bd_hayai

October 13, 2024

[2]:
```
import pandas as pd
```

[28]:
```
# Recieve CSV file from SSMS. Add a header
df_from_ssms = pd.read_csv("meks.csv", header=None)

# Get rid of all cols except for 1 and 2
df_from_ssms.drop([2, 3, 4], axis=1, inplace=True)

df_from_ssms.head(10)
```

[28]:

	0	1
0	266943	108992
1	266696	108993
2	266626	108994
3	266704	108995
4	266574	108284
5	267112	108283
6	267263	108279
7	266888	108285
8	267259	110254
9	267147	110253

[29]:
```
# Add column names.
# 'MEK' - Master entity key
# 'CN' - Control number
df_from_ssms.rename(columns={0: 'MEK', 1: 'CN_'}, inplace=True)

# Prepend two zeros to the front of the CNs. We can use apply() and zfill() fns
# converting each value to a string
# and then prepending 0s until the length is 8
df_from_ssms['CN_'] = df_from_ssms['CN_'].apply(lambda x: str(x).zfill(8))

df_from_ssms.head(10)
```

[29]:

	MEK	CN_
0	266943	00108992
1	266696	00108993
2	266626	00108994

```
3  266704  00108995
4  266574  00108284
5  267112  00108283
6  267263  00108279
7  266888  00108285
8  267259  00110254
9  267147  00110253
```

[30]:
```
# Retrieve CSV file sent in by end-user
df_from_enduser = pd.read_csv("end_user_sheet.csv")

df_from_enduser['CN'] = df_from_enduser['CN'].apply(lambda x: str(x).zfill(8))

# Use column indexing to remove everything except for the cols
# 5 - Control number
# 6 - Formatted cycle start date
# 7 - RTLS tag no.

df_from_enduser = df_from_enduser.iloc[:, [4, 5, 6]]

ssms_sz = df_from_ssms.shape[0]
end_user_sz = df_from_enduser.shape[0]

# Display sizes of data frames

print(f"Size of ssmss df is {ssms_sz}")
print(f"Size of end user df is {end_user_sz}")

df_from_enduser.head(10)
```

```
Size of ssmss df is 20
Size of end user df is 20
```

[30]:

	CN	CYCLESTARTDATE	RTLS
0	00108992	1991-09-01	000CCC14E6F6
1	00108993	1991-09-01	000CCC11B021
2	00108994	1991-09-01	000CCC11AE7B
3	00108995	1991-09-01	000CCC119C83
4	00108284	1991-09-01	000CCC120345
5	00108283	1991-09-01	000CCC11DEF1
6	00108279	1991-09-01	000CCC12052F
7	00108285	1991-09-01	000CCC11AE53
8	00110254	1991-09-01	000CCC11FC99
9	00110253	1991-09-01	000CCC120A4B

[31]:
```
# Concatenate the two Dataframes
df_hayai = pd.concat([df_from_enduser, df_from_ssms], axis=1)
```

```
# Change data type of MEK col to int. It will become a float by default for␣
 ↪reasons I still don't rly get
df_hayai['MEK'] = df_hayai['MEK'].astype('int')

df_hayai.head(10)
```

[31]:

	CN	CYCLESTARTDATE	RTLS	MEK	CN_
0	00108992	1991-09-01	000CCC14E6F6	266943	00108992
1	00108993	1991-09-01	000CCC11B021	266696	00108993
2	00108994	1991-09-01	000CCC11AE7B	266626	00108994
3	00108995	1991-09-01	000CCC119C83	266704	00108995
4	00108284	1991-09-01	000CCC120345	266574	00108284
5	00108283	1991-09-01	000CCC11DEF1	267112	00108283
6	00108279	1991-09-01	000CCC12052F	267263	00108279
7	00108285	1991-09-01	000CCC11AE53	266888	00108285
8	00110254	1991-09-01	000CCC11FC99	267259	00110254
9	00110253	1991-09-01	000CCC120A4B	267147	00110253

[32]:
```
def get_meks_from_ssms(control_numbers):
    # Generate individual CASE statements for each control number
    # The enumerate function gives us both the index (i) and value (cn) for␣
 ↪each iteration
    # We use an f-string to format each CASE statement with proper indentation
    case_statements = [f"        WHEN [ControlNo] = '{cn}' THEN {i+1}" for i,␣
 ↪cn in enumerate(control_numbers)]

    # Join all CASE statements into a single string, with each statement on a␣
 ↪new line
    case_block = "\n    ".join(case_statements)

    # Construct the full SQL command using an f-string
    # This allows us to embed our Python variables directly in the SQL string
    sql_command = f"""
    SELECT
        [EquipmentKey],
        [CN],
        [CycleDate],
        [CycleSetBy],
        [RTLSCode]
    FROM
        [URMCCEX3].[dbo].[Equipment]
    WHERE
        [ControlNo] IN (\n{', '.join([f"'{cn}'" for cn in control_numbers])}
        )
    ORDER BY
        CASE\n    {case_block}
    END;
```

```
    """
    return sql_command

# Initialize an empty list to store control numbers
control_numbers = []

# Iterate over each row in the df_hayai DataFrame
# The iterrows() method gives us both the index and the row data for each␣
 ↪iteration
for idx, row in df_from_enduser.iterrows():
    # Extract the 'CN' (Control Number) value from the current row
    cns = row['CN']

    # Add this Control Number to our list
    control_numbers.append(cns)

# Generate the SQL command using our list of control numbers
sql_command = get_meks_from_ssms(control_numbers)

# Print the generated SQL command to the console
print(sql_command)
```

```
    SELECT
        [EquipmentKey],
        [CN],
        [CycleDate],
        [CycleSetBy],
        [RTLSCode]
    FROM
        [URMCCEX3].[dbo].[Equipment]
    WHERE
        [ControlNo] IN (
'00108992', '00108993', '00108994', '00108995', '00108284', '00108283',
'00108279', '00108285', '00110254', '00110253', '00110273', '00110272',
'00111137', '00111136', '00111134', '00111135', '00111138', '00111139',
'00111142', '00111143'
        )
    ORDER BY
        CASE
            WHEN [ControlNo] = '00108992' THEN 1
            WHEN [ControlNo] = '00108993' THEN 2
            WHEN [ControlNo] = '00108994' THEN 3
            WHEN [ControlNo] = '00108995' THEN 4
            WHEN [ControlNo] = '00108284' THEN 5
            WHEN [ControlNo] = '00108283' THEN 6
            WHEN [ControlNo] = '00108279' THEN 7
            WHEN [ControlNo] = '00108285' THEN 8
```

```
            WHEN [ControlNo] = '00110254' THEN 9
            WHEN [ControlNo] = '00110253' THEN 10
            WHEN [ControlNo] = '00110273' THEN 11
            WHEN [ControlNo] = '00110272' THEN 12
            WHEN [ControlNo] = '00111137' THEN 13
            WHEN [ControlNo] = '00111136' THEN 14
            WHEN [ControlNo] = '00111134' THEN 15
            WHEN [ControlNo] = '00111135' THEN 16
            WHEN [ControlNo] = '00111138' THEN 17
            WHEN [ControlNo] = '00111139' THEN 18
            WHEN [ControlNo] = '00111142' THEN 19
            WHEN [ControlNo] = '00111143' THEN 20
    END;
```

[33]:
```
# Ensure that CN from dataframe and SSMS script are valid
cn_match = (df_hayai['CN'] == df_hayai['CN_']).all()

# Ensure no duplicate values
no_copies = (df_hayai['CN'].is_unique and df_hayai['CN_'].is_unique)

# Ensure no empty cells
no_empty_cells = not df_hayai[['CN', 'CN_']].isnull().values.any()

if cn_match:
    print("The CNs in the SSMS col and end_user column match!.")
else:
    print("The columns do not match.")

if no_copies:
    print("There are no duplicate values in both columns. All vals unique")
else:
    print("There are duplicates in one or both columns. There is an error␣
 ↪somewhere")

if no_empty_cells:
    print("There are no empty cells in the DataFrame.")
else:
    print("There are empty cells in the DataFrame.")
```

```
The CNs in the SSMS col and end_user column match!.
There are no duplicate values in both columns. All vals unique
There are no empty cells in the DataFrame.
```

[34]:
```
# We now generate SQL to be put into SSMS (SQL Server Management Studio)

def generate_sql_for_ssms(meks, dates, rtls_codes):
    # Ensure all input lists have the same length
```

```
    # This is a safety check to make sure we have matching data for each entry
    if not (len(meks) == len(dates) == len(rtls_codes)):
        raise ValueError("All input lists must have the same length")

    # Generate CASE statements for CycleDate
    # We create two list comprehensions.
    # One to create a WHEN-THEN statement for each MEK-date pair
    # One to create a WHEN-THEN statement for each MEK-rtls pair

    case_statements_for_meks = [
        f"        WHEN [EquipmentKey] = {mek} THEN '{date} 00:00:00.000'"
        for mek, date in zip(meks, dates)
    ]
    # Join all CASE statements into a single string, with each statement on a␣
↪new line
    case_block_meks = "\n    ".join(case_statements_for_meks)

    # Generate the IN clause for MasterEntityKey
    # We create a comma-separated list of MEKs, each wrapped in single quotes
    mek_list = ", ".join(f"'{mek}'" for mek in meks)

    case_statements_for_rtls = [
        f"        WHEN [EquipmentKey] = {mek} THEN '{rtls}'"
        for mek, rtls in zip(meks, rtls_codes)
    ]
    case_block_rtls = "\n".join(case_statements_for_rtls)

    mek_list = ", ".join(f"'{mek}'" for mek in meks)

    # Construct the full SQL command using an f-string
    # This allows us to embed our Python variables directly in the SQL string
    sql_command = f"""
    UPDATE [URMCCEX3].[dbo].[Equipment]
    SET
        CycleSetBy = 'Equipment',
        CycleDate = CASE\n    {case_block_meks}
        END,
    [RTLScode] = CASE\n{case_block_rtls}
    END
    WHERE
       [EquipmentKey] IN ({mek_list});
    """
    return sql_command

    # Return the complete SQL command as a string
    return sql_command
```

```python
# Initialize empty lists to store data from the DataFrame
master_entity_keys = []
cycle_start_dates = []
rtls_tag_codes = []

# Iterate over each row in the df_hayai DataFrame
# The iterrows() method gives us both the index and the row data for each␣
 ↪iteration
for idx, row in df_hayai.iterrows():
    # Extract the relevant values from the current row
    mek = row['MEK']  # Master Entity Key
    csd = row['CYCLESTARTDATE']  # Cycle Start Date
    rtls = row['RTLS']  # RTLS Tag Code

    # Append the extracted values to their respective lists
    master_entity_keys.append(mek)
    cycle_start_dates.append(csd)
    rtls_tag_codes.append(rtls)

# Generate the SQL command using our lists of MEKs, dates, and RTLS codes
sql_to_ssms = generate_sql_for_ssms(meks=master_entity_keys,␣
 ↪dates=cycle_start_dates, rtls_codes=rtls_tag_codes)

# Print the generated SQL command to the console
print(sql_to_ssms)
```

```
UPDATE [URMCCEX3].[dbo].[Equipment]
SET
    CycleSetBy = 'Equipment',
    CycleDate = CASE
       WHEN [EquipmentKey] = 266943 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266696 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266626 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266704 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266574 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267112 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267263 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266888 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267259 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267147 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267296 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266429 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 267027 THEN '1991-09-01 00:00:00.000'
       WHEN [EquipmentKey] = 266994 THEN '1991-10-01 00:00:00.000'
       WHEN [EquipmentKey] = 267033 THEN '1991-10-01 00:00:00.000'
       WHEN [EquipmentKey] = 267120 THEN '1991-10-01 00:00:00.000'
```

```
            WHEN [EquipmentKey] = 266643 THEN '1991-10-01 00:00:00.000'
            WHEN [EquipmentKey] = 266641 THEN '1991-10-01 00:00:00.000'
            WHEN [EquipmentKey] = 267050 THEN '1991-10-01 00:00:00.000'
            WHEN [EquipmentKey] = 267048 THEN '1991-10-01 00:00:00.000'
        END,
    [RTLScode] = CASE
        WHEN [EquipmentKey] = 266943 THEN '000CCC14E6F6'
        WHEN [EquipmentKey] = 266696 THEN '000CCC11B021'
        WHEN [EquipmentKey] = 266626 THEN '000CCC11AE7B'
        WHEN [EquipmentKey] = 266704 THEN '000CCC119C83'
        WHEN [EquipmentKey] = 266574 THEN '000CCC120345'
        WHEN [EquipmentKey] = 267112 THEN '000CCC11DEF1'
        WHEN [EquipmentKey] = 267263 THEN '000CCC12052F'
        WHEN [EquipmentKey] = 266888 THEN '000CCC11AE53'
        WHEN [EquipmentKey] = 267259 THEN '000CCC11FC99'
        WHEN [EquipmentKey] = 267147 THEN '000CCC120A4B'
        WHEN [EquipmentKey] = 267296 THEN '000CCC120650'
        WHEN [EquipmentKey] = 266429 THEN '000CCC1201A9'
        WHEN [EquipmentKey] = 267027 THEN '000CCC12036C'
        WHEN [EquipmentKey] = 266994 THEN '000CCC11FBAE'
        WHEN [EquipmentKey] = 267033 THEN '000CCC11B0B6'
        WHEN [EquipmentKey] = 267120 THEN '000CCC120963'
        WHEN [EquipmentKey] = 266643 THEN '000CCC119C64'
        WHEN [EquipmentKey] = 266641 THEN '000CCC11A01B'
        WHEN [EquipmentKey] = 267050 THEN '000CCC12015D'
        WHEN [EquipmentKey] = 267048 THEN '000CCC119BF1'
    END
    WHERE
       [EquipmentKey] IN ('266943', '266696', '266626', '266704', '266574',
'267112', '267263', '266888', '267259', '267147', '267296', '266429', '267027',
'266994', '267033', '267120', '266643', '266641', '267050', '267048');
```