Import Modules

In [6]:
```
# Import modules
import easygui
import nbimporter
import numpy as np
import pandas as pd
import re
import threading
import time
import xlwings as xw
import unidecode

from thefuzz import fuzz
from openpyxl.utils import column_index_from_string
from openpyxl.utils import get_column_letter
from pathlib import Path
from files import FindFiles
from transfer import ImportData, ExportData
```

In [2]:
```
# Search for files
find_files = FindFiles()
```

Filter DataFrame

In [8]:
```
# Import the deduplicated dataset
data_files= [
    Path(r"C:\Users\OneDrive\Academic\MSc Research Project\Literature
Search\Processing The Results (06-Mar-2023)\01 - Deduplication.xlsx"), \
            ]

# Import data class
import_data = ImportData(files_for_import = data_files, \
                         worksheet_index="final_deduplication (986, 9)", \
                         headings_data_row=1, \
                         index_column = None, \
                         right_data_column = "R", \
                         has_spacing_columns = True, \
                         column_to_copy = "Key")
```

```
# Import data to datframes
import_data.make_dataframes()
dataframe = import_data.dataframe
```

```
Excel files imported:

Deduplication Steps.xlsx                    986 rows    9 Columns
```

In [6]:

```
""" DataFrame Feltling """;
```

In [10]:

```
class Fettling:
    def __init__(self, df):
        self.df = df


    def add_columns(self):
        self.df[["Status", "Exclusion Reason One (Abstract)", "Exclusion
Reason Two (Abstract)", "Exclusion Reason Three (Abstract)", "Notes"]] = ""


    def double_to_int(self):
        self.df = self.df.astype({'Key': pd.Int64Dtype(), 'Year':
pd.Int64Dtype()})


    def reorder_columns(self):
        self.df = self.df[['Key',
                           'Author',
                           'Title',
                           'Year',
                           'Journal',
                           'Name of Database',
                           'Abstract',
                           'Status',
                           'Exclusion Reason One (Abstract)',
                           'Exclusion Reason Two (Abstract)',
                           'Exclusion Reason Three (Abstract)',
                           'Notes']].copy()


    def run(self):
        self.add_columns()
        self.double_to_int()
        self.reorder_columns()
```

```
        export_data = ExportData(self.df, f"final_deduplication
{self.df.shape}")
        export_data.export_to_excel()
        return self.df


if __name__ == '__main__':
    fettling = Fettling(primary_dataframe)
    primary_dataframe = fettling.run()
```

In [ ]:

```
""" DataFrame Filtering """;
```

In [11]:

```
class DataFilter:
    def __init__(self, df):
        self.df = df

        self.animal_words = [r'\bmice(s?)\b', \
                             r'\bmouse(s?)\b', \
                             r'\brat(s?)\b', \
                             r'\brabbit(s?)\b', \
                             r'\bcalve(s?)\b', \
                             r'\bfrog(s?)\b', \
                             r'\bdog(s?)\b', \
                             r'\bcat(s?)\b', \
                             r'\bsheep(s?)\b']

        self.paediatric_words = [r'\bjuvenile(s?)\b', \
                                 r'\bchild(?:ren)?\b', \
                                 r'\bchildhood(s?)\b', \
                                 r'\badolescent(s?)\b', \
                                 r'\bpa?ediatric(?:s)?\b', \
                                 r'\byoung adult(s?)\b']

    def find_words(self, element, target_words):
        if not isinstance(element, str): return False
        return any(re.search(target_word, element.lower()) for target_word in
target_words)


    def filter_animal_words(self):
        filt = self.df['Abstract'].apply(self.find_words, args=
(self.animal_words,))
        df_animals = self.df[filt]
```

```
        # Export datafram
        export_data = ExportData(df_animals, f"animals {df_animals.shape}")
        export_data.export_to_excel()


    def filter_paediatric_words(self):
        filt1 = self.df['Title'].apply(self.find_words, args=
(self.paediatric_words,))
        filt2 = self.df['Journal'].apply(self.find_words, args=
(self.paediatric_words,))
        df_paediatrics = self.df[filt1 | filt2]

        # Export datafram
        export_data = ExportData(df_paediatrics, f"paediatrics
{df_paediatrics.shape}")
        export_data.export_to_excel()



    def dataframe_filtering(self):
        filt1 = self.df['Abstract'].apply(self.find_words, args=
(self.animal_words,))
        filt2 = self.df['Title'].apply(self.find_words, args=
(self.paediatric_words,))
        filt3 = self.df['Journal'].apply(self.find_words, args=
(self.paediatric_words,))
        self.df = self.df[~filt1 & ~filt2 & ~filt3]

        # Export dataframe
        export_data = ExportData(self.df, f"electronically_filt
{self.df.shape}")
        export_data.export_to_excel()

    def run(self):
        self.filter_animal_words()
        self.filter_paediatric_words()
        self.dataframe_filtering()
        return self.df

if __name__ == '__main__':
    Datafilter = DataFilter(dataframe)
    dataframe = Datafilter.run()
```