**To filter names of appliances in SQLite:**  
WITH CleanedNames AS (

SELECT

-- Extract the main part of the appliance name (before any parentheses)

TRIM(SUBSTR(Name, 1, INSTR(Name, '(') - 1)) AS MainName

FROM

Appliance

),

FilteredNames AS (

SELECT

CASE

WHEN MainName LIKE 'Non halogen lamp%' THEN 'Non halogen lamp'

WHEN MainName LIKE 'Halogen lamp%' THEN 'Halogen lamp'

ELSE MainName

END AS FilteredName

FROM

CleanedNames

)

SELECT

FilteredName

FROM

FilteredNames

GROUP BY

FilteredName

ORDER BY

FilteredName;

**To filter words**

SELECT \* FROM mytable  
WHERE column1 LIKE '%word1%'  
   OR column1 LIKE '%word2%'  
   OR column1 LIKE '%word3%'

SELECT \* FROM Appliances WHERE name like "%washing machine%"

[Python RegEx (w3schools.com)](https://www.w3schools.com/python/python_regex.asp) – this is to filter by name or letters.

es chveni eqsperimentia rom gagvefiltra table

SELECT \* FROM Appliance, Consumption WHERE Appliance.HouseIDREF=Consumption.HouseIDREF AND Appliance.ID = Consumption.ApplianceIDREF AND Appliance.Name LIKE "%washing machine%"

**Filter sqlite data for appliances, epoch time, values and ID of houses:**   
  
SELECT Appliance.ID AS ApplianceID, Consumption.Value AS ConsumptionValue, Consumption.Epochtime AS EpochTime FROM Appliance JOIN Consumption ON Appliance.ID = Consumption.ApplianceIDRef WHERE Appliance.Name LIKE "%washing machine%"

**Add filtering by zip code as well**

**Filter v2**

SELECT

Consumption.Epochtime,

Consumption.HouseIDREF,

Appliance.Name,

House.Zipcode,

Consumption.State,

Consumption.Value

FROM

Appliance

JOIN

Consumption ON Appliance.ID = Consumption.ApplianceIDRef

JOIN

House ON Appliance.HouseIDREF = House.ID

**First simulation with questions**

import os

import pandas as pd

def get\_user\_input(appliance\_name, recommended\_flexibility):

    while True:

        user\_input = input(f"The appliance '{appliance\_name}' is recommended to be '{recommended\_flexibility}'. Do you agree? (press 'y' for yes, 'n' for no): ").strip().lower()

        if user\_input in ['y', 'n']:

            return user\_input

def save\_user\_preferences(user\_id, user\_name, final\_df):

    directory = r'D:\Grenoble\_university\6\_month\_internship\working space\data\User choice (flexibility)'

    filename = f'flexibility\_preference\_user-N{user\_id}\_{user\_name}.csv'

    filepath = os.path.join(directory, filename)

    if os.path.exists(filepath):

        while True:

            overwrite = input(f"The file for user {user\_name} already exists. Do you want to replace it? (press 'y' for yes, 'n' for no): ").strip().lower()

            if overwrite == 'y':

                final\_df.to\_csv(filepath, index=False)

                print(f"\n{user\_name}'s preferences saved successfully.")

                break

            elif overwrite == 'n':

                user\_name = input(f"Please enter a different name for user {user\_id}: ").strip()

                filename = f'flexibility\_preference\_user-N{user\_id}\_{user\_name}.csv'

                filepath = os.path.join(directory, filename)

                if not os.path.exists(filepath):

                    final\_df.to\_csv(filepath, index=False)

                    print(f"\n{user\_name}'s preferences saved successfully.")

                    break

            else:

                print("Invalid input. Please enter 'y' or 'n'.")

    else:

        final\_df.to\_csv(filepath, index=False)

        print(f"\n{user\_name}'s preferences saved successfully.")

def confirm\_flexibility(merged\_df, num\_users):

    for user\_id in range(1, num\_users + 1):

        user\_name = input(f"\nUser {user\_id}, please enter your name: ").strip()

        while not user\_name:

            print("Name cannot be empty.")

            user\_name = input("Please enter your name: ").strip()

        print(f"\n{user\_name}, please answer the following questions:")

        user\_flexibility = []

        for index, row in merged\_df.iterrows():

            appliance\_name = row['Name']

            recommended\_flexibility = row['Flexibility']

            if pd.isna(recommended\_flexibility):

                recommended\_flexibility = 'unknown'

            if recommended\_flexibility == 'unknown':

                while True:

                    user\_input = input(f"Do you consider '{appliance\_name}' to be flexible, not\_flexible, or uncategorized? (press 1 for flexible, 2 for not\_flexible, 3 for uncategorized): ").strip().lower()

                    if user\_input in ['1', '2', '3']:

                        if user\_input == '1':

                            user\_flexibility.append((appliance\_name, 'flexible'))

                        elif user\_input == '2':

                            user\_flexibility.append((appliance\_name, 'not\_flexible'))

                        elif user\_input == '3':

                            user\_flexibility.append((appliance\_name, 'uncategorized'))

                        break

                    else:

                        print("Invalid input. Please enter 1, 2, or 3.")

            else:

                user\_input = get\_user\_input(appliance\_name, recommended\_flexibility)

                if user\_input == 'y':

                    user\_flexibility.append((appliance\_name, recommended\_flexibility))

                else:

                    while True:

                        user\_input = input(f"How do you define '{appliance\_name}'? (press 1 for flexible, 2 for not\_flexible, 3 for uncategorized): ").strip().lower()

                        if user\_input in ['1', '2', '3']:

                            if user\_input == '1':

                                user\_flexibility.append((appliance\_name, 'flexible'))

                            elif user\_input == '2':

                                user\_flexibility.append((appliance\_name, 'not\_flexible'))

                            elif user\_input == '3':

                                user\_flexibility.append((appliance\_name, 'uncategorized'))

                            break

                        else:

                            print("Invalid input. Please enter 1, 2, or 3.")

        final\_df = pd.DataFrame(user\_flexibility, columns=['Name', 'Flexibility'])

        save\_user\_preferences(user\_id, user\_name, final\_df)

excel\_file\_path = r'D:\Grenoble\_university\6\_month\_internship\working space\data\Flexibility\_yes\_no.xlsx'

# Load Excel file and select specific columns

flexibility\_df = pd.read\_excel(excel\_file\_path, usecols=['Name', 'Flexibility'])

# Assuming appliance\_df is already loaded

appliance\_df = df\_1

# Merge the dataframes on 'Name' column

merged\_df = pd.merge(appliance\_df, flexibility\_df, on='Name', how='left')

def get\_num\_users():

    while True:

        num\_users\_input = input("How many users' preferences do you want to collect? ").strip()

        if num\_users\_input.isdigit():

            return int(num\_users\_input)

        else:

            print("Please enter a valid number.")

num\_users = get\_num\_users()

confirm\_flexibility(merged\_df, num\_users)