csv(file_location, encoding='latin-1')
        elif file_extension == 'json':
            df = pd.read_json(file_location)
        elif file_extension == 'xlsx':
            df = pd.read_excel(file_location)
        elif file_extension == 'sql':
            # Assuming you have a connection string or engine
            connection_string = simpledialog.askstring("Input", "Enter your SQL connection string:")
            df = pd.read_sql(file_location, connection_string)
        else:
            raise ValueError("Unsupported file extension")
        self.df=df
        return self.df
```

functions to adabt the other pages of the app

In [ ]:

```
    def win1_new(self):
        self.win1_fcn()
        self.columns=list(self.df.columns)
        self.listbox.config(selectmode=MULTIPLE)
        self.listbox1.config(selectmode=MULTIPLE)

        for i in self.columns:
            self.listbox.insert('end',i)
        self.butt.config(command=self.choose_columns)
        self.butt1.config(command=self.win2_new)
        return self.columns

    def win2_new(self):
        self.column_process=[self.listbox1.get(i) for i in self.listbox1.curselection()]
        self.clean(self.df,self.column_process)
        self.win1.withdraw()
        self.win2_fcn()
        self.butt.config(command=self.download)
        self.butt1.config(command=self.info)
        self.butt2.config(command=self.close)

    def close(self):
        self.win1.destroy()
        self.win2.destroy()
```

to export your file

In [ ]:

```
def download(self):
        # Prompt user to select file format
        file_format = filedialog.asksaveasfilename(defaultextension=".csv",
                                                   filetypes=[("CSV files", "*.csv"),
                                                              ("Excel files", "*.xlsx"),
                                                              ("JSON files", "*.json"),
```

```
                                                    ("All files", "*.*")])
        if file_format:
            if file_format.endswith('.csv'):
                self.df.to_csv(file_format, index=False)
            elif file_format.endswith('.xlsx'):
                self.df.to_excel(file_format, index=False)
            elif file_format.endswith('.json'):
                self.df.to_json(file_format)
            else:
                messagebox.showerror("Error", "Unsupported file format!")
            messagebox.showinfo("Success", f"DataFrame saved to {file_format}")
```

show a summary of your data

In [ ]:
```
def info(self):
        self.win3_fcn()
        self.butt.config(command=self.win3.destroy)
        self.label.config(text='the total records of data is :'+str(self.num_rows))
        self.label1.config(text='the records is null in data  :'+str(self.num_rows_nulls))
        self.label2.config(text='the final total records after cleaning :'+str(self.num_rows_final))
```

choose the coulmns you want in the exported file

In [ ]:
```
def choose_columns(self):
        selected=[self.listbox.get(i) for i in self.listbox.curselection()]
        for i in self.columns:
            if i in selected:
                self.listbox1.insert('end',i)
                continue
            else:
                self.df.drop([i],axis=1,inplace=True)
        return
```

check if you data has time it handle it

In [ ]:
```
 #change date to timestamp then get years only then make it int to remove zeros
    def year(self,df):
        if 'year' in list(df.columns):
            df['year'] = df['year'].astype('datetime64[ns]')
            df['year']=df['year'].dt.year
            df['year']=df['year'].astype(int)
            # sort as year
            df.sort_values(by='year' , axis=0,inplace=True)
        return df
```

1- import nltk to make words minig and data cleaning

2- first fun to remove the unicode charcters

3- sec fun to remove any repeated letters or num

4- then get data ready for mining analysis

5- last func to get the sentiment score

In [ ]:
```python
#libraries of cleaning
import unicodedata
import re

#minig of data
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize


    #fun to clean data second method
    def remove_non_ascii(self,text):
        return re.sub(r'[^\x00-\x7F]+', '', text)

    #remove rebeted numbers and letters
    def lett(self,column):
        result = []
        for item in column:
            m = ''
            k=''
            for char in item:
                if char.isalpha() or char==' ':
                    if k==char:
                        continue
                    else:
                        m += char
                        k=char
            result.append(m)
        return result

    # Preprocess tweets
    def preprocess_tweet(self,tweet):
        # Remove URLs, mentions, and special characters
        # Convert to lowercase
        # Tokenize the tweet
        return word_tokenize(str(tweet).lower())
```

```
    # Sentiment analysis
    def get_sentiment(self,tweet):
        sia = SentimentIntensityAnalyzer()
        sentiment_scores = sia.polarity_scores(' '.join(tweet))
        return sentiment_scores['compound']
```

the main function to apply in our data

remove nulls and duplicates of data

concate the tweets coulmns

In [ ]:

```
#main fun for dataframe cleaning
    def clean(self,dataframe,column):
        print(column)
        compined=pd.DataFrame(columns=['com'])
        dataframe=self.year(dataframe)
        #removie nulls
        self.num_rows=dataframe.shape[0]
        dataframe.dropna(axis=0,inplace=True)
        self.num_rows_nulls=self.num_rows-dataframe.shape[0]
        k=0
        for i in column:
            #removie duplicates
            dataframe.drop_duplicates(subset=[i],keep='first',inplace=True)
            #clean data from unicodes
            #first method
            dataframe[i]=dataframe[i].apply(lambda x: unicodedata.normalize('NFKD', x)\
                                            .encode('ascii', 'ignore').decode('utf-8'))
            #second method
            dataframe[i]=dataframe[i].apply(self.remove_non_ascii)

            #third method
            dataframe[i]=self.lett(dataframe[i])

            #remove duplicates again
            dataframe.drop_duplicates(subset=[i],keep='first',inplace=True)

            #make data mining
            #compined coulmn
            if k ==0:
                compined['com']=dataframe[i]
                k+=1
            else:
                compined['com']+= ' ' +dataframe[i]
        self.num_rows_final=self.num_rows - self.num_rows_nulls
        processed_tweets = compined['com'].apply(self.preprocess_tweet)
```

```
        dataframe['sentiment_score'] = processed_tweets.apply(self.get_sentiment)
        #dataframe['sentiment_category'] =dataframe['sentiment_score'].apply(self.categorize_tweets)
        return
```

run our app

In [ ]:

```
a=new()
a.mainloop()
```