from redcap import Project

import pandas as pd

import datetime

# REDCap API configuration

def configure\_project(api\_url, api\_key):

return Project(api\_url, api\_key)

project = configure\_project('https://ruff.mrc.gm:8443/redcap/api/', '63A451E2DA73C4D6A4C64A17B36B9565')

sen\_project = configure\_project('https://redcap.mrc.gm:8443/redcap/api/', '008DBC63CA736CB68A3949DEFE43CC5D')

# Village mapping

village\_mapping = {

'001':'Dumbuto', '002':'Sankandi', '003':'Nioro Jattaba', '004':'Jattaba', '005':'Jiffarong', '006':'Bajana',

'007':'Kuli Kunda', '008':'Jamaru', '009':'Brikamanding', '010':'Kantong Kunda', '011':'Jali', '013':'Manduar',

'014':'Bang Kuling', '015':'Gissay', '016':'Tankular', '017':'Joli', '018':'Kuyang', '019':'Bantasu', '020':'Santamba',

'021':'Missira', '022':'Taborangkoto', '023':'Burong', '024':'Jula Kunda', '025':'Karantaba', '026':'Mandina',

'027':'Janneh Kunda', '028':'Kemoto', '029':'Keneba', '030':'Batelling', '031':'Sandeng', '032':'Wudeba', '034':'Kenokoto',

'035':'Manari', '036':'Nineteen', '040':'WUROKANG', '041':'KWINELLA SANSANKONO', '042':'KWINELLA NIA KUNDA', '043':'TENDABA',

'044':'BUMARR', '045':'BAMBAKO', '046':'KUNDONG MARIAYA', '047':'NEMA', '048':'KUNDANG NUMU KUNDA', '049':'KUNDANG FULA KUNDA',

'050':'NEMA KUTA', '051':'JIRROFF', '052':'MADINA ANGALLEH', '053':'JATTA KUNDA', '054':'MANDINA CENTRAL', '055':'SARE SARJO',

'056':'SIBETO', '057':'SARE NDALLA', '058':'TABANANI', '060':'WILLINGARA', '061':'SARE MAMUDU'

}

# Function to export records and create DataFrame

def export\_records\_to\_df(project, forms, fields=None):

data = project.export\_records(forms=forms)

df = pd.DataFrame(data)

if fields:

df = df[fields]

return df

# Function to process sensitization data

def process\_sensitization\_data(sen\_project, village\_mapping):

fields = [

'in\_kiang\_number', 'in\_name', 'in\_age', 'in\_village', 'in\_compound',

'in\_eden\_number', 'in\_contact1', 'in\_contact2', 'in\_contact3', 'in\_contact4',

'in\_outcome', 'in\_study\_number'

]

df = export\_records\_to\_df(sen\_project, ['indigo\_sensitisation'], fields)

df = df[(df['in\_outcome'] == '1') & df['in\_kiang\_number'].notnull() & df['in\_study\_number'].notnull()]

df = df.rename(columns={'in\_study\_number': 'con\_participantid\_q1'})

df['in\_village'] = df['in\_village'].map(village\_mapping)

return df

# Function to process withdrawals data

def process\_withdrawals\_data(project):

df = export\_records\_to\_df(project, ['study\_termination'], ['con\_participantid\_q1', 'st\_date', 'st\_reason', 'redcap\_event\_name'])

df['st\_date'] = pd.to\_datetime(df['st\_date'], errors='coerce')

df = df[(df['redcap\_event\_name'] == 'end\_of\_study\_arm\_1') & df['st\_date'].notnull()]

df\_withdrawals = pd.DataFrame({

'con\_participantid\_q1': df['con\_participantid\_q1'],

'withdrawal date': df['st\_date'],

'Reason': df['st\_reason'],

'Status': 'Terminated'

})

df\_withdrawals.to\_csv('Withdrawals.csv', index=False)

return df\_withdrawals

# Function to process USS scan data

def process\_uss\_data(project, df\_withdrawals):

df = export\_records\_to\_df(project, ['ultrasound\_scan'], ['con\_participantid\_q1', 'ultsc\_date', 'ultsc\_type\_q5', 'ultsc\_reschedule', 'redcap\_event\_name'])

df\_con\_uss = export\_records\_to\_df(project, ['consent', 'ultrasound\_scan'], ['con\_participantid\_q1', 'con\_participant\_eden\_num\_q11', 'ultsc\_date', 'ga\_weeks', 'ga\_days', 'con\_cluster\_q12', 'redcap\_event\_name'])

df\_con\_uss = df\_con\_uss[df\_con\_uss['ultsc\_date'].notnull()]

df\_con\_uss = df\_con\_uss[~df\_con\_uss['con\_participantid\_q1'].isin(df\_withdrawals['con\_participantid\_q1'])]

df\_con\_uss['ga\_weeks'] = pd.to\_numeric(df\_con\_uss['ga\_weeks'], errors='coerce')

df\_con\_uss['ga\_days'] = pd.to\_numeric(df\_con\_uss['ga\_days'], errors='coerce')

today = pd.Timestamp.today()

uss\_date = pd.to\_datetime(df\_con\_uss['ultsc\_date'])

total\_ga\_days = (df\_con\_uss['ga\_weeks'] \* 7) + df\_con\_uss['ga\_days']

edd = uss\_date + pd.to\_timedelta(280 - total\_ga\_days, unit='days')

days\_diff = (today - uss\_date).dt.days

current\_ga = (days\_diff + total\_ga\_days) / 7

edd\_df = pd.DataFrame({

'con\_participantid\_q1': df\_con\_uss['con\_participantid\_q1'],

'USS Date': uss\_date,

'EDEN | Infant Number': df\_con\_uss['con\_participant\_eden\_num\_q11'],

'Gestational Age (Weeks)': df\_con\_uss['ga\_weeks'],

'Gestational Age (Days)': df\_con\_uss['ga\_days'],

'Cluster': df\_con\_uss['con\_cluster\_q12'],

'Current Gestational Age': current\_ga,

'(Expected)Delivery Date ((E)DD)': edd

})

edd\_df.to\_csv("EDD\_LIST.CSV", index=False)

return edd\_df

# Function to process weekly data

def process\_weekly\_data(df, week\_arm, file\_name):

df['ultsc\_date'] = pd.to\_datetime(df['ultsc\_date'], errors='coerce')

weekly\_df = df[(df['redcap\_event\_name'] == week\_arm) & df['ultsc\_date'].notnull()]

weekly\_df = pd.DataFrame({

'con\_participantid\_q1': weekly\_df['con\_participantid\_q1'],

'uss\_date': weekly\_df['ultsc\_date'],

'uss\_type': weekly\_df['ultsc\_type\_q5'],

'Status': f'{week\_arm.replace("\_arm\_1", "").replace("week\_", "Week ")} completed'

})

weekly\_df.to\_csv(file\_name, index=False)

return weekly\_df

# Process sensitization data

sen\_log\_data = process\_sensitization\_data(sen\_project, village\_mapping)

# Process withdrawals data

df\_withdrawals = process\_withdrawals\_data(project)

# Process USS scan data

edd\_df = process\_uss\_data(project, df\_withdrawals)

# Process weekly data

df\_uss = export\_records\_to\_df(project, ['ultrasound\_scan'], ['con\_participantid\_q1', 'ultsc\_date', 'ultsc\_type\_q5', 'redcap\_event\_name'])

wk20\_df = process\_weekly\_data(df\_uss, 'week\_20\_arm\_1', 'Week 20.csv')

wk28\_df = process\_weekly\_data(df\_uss, 'week\_28\_arm\_1', 'Week 28.csv')

wk36\_df = process\_weekly\_data(df\_uss, 'week\_36\_arm\_1', 'Week 36.csv')

##Delivery

# %% [markdown]

# DELIVERY

# %%

# Function to process delivery data

def process\_delivery\_data(project):

# Extract delivery data

delivery\_data = project.export\_records(forms=['delivery'])

# Convert data to DataFrame

delv\_data = pd.DataFrame(delivery\_data)

# Convert delivery date to datetime

delv\_data['del\_ddate\_q7'] = pd.to\_datetime(delv\_data['del\_ddate\_q7'], errors='coerce')

# Specify the fields to extract

df\_delv = delv\_data[['con\_participantid\_q1', 'del\_vdate\_q1', 'del\_place\_q2', 'del\_lbonsetdatetime\_q4', 'del\_gestage', 'del\_ddate\_q7', 'redcap\_event\_name']]

df\_delv = df\_delv[

(df\_delv['redcap\_event\_name'] == 'delivery\_\_postpart\_arm\_1') &

(df\_delv['del\_ddate\_q7'].notnull())

]

# Create DataFrame for delivery information

delivery\_df = pd.DataFrame({

'con\_participantid\_q1': df\_delv['con\_participantid\_q1'],

'Delivery Date': df\_delv['del\_ddate\_q7'],

'GA at Delivery': df\_delv['del\_gestage'],

'Status': 'Delivered'

})

# Save the DataFrame to a CSV file

delivery\_df.to\_csv('Delivery.csv', index=False)

return delivery\_df

# Process delivery data

delivery\_df = process\_delivery\_data(project)

## Infant completed visits

import pandas as pd

# %% [markdown]

# INFANT-MOTHER PAIR

# %%

# Function to process infant data

def process\_infant\_data(project):

# Extract infant anthropometry and stool collection data

inf\_data = project.export\_records(forms=['infant\_anthropometry', 'infant\_stool\_collection'])

inf\_data = pd.DataFrame(inf\_data)

# Select relevant columns and convert dates to datetime

inf\_data = inf\_data[['con\_participantid\_q1', 'ina\_infantid', 'ian\_date', 'ian\_visitoutcome',

'infant\_anthropometry\_complete', 'isc\_dvdate\_q1', 'isc\_dtime\_q2',

'isc\_stoolcollected\_q3', 'infant\_stool\_collection\_complete', 'redcap\_event\_name']]

inf\_data['ian\_date'] = pd.to\_datetime(inf\_data['ian\_date'], errors='coerce')

inf\_data['isc\_dvdate\_q1'] = pd.to\_datetime(inf\_data['isc\_dvdate\_q1'], errors='coerce')

inf\_data['isc\_dtime\_q2'] = pd.to\_datetime(inf\_data['isc\_dtime\_q2'], errors='coerce').dt.time

return inf\_data

# Function to apply visit conditions

def apply\_visit\_conditions(df):

return (

(df['ian\_date'].notnull()) &

(df['ian\_visitoutcome'].notnull()) &

(df['infant\_anthropometry\_complete'] == '2') &

(df['isc\_dvdate\_q1'].notnull()) &

(df['isc\_stoolcollected\_q3'].notnull()) &

(df['infant\_stool\_collection\_complete'] == '2')

)

# Function to process data for a given month

def process\_month\_data(df\_infant, df\_delv, month\_name):

inf\_month = df\_infant[

(df\_infant['redcap\_event\_name'] == month\_name) &

apply\_visit\_conditions(df\_infant)

]

# Merge with delivery data based on participant ID

month\_infant\_mother = df\_delv.merge(inf\_month, on='con\_participantid\_q1', how='left')

# Return required variables based on common participant ID

month\_infant\_mother = month\_infant\_mother[['con\_participantid\_q1', 'del\_ddate\_q7', 'ina\_infantid', 'ian\_date']]

month\_infant\_mother = month\_infant\_mother[

(month\_infant\_mother['del\_ddate\_q7'].notnull()) &

(month\_infant\_mother['ian\_date'].notnull())

]

# Extract required columns and save to CSV

month\_df = pd.DataFrame({

'con\_participantid\_q1': month\_infant\_mother['con\_participantid\_q1'],

'Delivery Date': pd.to\_datetime(month\_infant\_mother['del\_ddate\_q7']),

'Infant ID': month\_infant\_mother['ina\_infantid'],

'Infant anthrops Date': pd.to\_datetime(month\_infant\_mother['ian\_date']),

'Status': f'{month\_name.replace("\_arm\_2", "").capitalize()} completed'

})

month\_df.to\_csv(f'{month\_name.replace("\_arm\_2", "").capitalize()}.csv', index=False)

return month\_df

# Process delivery data

delivery\_df = process\_delivery\_data(project)

# Process infant data

infant\_df = process\_infant\_data(project)

# Process data for each month

month\_names = [

'month\_1\_arm\_2', 'month\_2\_arm\_2', 'month\_3\_arm\_2', 'month\_4\_arm\_2',

'month\_5\_arm\_2', 'month\_6\_arm\_2', 'month\_7\_arm\_2', 'month\_8\_arm\_2',

'month\_9\_arm\_2', 'month\_10\_arm\_2', 'month\_11\_arm\_2', 'month\_12\_arm\_2'

]

# Process and save data for each month

for month in month\_names:

process\_month\_data(infant\_df, delivery\_df, month)

## ANTENATAL CALLS

import pandas as pd

def load\_data(file\_path):

return pd.read\_csv(file\_path)

def merge\_and\_filter(df\_main, df\_filter, key):

merged\_df = pd.merge(df\_main, df\_filter, on=key, how='inner')

return df\_main[~df\_main[key].isin(merged\_df[key])]

def process\_data(df, visit\_type, gestational\_weeks\_limit=None):

df = df[['con\_participantid\_q1', 'con\_participant\_eden\_num\_q11', 'ultsc\_date', 'ultsc\_reschedule',

'ga\_weeks', 'ga\_days', 'con\_cluster\_q12', 'redcap\_event\_name']]

if gestational\_weeks\_limit:

df['ga\_weeks'] = pd.to\_numeric(df['ga\_weeks'], errors='coerce')

df['ga\_days'] = pd.to\_numeric(df['ga\_days'], errors='coerce')

df = df[(df['redcap\_event\_name'] == 'recruitment\_arm\_1') &

(df['ultsc\_date'].notnull()) &

(df['ga\_weeks'] <= gestational\_weeks\_limit)]

else:

df = df[(df['redcap\_event\_name'] == 'recruitment\_arm\_1') &

(df['ultsc\_date'].notnull()) &

(df['ultsc\_reschedule'].notnull())]

return df

def calculate\_dates(df, visit\_type):

today = pd.Timestamp.today()

df['uss\_date'] = pd.to\_datetime(df['ultsc\_date'])

df['due\_date'] = pd.to\_datetime(df['ultsc\_reschedule'])

df['ga\_weeks'] = pd.to\_numeric(df['ga\_weeks'], errors='coerce')

df['ga\_days'] = pd.to\_numeric(df['ga\_days'], errors='coerce')

total\_ga\_days = (df['ga\_weeks'] \* 7) + df['ga\_days']

df['edd'] = df['uss\_date'] + pd.to\_timedelta(280 - total\_ga\_days, unit='days')

days\_diff = (today - df['uss\_date']).dt.days

df['current\_ga'] = (days\_diff + total\_ga\_days) / 7

visit\_dates = {

'Rescheduled': (140, 147),

'Week 20': (140, 147),

'Week 28': (196, 203),

'Week 36': (252, 259)

}

if visit\_type in visit\_dates:

start\_diff, end\_diff = visit\_dates[visit\_type]

df['due\_date'] = df['uss\_date'] + pd.to\_timedelta(start\_diff - total\_ga\_days, unit='days')

df['last\_date'] = df['uss\_date'] + pd.to\_timedelta(end\_diff - total\_ga\_days, unit='days')

return df

def create\_dataframe(df, visit\_type):

df\_processed = df[['con\_participantid\_q1', 'con\_cluster\_q12', 'uss\_date', 'con\_participant\_eden\_num\_q11',

'ga\_weeks', 'ga\_days', 'edd', 'current\_ga', 'due\_date', 'last\_date']]

df\_processed.columns = ['con\_participantid\_q1', 'Cluster', 'USS Date', 'EDEN | Infant Number',

'Gestational Age (Weeks)', 'Gestational Age (Days)',

'(Expected)Delivery Date ((E)DD)', 'Current Gestational Age',

'Due Date', 'Last Date']

df\_processed['Visit'] = visit\_type

return df\_processed

def main():

# Load data

project\_data = project.export\_records(forms=['consent', 'ultrasound\_scan'])

project\_df = pd.DataFrame(project\_data)

# Week 20

week\_20\_data = process\_data(project\_df, visit\_type='Week 20', gestational\_weeks\_limit=20)

week\_20\_data = merge\_and\_filter(week\_20\_data, load\_data('Withdrawals.csv'), 'con\_participantid\_q1')

week\_20\_data = merge\_and\_filter(week\_20\_data, load\_data('Delivery.csv'), 'con\_participantid\_q1')

week\_20\_df = calculate\_dates(week\_20\_data, visit\_type='Week 20')

week\_20\_df = create\_dataframe(week\_20\_df, visit\_type='Week 20')

week\_20\_df = merge\_and\_filter(week\_20\_df, load\_data('Week 20.csv'), 'con\_participantid\_q1')

# Week 28

week\_28\_data = process\_data(project\_df, visit\_type='Week 28')

week\_28\_data = merge\_and\_filter(week\_28\_data, load\_data('Withdrawals.csv'), 'con\_participantid\_q1')

week\_28\_data = merge\_and\_filter(week\_28\_data, load\_data('Delivery.csv'), 'con\_participantid\_q1')

week\_28\_df = calculate\_dates(week\_28\_data, visit\_type='Week 28')

week\_28\_df = create\_dataframe(week\_28\_df, visit\_type='Week 28')

week\_28\_df = merge\_and\_filter(week\_28\_df, load\_data('Week 28.csv'), 'con\_participantid\_q1')

# Week 36

week\_36\_data = process\_data(project\_df, visit\_type='Week 36')

week\_36\_data = merge\_and\_filter(week\_36\_data, load\_data('Withdrawals.csv'), 'con\_participantid\_q1')

week\_36\_data = merge\_and\_filter(week\_36\_data, load\_data('Delivery.csv'), 'con\_participantid\_q1')

week\_36\_df = calculate\_dates(week\_36\_data, visit\_type='Week 36')

week\_36\_df = create\_dataframe(week\_36\_df, visit\_type='Week 36')

week\_36\_df = merge\_and\_filter(week\_36\_df, load\_data('Week 36.csv'), 'con\_participantid\_q1')

# Combine all DataFrames (if needed)

all\_df = pd.concat([week\_20\_df, week\_28\_df, week\_36\_df], ignore\_index=True)

return all\_df

# Run the main function and get the combined DataFrame

final\_df = main()

----------------------------------------------------------------------------------------------------

## POST-PARTUM CALLS

import pandas as pd

# Function to read and filter data based on exclusion lists

def read\_and\_exclude(filename, df, key):

exclusion\_df = pd.read\_csv(filename)

merged\_df = pd.merge(df, exclusion\_df, on=key, how='inner')

return df[~df[key].isin(merged\_df[key])]

# Function to calculate due date and last date

def calculate\_dates(df, months, days=None):

df['due\_date'] = df['del\_ddate\_q7'] + pd.DateOffset(months=months)

if days is not None:

df['last\_date'] = df['del\_ddate\_q7'] + pd.DateOffset(months=months, days=days)

else:

df['last\_date'] = df['del\_ddate\_q7'] + pd.DateOffset(months=months)

return df

# Post Partum Call

def post\_partum\_call(project, df\_delv):

# Extract neonate screening and infant ID

neonate\_data = project.export\_records(forms=['infantid', 'neonate\_screening'])

neonate\_df = pd.DataFrame(neonate\_data)

neonate\_df = neonate\_df[['con\_participantid\_q1', 'infantid', 'neo\_vdate', 'redcap\_event\_name']]

neonate\_df = neonate\_df[

(neonate\_df['redcap\_event\_name'] == 'baby\_check\_arm\_2') &

(neonate\_df['neo\_vdate'].notnull())

]

# Merge neonate data with delivery data

neonate\_mother = df\_delv.merge(neonate\_df, on='con\_participantid\_q1', how='inner')

neonate\_mother = neonate\_mother[['con\_participantid\_q1', 'del\_ddate\_q7', 'infantid']]

# Exclude withdrawals

neonate\_mother = read\_and\_exclude('Withdrawals.csv', neonate\_mother, 'con\_participantid\_q1')

return neonate\_mother

# Month Calls

def month\_call(neonate\_mother, months, visit\_name, days=None):

neonate\_mother = pd.DataFrame(neonate\_mother)

neonate\_mother['del\_ddate\_q7'] = pd.to\_datetime(neonate\_mother['del\_ddate\_q7'])

neonate\_mother = calculate\_dates(neonate\_mother, months, days)

month\_df = pd.DataFrame({

'con\_participantid\_q1': neonate\_mother['con\_participantid\_q1'],

'(Expected)Delivery Date ((E)DD)': pd.to\_datetime(neonate\_mother['del\_ddate\_q7']),

'EDEN | Infant Number': neonate\_mother['infantid'],

'Due Date': neonate\_mother['due\_date'],

'Last Date': neonate\_mother['last\_date'],

'Visit': visit\_name

})

# Exclude existing data

month\_data = pd.read\_csv(f'{visit\_name}.csv')

month\_df = read\_and\_exclude(f'{visit\_name}.csv', month\_df, 'con\_participantid\_q1')

return month\_df

# Execute functions

project = ... # Your project object

df\_delv = ... # Your delivery data DataFrame

# Post Partum Call

neonate\_mother = post\_partum\_call(project, df\_delv)

# Month Calls

month1\_df = month\_call(neonate\_mother, 1, 'Month 1')

month2\_df = month\_call(neonate\_mother, 2, 'Month 2', days=7)

month3\_df = month\_call(neonate\_mother, 3, 'Month 3', days=7)

month4\_df = month\_call(neonate\_mother, 4, 'Month 4', days=7)

month5\_df = month\_call(neonate\_mother, 5, 'Month 5', days=7)

month6\_df = month\_call(neonate\_mother, 6, 'Month 6', days=14)

# Output or save dataframes as needed

# month1\_df.to\_csv('Month 1 Updated.csv', index=False)

# month2\_df.to\_csv('Month 2 Updated.csv', index=False)

# month3\_df.to\_csv('Month 3 Updated.csv', index=False)

# month4\_df.to\_csv('Month 4 Updated.csv', index=False)

# month5\_df.to\_csv('Month 5 Updated.csv', index=False)

# month6\_df.to\_csv('Month 6 Updated.csv', index=False)

####call list generation code###

import pandas as pd

import datetime

# Function to concatenate due dates DataFrames

def concatenate\_due\_dates(\*dfs):

return pd.concat(dfs, ignore\_index=True)

# Function to merge with sensitization data

def merge\_with\_sensitization(due\_dates\_df, sensitization\_df):

return pd.merge(due\_dates\_df, sensitization\_df, on='con\_participantid\_q1', how='left')

# Function to format final DataFrame

def format\_final\_df(df):

return pd.DataFrame({

'PARTICIPANT ID': df['con\_participantid\_q1'],

'EDEN | INFANT ID': df['EDEN | Infant Number'],

'KIANG NUMBER': df['in\_kiang\_number'],

'NAME': df['in\_name'],

'CURRENT GA': df['Current Gestational Age'],

'VILLAGE': df['in\_village'],

'COMPOUND': df['in\_compound'],

'CONTACT1': df['in\_contact1'],

'CONTACT2': df['in\_contact2'],

'CONTACT3': df['in\_contact3'],

'CONTACT4': df['in\_contact4'],

'DUE DATE': df['Due Date'],

'LAST DUE DATE': df['Last Date'],

'Visit': df['Visit']

})

# Function to calculate the target due date

def calculate\_target\_due\_date():

current\_date = datetime.date.today()

next\_monday = current\_date + datetime.timedelta(days=(0 - current\_date.weekday() + 7) % 7)

target\_due\_date = next\_monday + datetime.timedelta(days=3)

return pd.to\_datetime(target\_due\_date)

# Function to filter call list based on due date

def filter\_call\_list(df\_due\_dates, target\_due\_date):

return df\_due\_dates[df\_due\_dates['DUE DATE'] <= target\_due\_date]

# Function to save DataFrame to CSV with a timestamp

def save\_to\_csv(df, base\_filename):

now = datetime.datetime.now()

timestamp = now.strftime('%Y%m%d\_%H%M%S')

filename = f'{base\_filename}\_{timestamp}.csv'

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

# Example usage

reschedules\_df = ... # Define or load your DataFrames here

wk20\_df = ...

wk28\_df = ...

wk36\_df = ...

month1\_df = ...

month2\_df = ...

month3\_df = ...

month4\_df = ...

month5\_df = ...

month6\_df = ...

month7\_df = ...

month8\_df = ...

month9\_df = ...

month10\_df = ...

month11\_df = ...

month12\_df = ...

sen\_log\_data = ... # Load sensitization log data

# Generate final due dates DataFrame

df\_due\_dates1 = concatenate\_due\_dates(

reschedules\_df, wk20\_df, wk28\_df, wk36\_df,

month1\_df, month2\_df, month3\_df, month4\_df,

month5\_df, month6\_df, month7\_df, month8\_df,

month9\_df, month10\_df, month11\_df, month12\_df

)

# Merge with sensitization data

df\_due\_dates = merge\_with\_sensitization(df\_due\_dates1, sen\_log\_data)

# Format the final DataFrame

df\_due\_dates\_formatted = format\_final\_df(df\_due\_dates)

# Calculate the target due date

target\_due\_date = calculate\_target\_due\_date()

# Filter the call list

df\_call\_list = filter\_call\_list(df\_due\_dates\_formatted, target\_due\_date)

# Save the call list to a CSV file

save\_to\_csv(df\_call\_list, 'CallList')