Python Version: 3.12
MySql Version: 8.0
IDE: PyCharm

MySQL Credentials:

User: root
Password: --------
Database: buildtax1
Tables: buildingpermit,buildingpermit1,taxassessor

Python Libraries:

1. **Pandas**
2. **Numpy**
3. **Mysql.connector**
4. **Csv**
5. **Datetime**
6. **re**
7. **tkinter**

Python Script to insert data into the BuildingPermit table:-

```
import pandas as pd
import numpy as np
import mysql.connector
import csv
from datetime import datetime
import re

dtype_mapping = {
    '[ATTOM ID]': str,
    'PermitNumber': str
}

site_df = pd.read_csv("2002_Building.csv", encoding="ISO-8859-1",
sep=',', quotechar='"', quoting=csv.QUOTE_MINIMAL, on_bad_lines='skip',
dtype=dtype_mapping)
```

```
site_df.replace({np.nan: None}, inplace=True)

def parse_date(date_str):
    for fmt in ('%m/%d/%Y', '%Y-%m-%d'):
        try:
            return datetime.strptime(date_str, fmt).strftime('%Y-%m-%d')
        except (ValueError, TypeError):
            continue
    return None

date_columns = ['EffectiveDate', 'PublicationDate']
for col in date_columns:
    site_df[col] = site_df[col].apply(parse_date)

site_df['[ATTOM ID]'] = site_df['[ATTOM ID]'].astype(str)
site_df['PermitNumber'] = site_df['PermitNumber'].astype(str)

def convert_scientific_notation(value):
    if re.match(r'^\d+\.?\d*e[\+\-]?\d+$', value, re.IGNORECASE):
        return str(int(float(value)))
    return value

site_df['PermitNumber'] =
site_df['PermitNumber'].apply(convert_scientific_notation)


db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '-----',
    'database': 'buildtax1'
}


connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()


create_table_query = """
CREATE TABLE IF NOT EXISTS BuildingPermit (
    sequence_no INT AUTO_INCREMENT PRIMARY KEY,
    BuildingPermitID__c TEXT,
    Attom_Id__c1 VARCHAR(1000),
    Full_Address VARCHAR(1000),
    House_Number__c VARCHAR(1000),
    StreetDirection__c VARCHAR(30),
    street VARCHAR(100),
    StreetSuffix__c VARCHAR(100),
    StreetPostDirection__c VARCHAR(30),
    Address_UnitPrefix__c VARCHAR(100),
```

```
    Address_Unit__c VARCHAR(100),
    City VARCHAR(500),
    State VARCHAR(100),
    PostalCode INT,
    ZIP4__c INT,
    EffectiveDate__c DATETIME,
    PermitNumber__c VARCHAR(1000),
    Attom_Status__c VARCHAR(200),
    Description TEXT,
    Type__c VARCHAR(300),
    Sub_Type__c VARCHAR(300),
    Business_Name__c VARCHAR(500),
    HomeOwner__c VARCHAR(500),
    PublicationDate__c DATETIME
);
"""


cursor.execute(create_table_query)


insert_query = """
INSERT INTO BuildingPermit (
    BuildingPermitID__c, Attom_Id__c1, Full_Address, House_Number__c,
    StreetDirection__c, street, StreetSuffix__c, StreetPostDirection__c,
    Address_UnitPrefix__c, Address_Unit__c, City, State, PostalCode,
ZIP4__c,
    EffectiveDate__c, PermitNumber__c, Attom_Status__c, Description,
Type__c,
    Sub_Type__c, Business_Name__c, HomeOwner__c, PublicationDate__c
) VALUES (
    %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
    %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
    %s, %s, %s
)
"""


for index, row in site_df.iterrows():
    row['[ATTOM ID]'] = str(row['[ATTOM ID]'])
    row['PermitNumber'] = str(row['PermitNumber'])
    insert_values = tuple(row)
    try:
        cursor.execute(insert_query, insert_values)
    except mysql.connector.Error as err:
        print(f"Error at index {index} with data {insert_values}:
{err}")
        continue


connection.commit()
```

```
cursor.close()
connection.close()
```

Python Script to insert data into the BuildingPermit1 table:-

```
import pandas as pd
import numpy as np
import mysql.connector
import csv
from datetime import datetime
import re

dtype_mapping = {
    '[ATTOM ID]': str,
    'PermitNumber': str
}

site_df = pd.read_csv("2002_Building.csv", encoding="ISO-8859-1",
sep=',', quotechar='"', quoting=csv.QUOTE_MINIMAL, on_bad_lines='skip',
dtype=dtype_mapping)

site_df.replace({np.nan: None}, inplace=True)

def parse_date(date_str):
    for fmt in ('%m/%d/%Y', '%Y-%m-%d'):
        try:
            return datetime.strptime(date_str, fmt).strftime('%Y-%m-%d')
        except (ValueError, TypeError):
            continue
    return None

date_columns = ['EffectiveDate', 'PublicationDate']
for col in date_columns:
    site_df[col] = site_df[col].apply(parse_date)

site_df['[ATTOM ID]'] = site_df['[ATTOM ID]'].astype(str)
site_df['PermitNumber'] = site_df['PermitNumber'].astype(str)

def convert_scientific_notation(value):
    if re.match(r'^\d+\.?\d*e[\+\-]?\d+$', value, re.IGNORECASE):
```

```
        return str(int(float(value)))
    return value

site_df['PermitNumber'] =
site_df['PermitNumber'].apply(convert_scientific_notation)


db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '-----',
    'database': 'buildtax1'
}


connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()


create_table_query = """
CREATE TABLE IF NOT EXISTS BuildingPermit1 (
    sequence_no INT AUTO_INCREMENT PRIMARY KEY,
    BuildingPermitID__c TEXT,
    Attom_Id__c1 VARCHAR(1000),
    Full_Address VARCHAR(1000),
    House_Number__c VARCHAR(1000),
    StreetDirection__c VARCHAR(30),
    street VARCHAR(100),
    StreetSuffix__c VARCHAR(100),
    StreetPostDirection__c VARCHAR(30),
    Address_UnitPrefix__c VARCHAR(100),
    Address_Unit__c VARCHAR(100),
    City VARCHAR(500),
    State VARCHAR(100),
    PostalCode INT,
    ZIP4__c INT,
    EffectiveDate__c DATETIME,
    PermitNumber__c VARCHAR(1000),
    Attom_Status__c VARCHAR(200),
    Description TEXT,
    Type__c VARCHAR(300),
    Sub_Type__c VARCHAR(300),
    Business_Name__c VARCHAR(500),
    HomeOwner__c VARCHAR(500),
    PublicationDate__c DATETIME
);
"""


cursor.execute(create_table_query)
```

```
insert_query = """
INSERT INTO BuildingPermit1 (
   BuildingPermitID__c, Attom_Id__c1, Full_Address, House_Number__c,
   StreetDirection__c, street, StreetSuffix__c, StreetPostDirection__c,
   Address_UnitPrefix__c, Address_Unit__c, City, State, PostalCode,
ZIP4__c,
   EffectiveDate__c, PermitNumber__c, Attom_Status__c, Description,
Type__c,
   Sub_Type__c, Business_Name__c, HomeOwner__c, PublicationDate__c
) VALUES (
   %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
   %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
   %s, %s, %s
)
"""


for index, row in site_df.iterrows():
   row['[ATTOM ID]'] = str(row['[ATTOM ID]'])
   row['PermitNumber'] = str(row['PermitNumber'])
   insert_values = tuple(row)
   try:
       cursor.execute(insert_query, insert_values)
   except mysql.connector.Error as err:
       print(f"Error at index {index} with data {insert_values}:
{err}")
       continue


connection.commit()


cursor.close()
connection.close()
```

Python Script to insert data into the TaxAssessor table:-

```
import pandas as pd
import mysql.connector
import numpy as np

site_df = pd.read_csv("2002.csv")

site_df.replace({np.nan: None}, inplace=True)
```

```
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '-----',
    'database': 'buildtax1'
}

connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

create_table_query = """
CREATE TABLE IF NOT EXISTS TaxAssessor (
    Attom_Id__c int PRIMARY KEY,
    FirstName varchar(700),
    MiddleName varchar(700),
    LastName varchar(700),
    Owner_Type__c varchar(200)
);
"""

cursor.execute(create_table_query)

insert_query = """INSERT IGNORE INTO TaxAssessor (
    Attom_Id__c, FirstName, MiddleName, LastName, Owner_Type__c
) VALUES (%s, %s, %s, %s, %s)"""

data_to_insert = [tuple(row) for row in site_df.itertuples(index=False,
name=None)]

try:
    cursor.executemany(insert_query, data_to_insert)
    connection.commit()
except mysql.connector.Error as err:
    print(f"Error: {err}")

cursor.close()
connection.close()
```

Python Script to download Matching data from BuildingPermit and TaxAssessor table

```
import mysql.connector
import pandas as pd
import os
import tkinter as tk
from tkinter import filedialog
```

```
def get_folder_path():
    root = tk.Tk()
    root.withdraw()
    folder_path = filedialog.askdirectory()
    if not folder_path:
        raise ValueError("No folder selected.")
    return folder_path


folder_path = get_folder_path()

batch_size = 50000

connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='-----',
    database='buildtax1'
)

try:
    file_index = 1
    last_sequence_no = 0
    total_records_processed = 0

    while True:
        query = f"""
        SELECT bp1.*, ta.*
        FROM BuildingPermit bp1
        JOIN TaxAssessor ta ON ta.Attom_Id__c = bp1.Attom_Id__c1
        WHERE bp1.sequence_no > {last_sequence_no}
        ORDER BY bp1.sequence_no
        LIMIT {batch_size}
        """
        cursor = connection.cursor(dictionary=True)
        cursor.execute(query)
        result = cursor.fetchall()
        if not result:
            break

        df = pd.DataFrame(result)

        start_seq = total_records_processed + 1
        end_seq = total_records_processed + len(df)
        file_name = os.path.join(folder_path,
f'BuildingPermit({start_seq} to {end_seq}).csv')
        df.to_csv(file_name, index=False)

        print(f'{file_name} has been created with {len(df)} rows.')
```

```
        total_records_processed += len(df)
        file_index += 1
        last_sequence_no = df['sequence_no'].iloc[-1]

    cursor.close()

finally:
    connection.close()
```

Python Script to download Matching data from BuildingPermit1 and TaxAssessor table

```
import mysql.connector
import pandas as pd
import os
import tkinter as tk
from tkinter import filedialog


def get_folder_path():
    root = tk.Tk()
    root.withdraw()
    folder_path = filedialog.askdirectory()
    if not folder_path:
        raise ValueError("No folder selected.")
    return folder_path


folder_path = get_folder_path()

batch_size = 50000

connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='-----',
    database='buildtax1'
)

try:
    file_index = 1
    last_sequence_no = 0
    total_records_processed = 0

    while True:
        query = f"""
        SELECT bp1.*, ta.*
        FROM BuildingPermit1 bp1
```

```
        JOIN TaxAssessor ta ON ta.Attom_Id__c = bp1.Attom_Id__c1
        WHERE bp1.sequence_no > {last_sequence_no}
        ORDER BY bp1.sequence_no
        LIMIT {batch_size}
        """
        cursor = connection.cursor(dictionary=True)
        cursor.execute(query)
        result = cursor.fetchall()
        if not result:
            break

        df = pd.DataFrame(result)

        start_seq = total_records_processed + 1
        end_seq = total_records_processed + len(df)
        file_name = os.path.join(folder_path,
f'BuildingPermit1({start_seq} to {end_seq}).csv')
        df.to_csv(file_name, index=False)

        print(f'{file_name} has been created with {len(df)} rows.')

        total_records_processed += len(df)
        file_index += 1
        last_sequence_no = df['sequence_no'].iloc[-1]

    cursor.close()

finally:
    connection.close()
```

Python Script to download Non Matching data from BuildingPermit and TaxAssessor table

```
import mysql.connector
import pandas as pd
import os
import tkinter as tk
from tkinter import filedialog

def get_folder_path():
    root = tk.Tk()
    root.withdraw()
    folder_path = filedialog.askdirectory()
```

```
    if not folder_path:
        raise ValueError("No folder selected.")
    return folder_path

folder_path = get_folder_path()

batch_size = 50000

connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='-----',
    database='buildtax1'
)

try:
    file_index = 1
    last_sequence_no = 0
    total_records_processed = 0

    while True:
        query = f"""
        SELECT bp1.*, ta.*
        FROM BuildingPermit bp1
        LEFT JOIN TaxAssessor ta ON ta.Attom_Id__c = bp1.Attom_Id__c1
        WHERE ta.Attom_Id__c IS NULL AND bp1.sequence_no >
{last_sequence_no}
        ORDER BY bp1.sequence_no
        LIMIT {batch_size}
        """
        cursor = connection.cursor(dictionary=True)
        cursor.execute(query)
        result = cursor.fetchall()
        if not result:
            break

        df = pd.DataFrame(result)

        start_seq = total_records_processed + 1
        end_seq = total_records_processed + len(df)
        file_name = os.path.join(folder_path,
f'BuildingPermit({start_seq} to {end_seq}).csv')
        df.to_csv(file_name, index=False)

        print(f'{file_name} has been created with {len(df)} rows.')

        total_records_processed += len(df)
        file_index += 1
        last_sequence_no = df['sequence_no'].iloc[-1]

    cursor.close()
```

```
finally:
    connection.close()
```

Python Script to download Non Matching data from BuildingPermit1 and TaxAssessor table

```
import mysql.connector
import pandas as pd
import os
import tkinter as tk
from tkinter import filedialog

def get_folder_path():
    root = tk.Tk()
    root.withdraw()
    folder_path = filedialog.askdirectory()
    if not folder_path:
        raise ValueError("No folder selected.")
    return folder_path

folder_path = get_folder_path()

batch_size = 50000

connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='-----',
    database='buildtax1'
)

try:
    file_index = 1
    last_sequence_no = 0
    total_records_processed = 0

    while True:
        query = f"""
        SELECT bp1.*, ta.*
```

```
        FROM BuildingPermit1 bp1
        LEFT JOIN TaxAssessor ta ON ta.Attom_Id__c = bp1.Attom_Id__c1
        WHERE ta.Attom_Id__c IS NULL AND bp1.sequence_no >
{last_sequence_no}
        ORDER BY bp1.sequence_no
        LIMIT {batch_size}
        """
        cursor = connection.cursor(dictionary=True)
        cursor.execute(query)
        result = cursor.fetchall()
        if not result:
            break

        df = pd.DataFrame(result)

        start_seq = total_records_processed + 1
        end_seq = total_records_processed + len(df)
        file_name = os.path.join(folder_path,
f'BuildingPermit1({start_seq} to {end_seq}).csv')
        df.to_csv(file_name, index=False)

        print(f'{file_name} has been created with {len(df)} rows.')

        total_records_processed += len(df)
        file_index += 1
        last_sequence_no = df['sequence_no'].iloc[-1]

    cursor.close()

finally:
    connection.close()
```