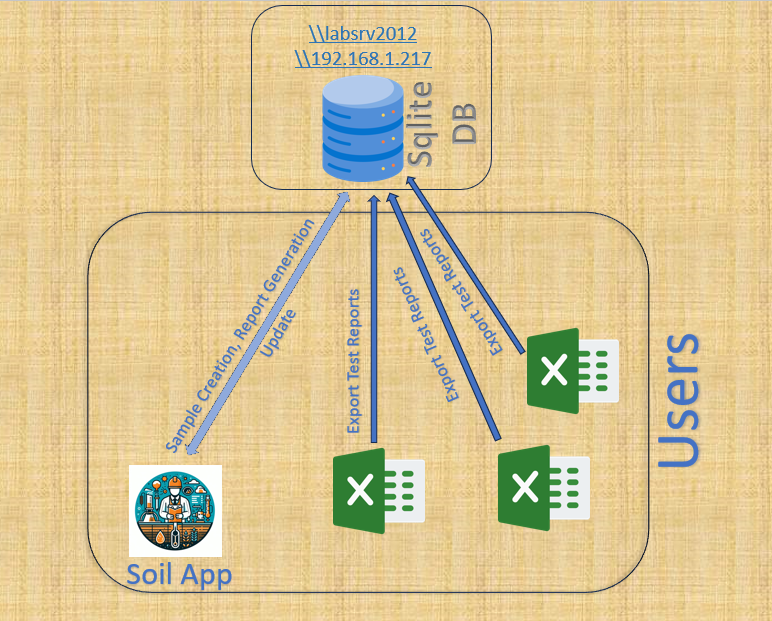
**Soil Sample Management Application**

**Introduction**

This project is a GUI-based application for managing soil sample records, using Tkinter for the GUI and SQLite for the database. It allows users to create new records, update existing ones, and generate reports. The overall project includes existing Excel files, with an executable file handling sample creation, report generation, and updates. Newly added buttons in the Excel files enable exporting data to the database.

Source Code and Edited excel files can be downloaded from here: <https://github.com/CETESTINDIA-MIS-IT/SOIL-REPORT-PROJECT>



**Prerequisites**

* Windows 10 or Later (Users)
* Must be connected to Lab Server([\\labsrv2012](file:///\\labsrv2012))(Users)
* Python 3.0 or later (Developers)
* Pyinstaller or Autopytoexe Package for compilation (Developers)
* Must have SQLITE 3 ODBC driver installed [(https://www.ch-werner.de/sqliteodbc/) (](https://www.ch-werner.de/sqliteodbc/)Users)

**Installation**

* This Program Does Not require installation. It’s a click to run software.

**File Structure**

* main3.py: Main application script containing all the logic for the GUI and database operations.

**Configuration**

**Database Connectivity Handling**

**Overview:** The application connects to an SQLite database stored in a central location. Due to occasional network connectivity issues, the database can be accessed via two different network paths: directly through IP address or through a hostname.

**Database Locations:**

* **Primary Location:**
  + Path: \\192.168.1.217\soil\_database\soil.db
  + Description: This is the direct IP address path to the SQLite database file. It serves as the primary connection route.
* **Secondary Location:**
  + Path: \\labsrv2012\soil\_database\soil.db
  + Description: In case direct access via IP encounters connectivity issues, the database can be accessed using the hostname labsrv2012. This provides an alternative route to ensure uninterrupted database connectivity.

**Connection Logic:** The application attempts to connect to the database using the primary location first (\\192.168.1.217\soil\_database\soil.db). If connection fails due to network issues or other errors, it retries using the secondary location (\\labsrv2012\soil\_database\soil.db). This approach ensures robust connectivity, mitigating potential disruptions caused by network instability.

**Error Handling:** If all attempts to connect to both locations fail, an error message is displayed prompting users to contact the IT department for further assistance.

**Usage:** Ensure that network configurations allow access to both IP and hostname paths as specified above to maintain seamless operation of the application.

**Application Features**

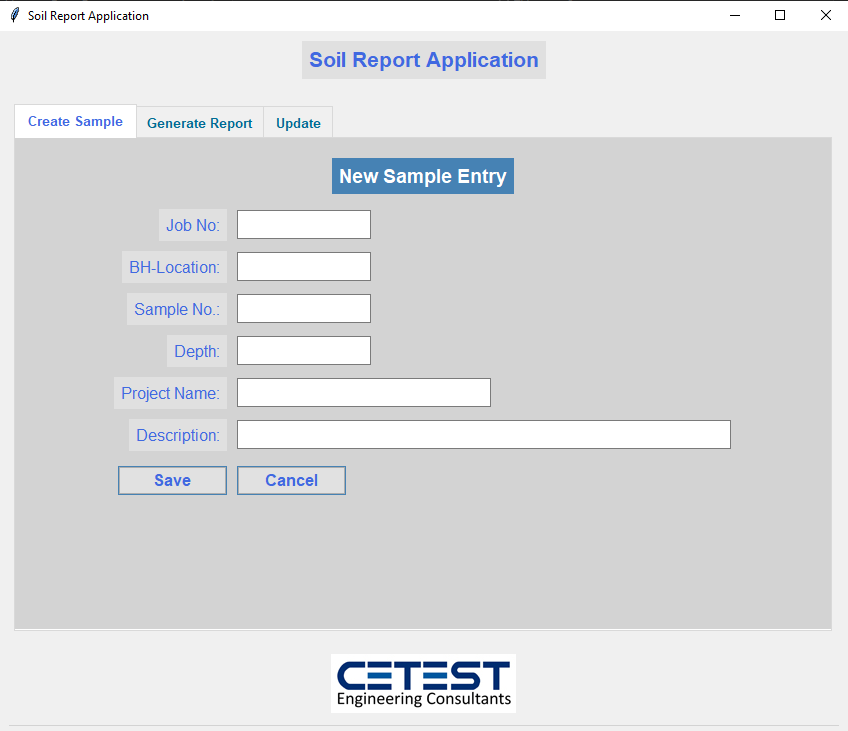
**Main Window**

The main window of the application contains tabs for different functionalities:

1. **Create Sample**: Form to create a new soil sample record.
2. **Generate Report**: Generate reports based on the records.
3. **Update**: Update existing records.

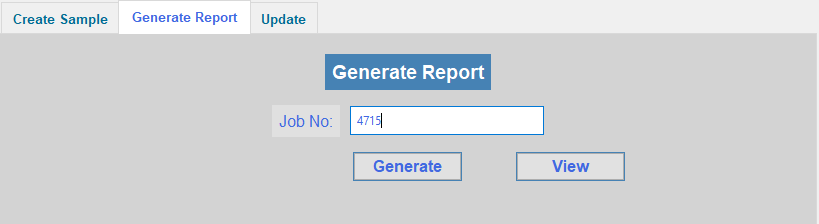
**Create Sample Tab**

* **Form Fields**:
  + Job No
  + Bore Hole
  + Sample No
  + Depth
  + Project Name
  + Description
* **Actions**:
  + Save: Validates and saves the new record to the database.
  + Clear: Clears the form fields.



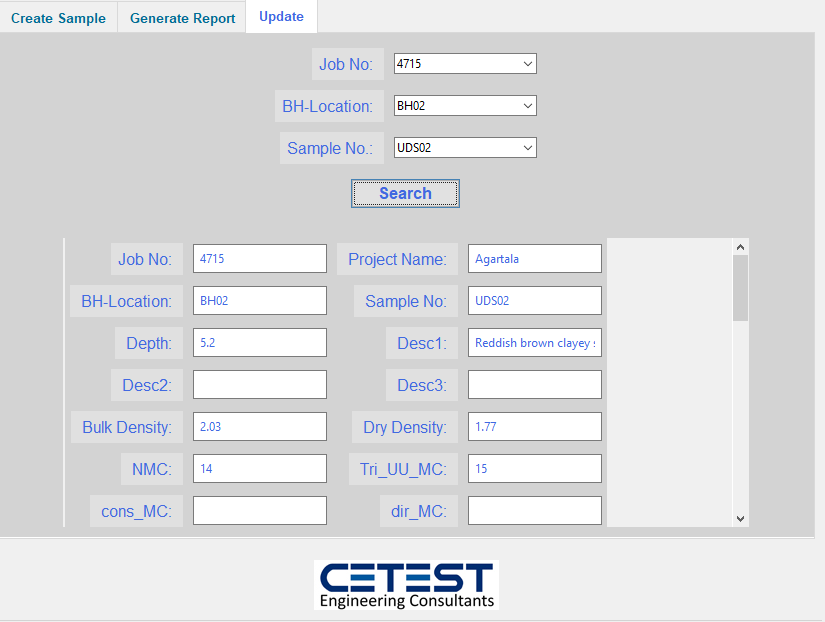
**Generate Report Tab**

* **Form Fields**:
  + Job Number for report generation.
* **Actions**:
  + View: Display BH-Location, Sample No, Depth along with various test types.
  + Generate Report: Generate a Excel report based on a entered job Number.



**Update Tab**

* **Form Fields**:
  + Select Job Number, BH-Location and Sample No from Dropdown.
* **Actions**:
  + Search: Fetch and display records for updating.
  + Update: Save changes to the selected record.
  + Cancel: Cancel the update operation.

****

**Database Design**

The database structure is straightforward. Initially, a unique UID (primary key) is created by the executable file based on a combination of job number, borehole location, and sample number. The database consists of 65 columns: one for the primary key (auto-generated), one for the job number, one for the borehole location, and one for the sample number. A new row can only be created if this combination of job number, borehole location, and sample number does not already exist.

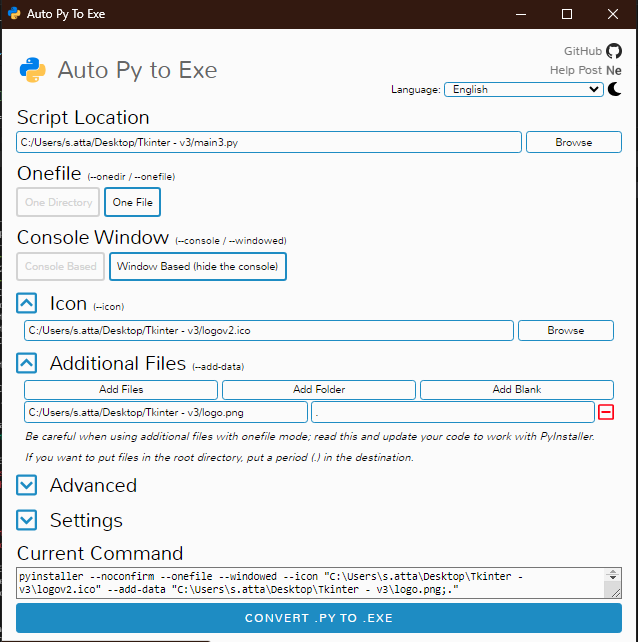
Several Excel files are involved in exporting data to the database. For example, one Excel file might dump data into columns 15, 20, and 22, while another might dump data into columns 45 and 49. Multiple Excel files can export their data to a single row in the database without any overlap, as each file handles separate columns. The data is dumped using the job number, borehole location, and sample number to identify the correct row.

If an existing row matching these criteria cannot be found, the Excel buttons will throw an error, as they cannot create new rows. The responsibility for creating new rows lies solely with the executable file.

**Debugging Some Common Errors**

# **# How to compile the executable?**

Make Sure you have Autopytoexe installed and run the application. Now, make the configuration like below:



Now all the packages will be converted into one executable file. Note that, if One File option is selected then it may take long to start the program. If One Dir option is selected then the program works faster. If One Dir Option is selected then the \_internal directory has to be place in the same place as the executable.

# **# How to handle driver errors in Excel?**

The executable file itself doesn’t require a driver to run. However, when working with Excel exports, the SQLite ODBC driver is necessary. Free drivers can be downloaded from <https://www.ch-werner.de/sqliteodbc/> . Although many drivers are listed on the website, for users on Windows 10 64-bit, there are two recommended drivers. If the first driver doesn’t work, try installing the second one.

* + 1. <https://www.ch-werner.de/sqliteodbc/sqliteodbc.exe>
    2. <https://www.ch-werner.de/sqliteodbc/sqliteodbc_w64.exe>

**Soil Sample Management Excel Integration**

**Introduction**

The project includes several Excel files, each with specific functionality added for this project. The Excel files involved are:

* 1. **Cn Template - 17.04.2024.xls**
  2. **DRSH EXPORT.xls**
  3. **Gr Size\_Hydrometer 17.04.2024.xls**
  4. **NMC-LL-PL-SL EXPORT.xls**
  5. **TRSH-UU EXPORT.xls**
  6. **UNCONFD & REMOULD EXPORT.xls**

Each Excel file contains two newly added buttons: "Export 2" and "Delete 2". These buttons interact with the SQLite database using a direct connection established through the SQLite driver. These functionalities are independent of the executable file.

* The **Export 2** button selects specific cells from the Excel file and uploads their contents to the corresponding columns in the database. It Exports the data only if the Sample no is created against a Bore Hole and a Job Number (New sample is created from the executable file).
* The **Delete 2** button removes only the values that were exported to the database via the Export 2 button, without deleting the entire row from the database.

This is the only new functionality added to these older Excel files during the project.

# APPENDIX

**main3.py code**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

from tkinter import filedialog

import sqlite3

import pandas as pd

import threading

from PIL import Image, ImageTk

import sys, os

import ctypes

db\_location = r"soil.db"

db\_locations = [r"\\198.168.1.217\soil\_database\soil.db", r"\\labsrv2012\soil\_database\soil.db"]

##Check If database is available

def resource\_path(relative\_path):

""" Get absolute path to resource, works for dev and for PyInstaller """

try:

# PyInstaller creates a temp folder and stores path in \_MEIPASS

base\_path = sys.\_MEIPASS

except Exception:

base\_path = os.path.abspath(".")

return os.path.join(base\_path, relative\_path)

# Define main application class

class SoilReportApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.geometry("850x700")

self.root.title("Soil Report Application")

style = ttk.Style()

style.configure("TLabel", font=("Helvetica", 12), padding=5, background="#E0E0E0", foreground="#4169E1")

style.configure("TEntry", font=("Helvetica", 12,), padding=5, fieldbackground="#F0F0F0", foreground="#4169E1")

style.configure("TButton", font=("Helvetica", 12, "bold"), padding=1, background="#4682B4", foreground="#4169E1", relief="raised")

style.map("TButton", background=[('active', '#5A9BD3'), ('hover', '#ADD8E6')], foreground=[('active', '#000000'), ('hover', '#000000')])

style.configure("Header.TLabel", font=("Helvetica", 14, "bold"), padding=5, background="#4682B4", foreground="#FFFFFF")

style.configure("Desc.TEntry", font=("Helvetica", 12), padding=5, fieldbackground="#F0F0F0", foreground="#4169E1")

style.configure("TFrame", background="#D3D3D3")

# For the 'hover' state to work, the 'TButton' style must be applied as below

style.map("TButton",

background=[('!active', '#4682B4'), ('hover', '#ADD8E6'), ('pressed', '#5A9BD3')],

foreground=[('!active', '#4169E1'), ('hover', '#000000'), ('pressed', '#FFFFFF')]

)

style.configure("TNotebook.Tab", font=("Helvetica", 10, "bold"), padding=[8, 5], background="#4682B4", foreground="#016d93")

style.map("TNotebook.Tab", background=[("selected", "#4169E1")], foreground=[("selected", "#4169E1")])

# Header Label

header = ttk.Label(root, text="Soil Report Application", font=("Helvetica", 16, "bold"), background="")

header.pack(pady=10)

# Create notebook (tabs container)

self.notebook = ttk.Notebook(root)

self.notebook.pack(expand=1, fill='both', padx=15, pady=15)

# Create tabs

self.create\_sample\_tab = ttk.Frame(self.notebook)

self.generate\_report\_tab = ttk.Frame(self.notebook)

self.update\_tab = ttk.Frame(self.notebook)

# Add tabs to the notebook

self.notebook.add(self.create\_sample\_tab, text='Create Sample')

self.notebook.add(self.generate\_report\_tab, text='Generate Report')

self.notebook.add(self.update\_tab, text='Update')

def rebind\_enter(event=None):

current\_tab = self.notebook.index(self.notebook.select())

if current\_tab == 0:

root.bind('<Return>', lambda e: save\_new\_record())

elif current\_tab == 1:

root.bind('<Return>', lambda e: view\_report())

elif current\_tab == 2:

root.bind('<Return>', lambda e: search\_action())

# Bind the <<NotebookTabChanged>> event to call rebind\_enter

self.notebook.bind('<<NotebookTabChanged>>', rebind\_enter)

# Call rebind\_enter initially to set the binding based on the default tab

rebind\_enter()

# Add content to each tab

self.add\_create\_sample\_content()

self.add\_generate\_report\_content()

self.add\_update\_content()

# Create the bottom frame

self.bottom\_frame = ttk.Frame(root)

self.bottom\_frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=5)

# Add initial content to the bottom frame

# self.bottom\_label = ttk.Label(self.bottom\_frame, text="Select a tab to load content here")

# self.bottom\_label.pack()

# Bind tab change event

#self.notebook.bind("<<NotebookTabChanged>>", self.on\_tab\_change)

# Resize and display image

image\_path = resource\_path("Logo.png")

image = Image.open(image\_path)

photo = ImageTk.PhotoImage(image)

# Use regular Tkinter Label for the image

self.image\_label = tk.Label(root, image=photo)

self.image\_label.image = photo # Keeping a reference

self.image\_label.pack(pady=5)

def add\_create\_sample\_content(self):

global save\_new\_record

# Create StringVar instances for entry fields

job\_no\_var = tk.StringVar()

bh\_loc\_var = tk.StringVar()

sample\_no\_var = tk.StringVar()

depth\_var = tk.StringVar()

project\_name\_var = tk.StringVar()

desc\_var = tk.StringVar()

def clear\_entries():

bh\_loc\_var.set("")

sample\_no\_var.set("")

depth\_var.set("")

# desc\_var.set("")

def is\_null\_or\_empty(\*args):

for arg in args:

if arg is None or arg == "":

return True

else:

return False

def save\_new\_record():

if is\_null\_or\_empty(job\_no\_var.get(), bh\_loc\_var.get(), sample\_no\_var.get(), depth\_var.get(), project\_name\_var.get(), desc\_var.get()):

messagebox.showerror("Error", "Please fill in all fields")

return

# Ask for confirmation

confirmed = messagebox.askyesno("Confirm Save", "Are you sure you want to save the new record?")

if not confirmed:

return

try:

conn = sqlite3.connect(db\_location)

c = conn.cursor()

c.execute("SELECT uid FROM soil WHERE job\_no = ? AND Bore\_Hole = ? AND sample\_no = ?",

(job\_no\_var.get(), bh\_loc\_var.get(), sample\_no\_var.get()))

result = c.fetchone()

if result:

messagebox.showerror("Duplicate Entry", "Record Already Exists. Please verify inputs.")

else:

c.execute(

"INSERT INTO soil (job\_no, Bore\_Hole, sample\_no, depth, Project, Desc1) VALUES (?, ?, ?, ?, ?, ?)",

(job\_no\_var.get(), bh\_loc\_var.get(), sample\_no\_var.get(), depth\_var.get(), project\_name\_var.get(), desc\_var.get()))

if c.rowcount == 1:

conn.commit()

messagebox.showinfo("Success!", "New Record created successfully")

clear\_entries()

else:

messagebox.showinfo("Error!", "Failed to insert the new record")

except sqlite3.Error as e:

messagebox.showerror("Error!", f"Error while connecting to the database\n{e}")

finally:

if conn:

conn.close()

# Frame for form

frame = ttk.Frame(self.create\_sample\_tab, padding=(20, 20, 20, 20))

frame.grid(row=0, column=0, sticky="nsew")

# Header

header\_label = ttk.Label(frame, text="New Sample Entry", style="Header.TLabel")

header\_label.grid(row=0, column=0, columnspan=2, pady=(0, 10))

# Create labels and entry fields

fields = [

("Job No:", job\_no\_var),

("BH-Location:", bh\_loc\_var),

("Sample No.:", sample\_no\_var),

("Depth:", depth\_var),

("Project Name:", project\_name\_var),

("Description:", desc\_var)

]

entry\_widgets = []

for i, (label\_text, var) in enumerate(fields):

label = ttk.Label(frame, text=label\_text, style="TLabel")

# Adjust width based on label\_text

if label\_text == "Project Name:":

width = 40

elif label\_text == "Description:":

width = 80

else:

width = 20

entry = ttk.Entry(frame, textvariable=var, style="TEntry", width=width)

label.grid(row=i + 1, column=0, padx=5, pady=5, sticky='e')

entry.grid(row=i + 1, column=1, padx=5, pady=5, sticky='w')

entry\_widgets.append(entry)

# Create button

save\_btn = ttk.Button(frame, text="Save", style="TButton", command=save\_new\_record)

cancel\_btn = ttk.Button(frame, text="Cancel", style="TButton", command=self.root.destroy)

save\_btn.grid(row=len(fields) + 1, column=0, padx=5, pady=10, sticky='e')

cancel\_btn.grid(row=len(fields) + 1, column=1, padx=5, pady=10, sticky='w')

# frame.bind('<Return>', lambda event: save\_new\_record())

# Ensure the frame expands to fill the window

self.create\_sample\_tab.grid\_columnconfigure(0, weight=1)

self.create\_sample\_tab.grid\_rowconfigure(0, weight=1)

frame.grid\_columnconfigure(0, weight=1)

frame.grid\_columnconfigure(1, weight=1)

def add\_generate\_report\_content(self):

global view\_report

def fetch\_job\_no():

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

query = "SELECT DISTINCT job\_no FROM soil"

cursor.execute(query)

values = [row[0] for row in cursor.fetchall()]

conn.close()

return values

def generate\_report():

job\_no = job\_combo\_box.get()

if not job\_no:

messagebox.showerror("Input Error", "Please select a job number.")

return

def generate():

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

query = "SELECT \* FROM soil WHERE job\_no = ?"

cursor.execute(query, (job\_no,))

rows = cursor.fetchall()

# Fetch column names excluding 'uid'

cursor.execute("PRAGMA table\_info(soil)")

columns\_info = cursor.fetchall()

columns = [info[1] for info in columns\_info if info[1] != 'uid']

conn.close()

if rows:

# Adjust rows to match the column names by excluding the last 'uid' field

rows = [row[:-1] for row in rows] # Exclude the 'uid' field from each row

# Create DataFrame

df = pd.DataFrame(rows, columns=columns)

df.insert(0, '', '')

# Save DataFrame to Excel file

filename = filedialog.asksaveasfilename(defaultextension=".xlsx",

initialfile=f"{job\_combo\_box.get()}.xlsx",

filetypes=[("Excel files", "\*.xlsx"), ("All files", "\*.\*")])

if filename:

df.to\_excel(filename, index=False)

messagebox.showinfo("Success", "Report generated successfully!")

else:

messagebox.showinfo("No Data", "No records found for the selected job number.")

# progress\_bar.stop()

# progress\_bar.start()

threading.Thread(target=generate).start()

def view\_report():

job\_no = job\_combo\_box.get()

if not job\_no:

messagebox.showerror("Input Error", "Please select a job number.")

return

data = fetch\_data(job\_no)

if data:

show\_data\_window(data,job\_no)

else:

messagebox.showinfo("No Data", "No records found for the selected job number.")

def fetch\_data(job\_no):

try:

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

query = "SELECT Bore\_Hole, sample\_no, depth, Test\_Type1, Test\_Type2, Test\_Type3, Test\_Type4, bulk\_density, NMC, LL, PL, SL, VREO18,sand, clay,SPG FROM soil WHERE job\_no = ?"

cursor.execute(query, (job\_no,))

data = cursor.fetchall()

conn.close()

return data

except Exception as e:

messagebox.showerror("Database Error", str(e))

return None

def show\_data\_window(data, jobNo):

data\_window = tk.Toplevel(self.root)

data\_window.title(f"Preview Report for Job No: {jobNo}")

columns = ["BH Loc.", "Sample No", "Depth", "TRSH-UU", "UNCONFD", "REMOULD", "DRSH-CQ", "BD", "NMC", "LL", "PL", "SL", "Void Ratio", "Sieve", "HYD", "SPG"]

tree = ttk.Treeview(data\_window, columns=columns, show='headings')

# Style the Treeview

style = ttk.Style()

style.configure("Treeview.Heading", font=("Helvetica", 9, "bold"), background="white", foreground="#4682b4")

style.configure("Treeview", font=("Helvetica", 10), rowheight=25)

style.map("Treeview", background=[("selected", "#ADD8E6")], foreground=[("selected", "black")])

# Create headings with styling

for col in columns:

tree.heading(col, text=col, anchor=tk.CENTER)

tree.column(col, anchor=tk.CENTER, width=70)

# Define tags for alternate row colors

tree.tag\_configure('oddrow', background='#f0f0ff')

tree.tag\_configure('evenrow', background='#ffffff')

# Insert data rows

for idx, row in enumerate(data):

sanitized\_row = ["x" if ele is None or ele == "" else "✓" if idx >= 3 else ele for idx, ele in enumerate(row)]

tag = 'oddrow' if idx % 2 == 0 else 'evenrow'

tree.insert("", tk.END, values=sanitized\_row, tags=(tag,))

# Add scrollbar

scrollbar = ttk.Scrollbar(data\_window, orient="vertical", command=tree.yview)

tree.configure(yscroll=scrollbar.set)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

tree.pack(fill=tk.BOTH, expand=True)

# Ensure the headers are visible

tree.pack(fill=tk.BOTH, expand=True)

data\_window.update\_idletasks()

# Frame for Generate Report tab content

frame = ttk.Frame(self.generate\_report\_tab, padding=(20, 20, 20, 20))

frame.grid(row=0, column=0, sticky="nsew")

# Header

header\_label = ttk.Label(frame, text="Generate Report", style="Header.TLabel")

header\_label.grid(row=0, column=0, columnspan=2, pady=(0, 10))

# Job Number ComboBox

job\_label = ttk.Label(frame, text="Job No:", style="TLabel")

job\_combo\_box = ttk.Combobox(frame, values=fetch\_job\_no(), style="TEntry", width=30)

job\_combo\_box.grid(row=1, column=1, padx=5, pady=5, sticky='w')

job\_label.grid(row=1, column=0, padx=5, pady=5, sticky='e')

# Generate Button

generate\_btn = ttk.Button(frame, text="Generate", style="TButton", command=generate\_report)

generate\_btn.grid(row=2, column=0, columnspan=2, padx=5, pady=10)

# View Button

view\_btn = ttk.Button(frame, text="View", style="TButton", command=view\_report)

view\_btn.grid(row=2, column=1, columnspan=2, padx=5, pady=10)

# frame.bind('<Return>', lambda event: view\_report())

# Ensure the frame expands to fill the window

self.generate\_report\_tab.grid\_columnconfigure(0, weight=1)

self.generate\_report\_tab.grid\_rowconfigure(0, weight=1)

frame.grid\_columnconfigure(0, weight=1)

frame.grid\_columnconfigure(1, weight=1)

def add\_update\_content(self):

global search\_action

def fetch\_values(query, params=()):

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

cursor.execute(query, params)

values = [row[0] for row in cursor.fetchall()]

conn.close()

return values

def fetch\_record(job\_no, bh\_loc, sample\_no):

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM soil WHERE job\_no = ? AND Bore\_Hole = ? AND sample\_no = ?", (job\_no, bh\_loc, sample\_no))

record = cursor.fetchone()

conn.close()

return record

def update\_record(job\_no, bh\_loc, sample\_no, new\_values):

conn = sqlite3.connect(db\_location)

cursor = conn.cursor()

fields = ", ".join([f"{field} = ?" for field in new\_values.keys()])

values = list(new\_values.values())

uid = cursor.execute("SELECT uid FROM soil WHERE job\_no = ? AND Bore\_Hole = ? AND sample\_no = ?", (job\_no, bh\_loc, sample\_no)).fetchone()[0]

values.append(uid) # Append uid to values

cursor.execute(f"UPDATE soil SET {fields} WHERE uid = ?", tuple(values)) # Pass values as a tuple

conn.commit()

conn.close()

# Search Frame

search\_frame = ttk.Frame(self.update\_tab)

search\_frame.pack(pady=10)

# header\_label = ttk.Label(search\_frame, text="Update", style="Header.TLabel")

# header\_label.grid(row=0, column=0, columnspan=2, pady=(0, 10), sticky='e')

job\_search\_label = ttk.Label(search\_frame, text="Job No: ")

job\_search\_label.grid(row=0, column=0, padx=5, pady=5, sticky='e')

bh\_loc\_search\_label = ttk.Label(search\_frame, text="BH-Location: ")

bh\_loc\_search\_label.grid(row=1, column=0, padx=5, pady=5, sticky='e')

sample\_no\_search\_label = ttk.Label(search\_frame, text="Sample No.: ")

sample\_no\_search\_label.grid(row=2, column=0, padx=5, pady=5, sticky='e')

# Fetch values for combo boxes from database

job\_nos = fetch\_values("SELECT DISTINCT job\_no FROM soil")

# Create combo boxes

job\_entry\_search = ttk.Combobox(search\_frame, values=job\_nos, state="readonly")

job\_entry\_search.grid(row=0, column=1, padx=5, pady=5, sticky='w')

bh\_loc\_search = ttk.Combobox(search\_frame, state="readonly")

bh\_loc\_search.grid(row=1, column=1, padx=5, pady=5, sticky='w')

sample\_no\_search = ttk.Combobox(search\_frame, state="readonly")

sample\_no\_search.grid(row=2, column=1, padx=5, pady=5, sticky='w')

def update\_bh\_locations(event):

job\_no = job\_entry\_search.get()

if job\_no:

bh\_locations = fetch\_values("SELECT DISTINCT Bore\_Hole FROM soil WHERE job\_no = ?", (job\_no,))

bh\_loc\_search['values'] = bh\_locations

bh\_loc\_search.set('')

sample\_no\_search.set('')

sample\_no\_search['values'] = []

def update\_sample\_nos(event):

job\_no = job\_entry\_search.get()

bh\_loc = bh\_loc\_search.get()

if job\_no and bh\_loc:

sample\_nos = fetch\_values("SELECT DISTINCT sample\_no FROM soil WHERE job\_no = ? AND Bore\_Hole = ?", (job\_no, bh\_loc))

sample\_no\_search['values'] = sample\_nos

sample\_no\_search.set('')

job\_entry\_search.bind("<<ComboboxSelected>>", update\_bh\_locations)

bh\_loc\_search.bind("<<ComboboxSelected>>", update\_sample\_nos)

# Frame 2 with scrollbar and dynamic labels and entries (initially hidden)

container = ttk.Frame(self.update\_tab)

canvas = tk.Canvas(container)

scrollbar = ttk.Scrollbar(container, orient="vertical", command=canvas.yview)

scrollable\_frame = ttk.Frame(canvas)

scrollable\_frame.bind(

"<Configure>",

lambda e: canvas.configure(

scrollregion=canvas.bbox("all")

)

)

canvas.create\_window((0, 0), window=scrollable\_frame, anchor="nw")

canvas.configure(yscrollcommand=scrollbar.set)

field\_names = "job\_no Project Bore\_Hole sample\_no depth Desc1 Desc2 Desc3 bulk\_density dry\_density NMC Tri\_UU\_MC cons\_MC dir\_MC col15 col16 SPG VREO18 Test\_Type1 Test\_Type2 Test\_Type3 Test\_Type4 PC1 STRN1 PC2 STRN2 PC3 STRN3 PC4 STRN4 PC5 STRN5 PC6 STRN6 PC7 STRN7 PC8 STRN8 PC9 STRN9 PC10 STRN10 PC11 STRN11 PC12 STRN12 COHE1 Fri1 COHE2 Fri2 COHE3 Fri3 COHE4 Fri4 LL PL sand silt clay gravel layer\_no SL col63 col164".split()

label\_Names = "Job No, Project Name, BH-Location, Sample No, Depth, Desc1, Desc2, Desc3, Bulk Density, Dry Density, NMC, Tri\_UU\_MC, cons\_MC, dir\_MC, col15, col16, SPG, VREO18, Test-Type1, Test Type2, Test-Type3, Test-Type4, PC1, STRN1, PC2, STRN2, PC3, STRN3, PC4, STRN4, PC5, STRN5, PC6, STRN6, PC7, STRN7, PC8, STRN8, PC9, STRN9, PC10, STRN10, PC11, STRN11, PC12, STRN12, COHE1, Fri1, COHE2, Fri2, COHE3, Fri3, COHE4, Fri4, LL, PL, sand, silt, clay, gravel, Layer No, SL, Col63, Col64".split(",")

entries = {}

for i, field in enumerate(label\_Names):

row, col = divmod(i, 2) # Arrange in two columns

label = ttk.Label(scrollable\_frame, text=field + ": ")

label.grid(row=row, column=col\*2, padx=5, pady=5, sticky='e')

entry = ttk.Entry(scrollable\_frame)

entry.grid(row=row, column=col\*2+1, padx=5, pady=5, sticky='w')

entries[f"{field\_names[i]}"] = entry

container.pack\_forget() # Hide the container initially

def search\_action():

job\_no = job\_entry\_search.get()

bh\_loc = bh\_loc\_search.get()

sample\_no = sample\_no\_search.get()

record = fetch\_record(job\_no, bh\_loc, sample\_no)

if record:

# Show container and populate entries with the record data

container.pack(fill="both", expand=True, pady=10, padx=65, ipadx=10, ipady=10)

canvas.pack(side="left", fill="both", expand=True)

scrollbar.pack(side="right", fill="y")

for i, field in enumerate(field\_names):

entry = entries[field]

entry.delete(0, tk.END)

if record[i] is not None:

entry.insert(0, str(record[i]))

else:

messagebox.showwarning("No Record Found", "No record exists for the provided details.")

def update\_action():

job\_no = job\_entry\_search.get()

bh\_loc = bh\_loc\_search.get()

sample\_no = sample\_no\_search.get()

confirmed = messagebox.askyesno("Confirmation", "Are you sure you want to update the record?")

if not confirmed:

return

new\_values = {field: str(entries[field].get()) for field in field\_names}

update\_record(job\_no, bh\_loc, sample\_no, new\_values)

messagebox.showinfo("Update Successful", "Record updated successfully.")

container.pack\_forget()

def cancel\_action():

container.pack\_forget() # Hide the frame

# Button to trigger search action

search\_button = ttk.Button(search\_frame, text="Search", command=search\_action, style='TButton')

search\_button.grid(row=3, column=0, columnspan=2, pady=10)

# search\_frame.bind('<Return>', lambda event: search\_action())

# Update and Cancel buttons

update\_button = ttk.Button(scrollable\_frame, text="Update", command=update\_action, style='TButton')

update\_button.grid(row=(len(field\_names) + 1) // 2, column=0, columnspan=2, pady=10)

cancel\_button = ttk.Button(scrollable\_frame, text="Cancel", command=cancel\_action, style='TButton')

cancel\_button.grid(row=(len(field\_names) + 1) // 2, column=2, columnspan=2, pady=10)

# Pack everything (initially hidden)

container.pack(fill="both", expand=True, pady=10, padx=65, ipadx=10, ipady=10)

canvas.pack(side="left", fill="both", expand=True)

scrollbar.pack(side="right", fill="y")

container.pack\_forget() # Hide the container initially

# def on\_tab\_change(self, event):

# selected\_tab = event.widget.tab(event.widget.index("current"))["text"]

# if selected\_tab == 'Create Sample':

# self.bottom\_label.config(text="Create Sample")

# elif selected\_tab == 'Generate Report':

# self.bottom\_label.config(text="Generate Report")

# elif selected\_tab == 'Update':

# self.bottom\_label.config(text="Update")

# Main Application

if \_\_name\_\_ == "\_\_main\_\_":

def connect\_to\_database(db\_locations):

for db\_location in db\_locations:

try:

conn = sqlite3.connect(db\_location)

c = conn.cursor()

return conn, c

except sqlite3.Error as e:

print(f"Error connecting to {db\_location}: {e}")

continue

return None, None

conn, c = connect\_to\_database(db\_locations)

if conn is None:

MessageBox = ctypes.windll.user32.MessageBoxW

MessageBox(None, 'Error connecting to database. Please inform IT department.', 'Error', 0)

sys.exit()

root = tk.Tk()

app = SoilReportApp(root)

root.mainloop()

Here is the SQL Dump code for database:

--

-- File generated with SQLiteStudio v3.4.4 on Mon Jul 8 15:06:25 2024

--

-- Text encoding used: System

--

PRAGMA foreign\_keys = off;

BEGIN TRANSACTION;

-- Table: soil

CREATE TABLE IF NOT EXISTS soil (

job\_no TEXT,

Project TEXT,

Bore\_Hole TEXT,

sample\_no TEXT,

depth REAL,

Desc1 TEXT,

Desc2 TEXT,

Desc3 TEXT,

bulk\_density REAL,

dry\_density REAL,

NMC INTEGER,

Tri\_UU\_MC INTEGER,

cons\_MC INTEGER,

dir\_MC INTEGER,

col15 INTEGER,

col16 INTEGER,

SPG REAL,

VREO18 REAL,

Test\_Type1 TEXT,

Test\_Type2 TEXT,

Test\_Type3 TEXT,

Test\_Type4 TEXT,

PC1 REAL,

STRN1 REAL,

PC2 REAL,

STRN2 REAL,

PC3 REAL,

STRN3 REAL,

PC4 REAL,

STRN4 REAL,

PC5 REAL,

STRN5 REAL,

PC6 REAL,

STRN6 REAL,

PC7 REAL,

STRN7 REAL,

PC8 REAL,

STRN8 REAL,

PC9 REAL,

STRN9 REAL,

PC10 REAL,

STRN10 REAL,

PC11 REAL,

STRN11 REAL,

PC12 REAL,

STRN12 REAL,

COHE1 REAL,

Fri1 REAL,

COHE2 REAL,

Fri2 REAL,

COHE3 REAL,

Fri3 REAL,

COHE4 REAL,

Fri4 REAL,

LL INTEGER,

PL INTEGER,

sand INTEGER,

silt INTEGER,

clay INTEGER,

gravel INTEGER,

layer\_no INTEGER,

SL INTEGER,

col63 TEXT,

col164 TEXT,

uid INTEGER PRIMARY KEY AUTOINCREMENT

UNIQUE

);

-- Index: jobindex

CREATE INDEX IF NOT EXISTS jobindex ON soil (

job\_no ASC

);

-- Index: sqlite\_autoindex\_soil\_1

CREATE UNIQUE INDEX IF NOT EXISTS sqlite\_autoindex\_soil\_1 ON soil (

uid COLLATE BINARY

);

COMMIT TRANSACTION;

PRAGMA foreign\_keys = on;

**Below are the VBA codes which were newly added to the existing excel files.**

**Cn Template - 17.04.2024.xls**

Sub CN\_Template\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim depth As Double

Dim cons\_MC As Integer

Dim VREO18 As Double

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("J2").value & ws.Range("K2").value

sample\_no = ws.Range("J3").value & ws.Range("K3").value

cons\_MC = ws.Range("G43").value

VREO18 = ws.Range("G45").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET cons\_MC = " & cons\_MC & ", VREO18 = " & VREO18 & " WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete2()

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

' Variables for identifiers

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim wb As Workbook

Dim ws As Worksheet

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("J2").value & ws.Range("K2").value

sample\_no = ws.Range("J3").value & ws.Range("K3").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET cons\_MC = NULL, VREO18 = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Values deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

**DRSH EXPORT.xls**

Sub DRSH\_CQ\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim Test\_type\_4 As String

Dim bulk\_density As Double

Dim dry\_density As Double

Dim dir\_MC As Integer

Dim PC10 As Double

Dim PC11 As Double

Dim PC12 As Double

Dim STRN10 As Double

Dim STRN11 As Double

Dim STRN12 As Double

Dim COHE4 As Double

Dim Fri4 As Double

' Initialize variables with cell values

Test\_type\_4 = "DRSH-CQ"

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("G2").value

bulk\_density = ws.Range("G25").value

dry\_density = ws.Range("G26").value

dir\_MC = ws.Range("G27").value

PC10 = ws.Range("A26").value

PC11 = ws.Range("A27").value

PC12 = ws.Range("A28").value

STRN10 = ws.Range("B26").value

STRN11 = ws.Range("B27").value

STRN12 = ws.Range("B28").value

COHE4 = ws.Range("C26").value

Fri4 = ws.Range("D26").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET Test\_Type4 = '" & Test\_type\_4 & "', bulk\_density = " & bulk\_density & ", dry\_density = " & dry\_density & ", dir\_MC = " & dir\_MC & \_

", PC10 = " & PC10 & ", PC11 = " & PC11 & ", PC12 = " & PC12 & ", STRN10 = " & STRN10 & ", STRN11 = " & STRN11 & ", STRN12 = " & STRN12 & \_

", COHE4 = " & COHE4 & ", Fri4 = " & Fri4 & " WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete2()

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("G2").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET Test\_Type4 = NULL, bulk\_density = NULL, dry\_density = NULL, dir\_MC = NULL, " & \_

"PC10 = NULL, PC11 = NULL, PC12 = NULL, STRN10 = NULL, STRN11 = NULL, STRN12 = NULL, " & \_

"COHE4 = NULL, Fri4 = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

**Gr Size\_Hydrometer 17.04.2024.xls**

Sub GR\_SIZE\_HYDROMETER\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

Dim cellvalue As Variant

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim SPG As Double

Dim gravel As Integer

Dim sand As Integer

Dim silt As Integer

Dim clay As Integer

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

JobNo = ws.Range("E1").value

BH\_Location = ws.Range("H1").value

sample\_no = ws.Range("H2").value

SPG = ws.Range("D57").value

gravel = ws.Range("K54").value

sand = ws.Range("K55").value

silt = ws.Range("K56").value

clay = ws.Range("K57").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET SPG = " & SPG & ", gravel = " & gravel & ", sand = " & sand & ", silt = " & silt & ", " & \_

"clay = " & clay & " WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

Exit Sub

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete2()

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Open the first connection and try the second one if it fails

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim wb As Workbook

Dim ws As Worksheet

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

' Variables for identifiers

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

' Initialize variables with cell values

On Error GoTo errHandler

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

JobNo = ws.Range("E1").value

BH\_Location = ws.Range("H1").value

sample\_no = ws.Range("H2").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET SPG = NULL, gravel = NULL, sand = NULL, silt = NULL, clay = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Values deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

**NMC-LL-PL-SL EXPORT.xls**

Sub NMC\_LL\_PL\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

' Reset error handler after successful connection

On Error GoTo errHandler

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim NMC As Integer

Dim LL As Integer

Dim PL As Integer

Dim SL As Integer

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

NMC = ws.Range("H13").value

LL = ws.Range("C24").value

PL = ws.Range("C13").value

SL = ws.Range("C53").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET NMC = " & NMC & ", LL = " & LL & ", PL = " & PL & ", SL = " & SL & " WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete2()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\192.168.1.217\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

' Reset error handler after successful connection

On Error GoTo errHandler

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for identifiers

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET NMC = NULL, LL = NULL, PL = NULL, SL = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Values deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

**TRSH-UU EXPORT.xls**

Sub TRSH\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\labsrv2012\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

' Reset error handler after successful connection

On Error GoTo errHandler

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim Test\_Type1 As String

Dim wb As Workbook

Dim ws As Worksheet

Dim bulk\_density As Double

Dim dry\_density As Double

Dim dir\_MC As Integer

Dim PC1 As Double

Dim PC2 As Double

Dim PC3 As Double

Dim STRN1 As Double

Dim STRN2 As Double

Dim STRN3 As Double

Dim COHE1 As Double

Dim Fri1 As Double

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

Test\_Type1 = "TRSH-UU"

bulk\_density = ws.Range("I6").value

dry\_density = ws.Range("I7").value

dir\_MC = ws.Range("I8").value

PC1 = ws.Range("I12").value

PC2 = ws.Range("I13").value

PC3 = ws.Range("I14").value

STRN1 = ws.Range("J12").value

STRN2 = ws.Range("J13").value

STRN3 = ws.Range("J14").value

COHE1 = ws.Range("N86").value

Fri1 = ws.Range("N8").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET Test\_Type1 = '" & Test\_Type1 & "', bulk\_density = " & bulk\_density & ", dry\_density = " & dry\_density & ", dir\_MC = " & dir\_MC & ", " & \_

"PC1 = " & PC1 & ", PC2 = " & PC2 & ", PC3 = " & PC3 & ", STRN1 = " & STRN1 & ", STRN2 = " & STRN2 & ", STRN3 = " & STRN3 & ", COHE1 = " & COHE1 & ", Fri1 = " & Fri1 & \_

" WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete\_2()

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\labsrv2012\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

' Reset error handler after successful connection

On Error GoTo errHandler

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for identifiers

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET Test\_Type1 = NULL, bulk\_density = NULL, dry\_density = NULL, dir\_MC = NULL, PC1 = NULL, PC2 = NULL, PC3 = NULL, " & \_

"STRN1 = NULL, STRN2 = NULL, STRN3 = NULL, COHE1 = NULL, Fri1 = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Values deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

**UNCONFD & REMOULD EXPORT.xls**

Sub UNCONFD\_AND\_REMOULD\_EXPORT()

' Check if Devart ODBC Driver for SQLite is installed

Dim driverInstalled As Boolean

driverInstalled = IsODBCDriverInstalled("SQLite3 ODBC Driver")

If Not driverInstalled Then

MsgBox "SQLite Driver is Not Installed. Please install it from: www.ch-werner.de/sqliteodbc/"

Exit Sub

End If

' Initialize ADO Connection object

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection string for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\labsrv2012\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strCon

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim exampleSQL As String

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for data to be inserted

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Dim STRN4 As Double

Dim STRN5 As Double

Dim STRN6 As Double

Dim STRN7 As Double

Dim STRN8 As Double

Dim STRN9 As Double

Dim COHE2 As Double

Dim COHE3 As Double

Dim Test\_Type2 As String

Dim Test\_Type3 As String

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

Test\_Type2 = "UNCONFD"

Test\_Type3 = "REMOULD"

STRN4 = ws.Range("I12").value

STRN5 = ws.Range("I13").value

STRN6 = ws.Range("I14").value

STRN7 = ws.Range("J12").value

STRN8 = ws.Range("J13").value

STRN9 = ws.Range("J14").value

COHE2 = ws.Range("I18").value

COHE3 = ws.Range("J18").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record

updateSQL = "UPDATE soil SET PC4 = 0, PC5 = 0, PC6 =0, PC7 = 0, PC8 = 0, PC9 = 0, Test\_Type2 = '" & Test\_Type2 & "', Test\_Type3 = '" & Test\_Type3 & "', STRN4 = " & STRN4 & ", STRN5 = " & STRN5 & ", " & \_

"STRN6 = " & STRN6 & ", STRN7 = " & STRN7 & ", STRN8 = " & STRN8 & ", STRN9 = " & STRN9 & ", COHE2 = " & COHE2 & ", COHE3 = " & COHE3 & \_

" WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Data updated successfully!"

Else

' If the record does not exist, show a message

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub

Function IsODBCDriverInstalled(driverName As String) As Boolean

Dim regKey As String

regKey = "HKEY\_LOCAL\_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\" & driverName

On Error Resume Next

Dim value As String

value = CreateObject("WScript.Shell").RegRead(regKey & "\Driver")

On Error GoTo 0

If value <> "" Then

IsODBCDriverInstalled = True

Else

IsODBCDriverInstalled = False

End If

End Function

Sub delete\_2()

Dim conn As New ADODB.Connection

Dim strConn As String

Dim strConn2 As String

' Provide the connection strings for SQLite using Devart ODBC Driver

strConn = "Driver={SQLite3 ODBC Driver};Database=C:\Users\s.atta\Desktop\Tkinter - v3\soil.db;"

strConn2 = "Driver={SQLite3 ODBC Driver};Database=\\labsrv2012\soil\_database\soil.db;"

' Try opening the first connection

On Error GoTo trySecondConn

conn.Open strConn

GoTo connected

trySecondConn:

' Try opening the second connection if the first fails

On Error GoTo errHandler

conn.Open strConn2

connected:

On Error GoTo errHandler

Dim checkSQL As String

Dim updateSQL As String

Dim uid As Long

Dim rs As New ADODB.Recordset

Dim wb As Workbook

Dim ws As Worksheet

' Variables for identifiers

Dim JobNo As String

Dim BH\_Location As String

Dim sample\_no As String

Set wb = ThisWorkbook

Set ws = wb.ActiveSheet

' Initialize variables with cell values

JobNo = ws.Range("B2").value

BH\_Location = ws.Range("E2").value

sample\_no = ws.Range("H2").value

' Check if JobNo, BH\_Location, and sample\_no already exist in the database

checkSQL = "SELECT uid FROM soil WHERE job\_no = '" & JobNo & "' AND Bore\_Hole = '" & BH\_Location & "' AND sample\_no = '" & sample\_no & "';"

rs.Open checkSQL, conn

If Not rs.EOF Then

' If the record exists, retrieve the uid

uid = rs.Fields("uid").value

rs.Close

' Update the existing record by clearing the specified fields

updateSQL = "UPDATE soil SET PC4 = NULL, PC5 = NULL, PC6 = NULL, PC7 = NULL, PC8 = NULL, PC9 = NULL, Test\_Type3 = NULL, Test\_Type2 = NULL, STRN4 = NULL, STRN5 = NULL, STRN6 = NULL, STRN7 = NULL, STRN8 = NULL, STRN9 = NULL, " & \_

"COHE2 = NULL, COHE3 = NULL WHERE uid = " & uid & ";"

conn.Execute updateSQL

MsgBox "Values deleted successfully!"

Else

' If the record does not exist

MsgBox "Sample Number Not found", vbCritical, "Warning"

End If

' Close the connection

conn.Close

Set conn = Nothing

Exit Sub

errHandler:

MsgBox "Error connecting to SQLite database: " & Err.Description

If Not conn Is Nothing Then

conn.Close

Set conn = Nothing

End If

End Sub