Exploring Building Permits DataSet

CS520-2023Fall   Project Presention

Group#9   Jie Xu

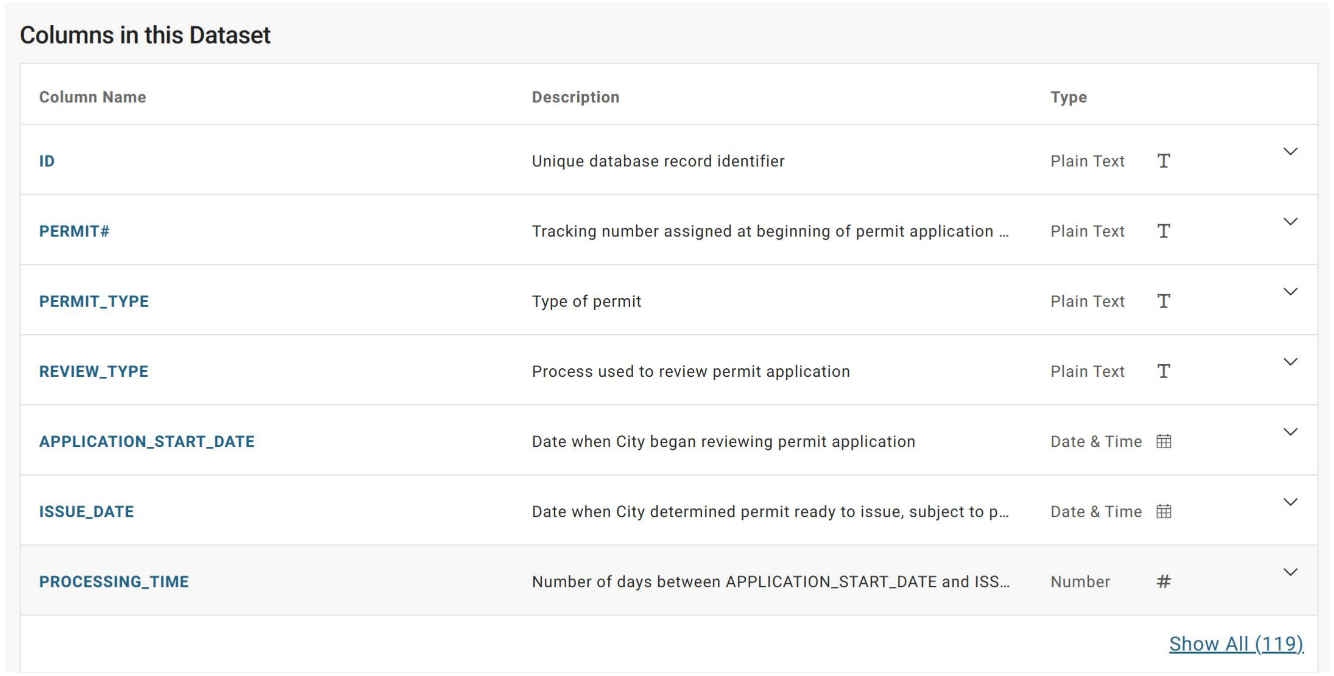
DataSet information

Building Permits

- This dataset includes information about currently-valid building permits issued by the City of Chicago from 2006 to the present
- https://data.cityofchicago.org/resource/ydr8-5enu.csv

DataSet Details

- The contents of the data set include building permit types, issuance dates, fees, contact information and geographic information (ward, community area and census tract)



Columns in this Dataset

Column Name	Description	Type
ID	Unique database record identifier	Plain Text T
PERMIT#	Tracking number assigned at beginning of permit application ...	Plain Text T
PERMIT_TYPE	Type of permit	Plain Text T
REVIEW_TYPE	Process used to review permit application	Plain Text T
APPLICATION_START_DATE	Date when City began reviewing permit application	Date & Time
ISSUE_DATE	Date when City determined permit ready to issue, subject to p...	Date & Time
PROCESSING_TIME	Number of days between APPLICATION_START_DATE and ISS...	Number #

Show All (119)

Issue Summary

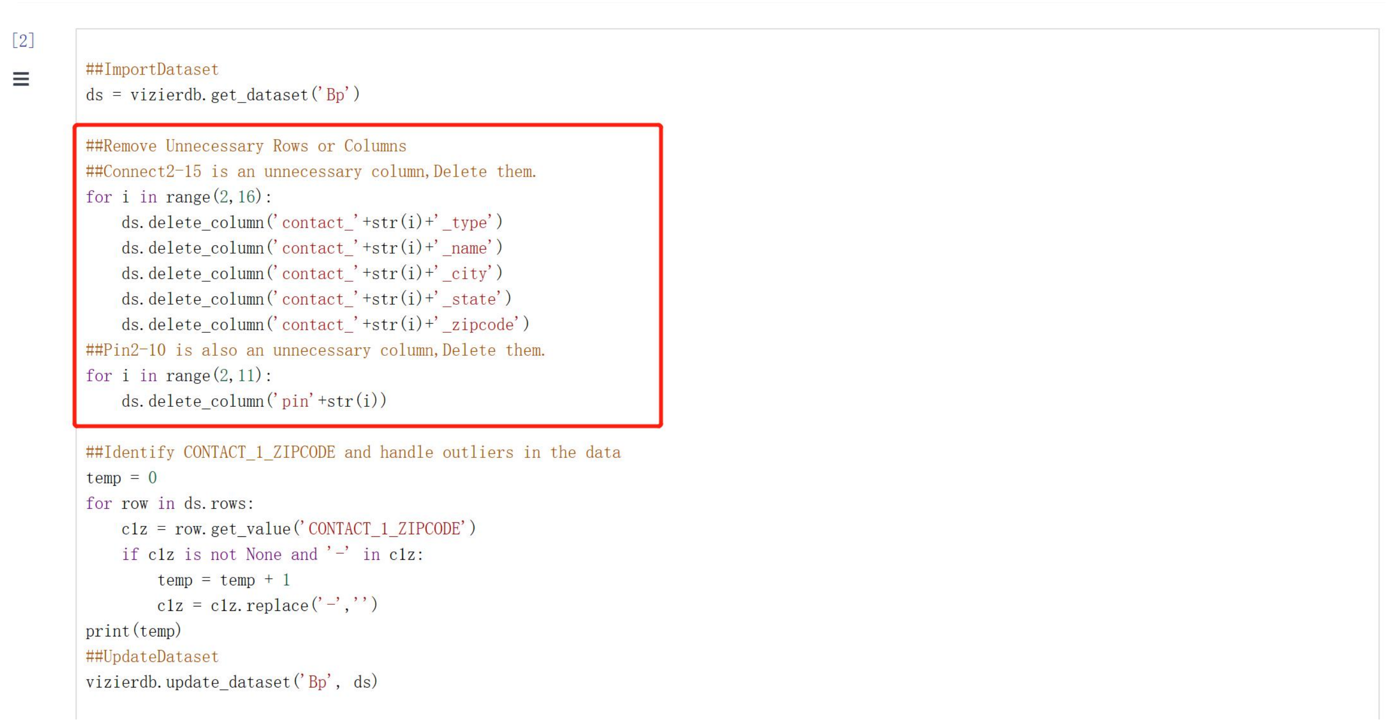
a. Remove Unnecessary Rows or Columns
b. Outlier Detection and Handling
c. Value is incorrect
d. Handle Missing Values

Issue

A.Remove Unnecessary Rows or Columns

Solution

- Connect2-15 is an unnecessary column,Delete them.
- Pin2-10 is also an unnecessary column,Delete them.

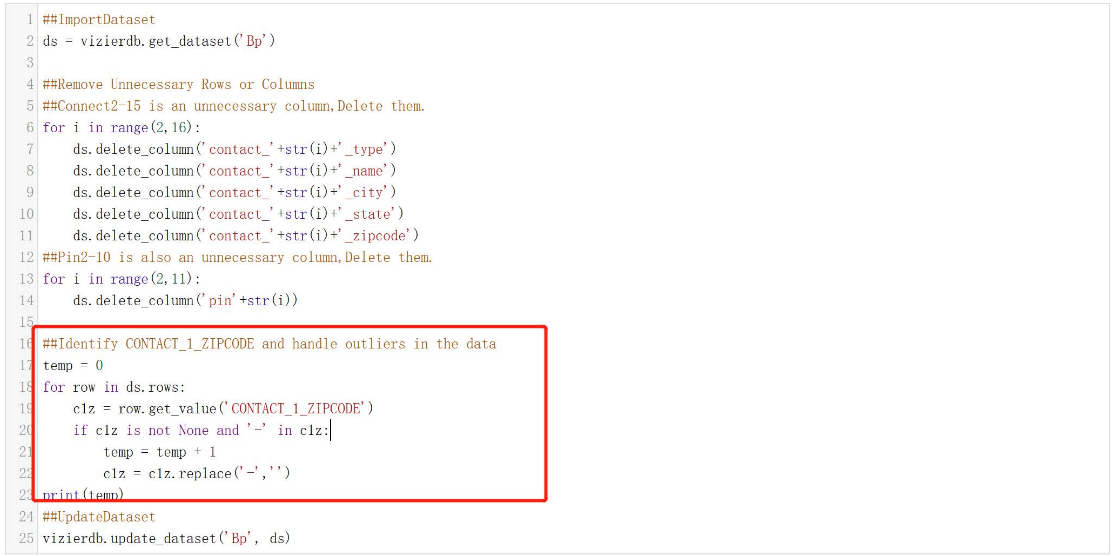


```
[2]
##ImportDataset
ds = vizierdb.get_dataset('Bp')

##Remove Unnecessary Rows or Columns
##Connect2-15 is an unnecessary column,Delete them.
for i in range(2,16):
    ds.delete_column('contact_'+str(i)+'_type')
    ds.delete_column('contact_'+str(i)+'_name')
    ds.delete_column('contact_'+str(i)+'_city')
    ds.delete_column('contact_'+str(i)+'_state')
    ds.delete_column('contact_'+str(i)+'_zipcode')
##Pin2-10 is also an unnecessary column,Delete them.
for i in range(2,11):
    ds.delete_column('pin'+str(i))

##Identify CONTACT_1_ZIPCODE and handle outliers in the data
temp = 0
for row in ds.rows:
    clz = row.get_value('CONTACT_1_ZIPCODE')
    if clz is not None and '-' in clz:
        temp = temp + 1
        clz = clz.replace('-','')
print(temp)
##UpdateDataset
vizierdb.update_dataset('Bp', ds)
```

Issue

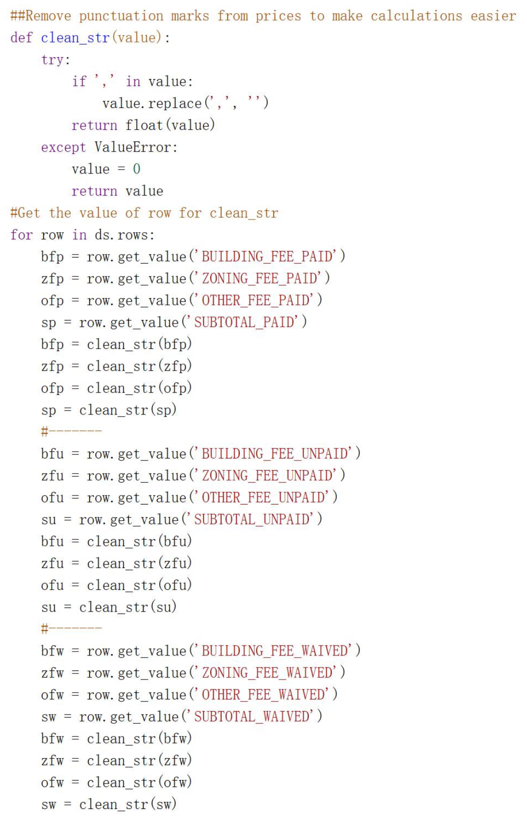
B.Outlier Detection and Handling

- Solution
- There are symbols like ‘-’ in CONTACT_1_ZIPCODE that need to be removed



```
##ImportDataset
ds = vizierdb.get_dataset('Bp')

##Remove Unnecessary Rows or Columns
##Connect2-15 is an unnecessary column,Delete them.
for i in range(2,16):
    ds.delete_column('contact_'+str(i)+'_type')
    ds.delete_column('contact_'+str(i)+'_name')
    ds.delete_column('contact_'+str(i)+'_city')
    ds.delete_column('contact_'+str(i)+'_state')
    ds.delete_column('contact_'+str(i)+'_zipcode')
##Pin2-10 is also an unnecessary column,Delete them.
for i in range(2,11):
    ds.delete_column('pin'+str(i))

##Identify CONTACT_1_ZIPCODE and handle outliers in the data
temp = 0
for row in ds.rows:
    clz = row.get_value('CONTACT_1_ZIPCODE')
    if clz is not None and '-' in clz:
        temp = temp + 1
        clz = clz.replace('-','')
print(temp)
##UpdateDataset
vizierdb.update_dataset('Bp', ds)
```

- There are punctuation marks like ’ ,’ in the price. Remove the symbol to facilitate subsequent calculations.



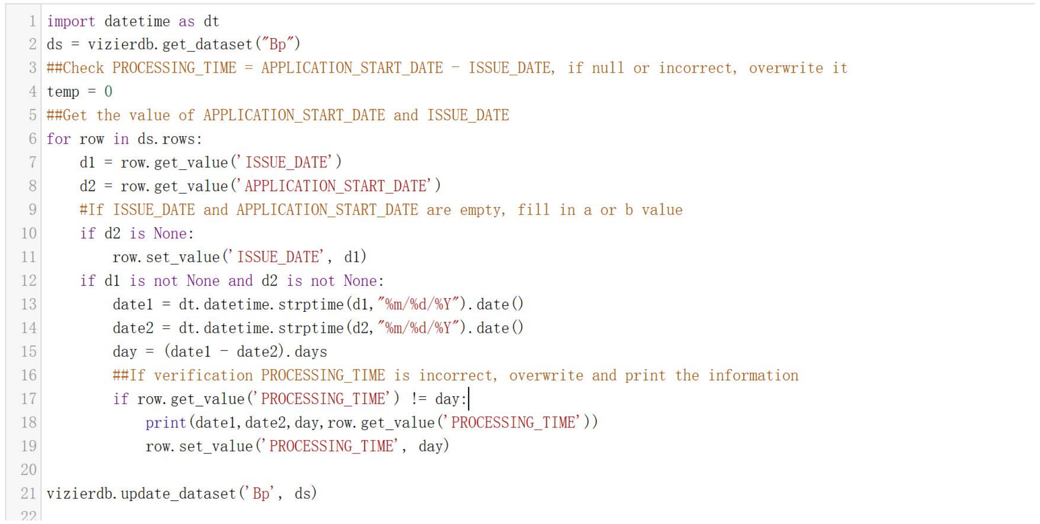
```
##Remove punctuation marks from prices to make calculations easier
def clean_str(value):
    try:
        if ',' in value:
            value.replace(',', '')
        return float(value)
    except ValueError:
        value = 0
        return value
#Get the value of row for clean_str
for row in ds.rows:
    bfp = row.get_value('BUILDING_FEE_PAID')
    zfp = row.get_value('ZONING_FEE_PAID')
    ofp = row.get_value('OTHER_FEE_PAID')
    sp = row.get_value('SUBTOTAL_PAID')
    bfp = clean_str(bfp)
    zfp = clean_str(zfp)
    ofp = clean_str(ofp)
    sp = clean_str(sp)
    #-------
    bfu = row.get_value('BUILDING_FEE_UNPAID')
    zfu = row.get_value('ZONING_FEE_UNPAID')
    ofu = row.get_value('OTHER_FEE_UNPAID')
    su = row.get_value('SUBTOTAL_UNPAID')
    bfu = clean_str(bfu)
    zfu = clean_str(zfu)
    ofu = clean_str(ofu)
    su = clean_str(su)
    #-------
    bfw = row.get_value('BUILDING_FEE_WAIVED')
    zfw = row.get_value('ZONING_FEE_WAIVED')
    ofw = row.get_value('OTHER_FEE_WAIVED')
    sw = row.get_value('SUBTOTAL_WAIVED')
    bfw = clean_str(bfw)
    zfw = clean_str(zfw)
    ofw = clean_str(ofw)
    sw = clean_str(sw)
```

Issue

C. Value is incorrect

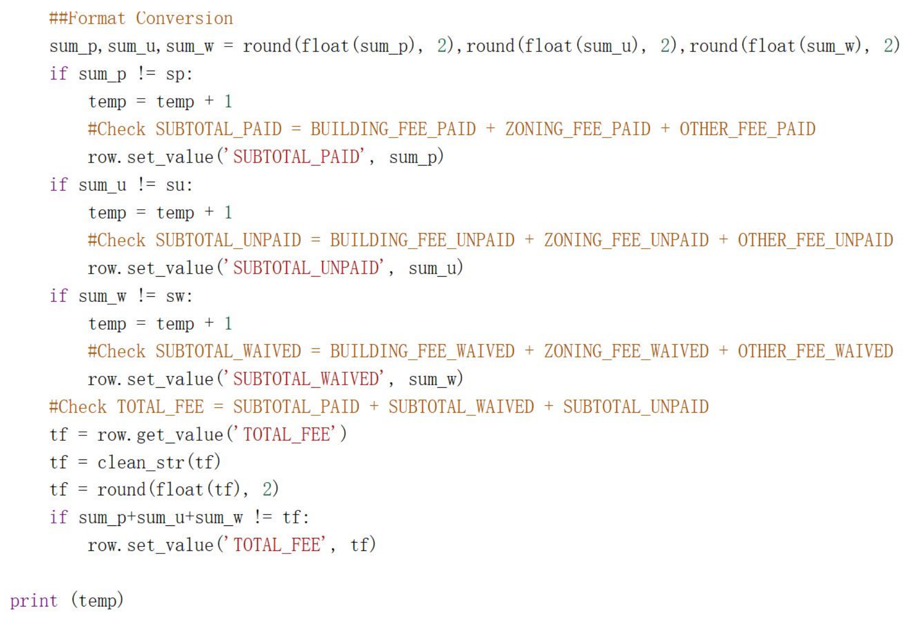
Solution

- Check whether PROCESSING_TIME is equal to APPLICATION_START_DATE + ISSUE_DATE,and handle the null value



```
import datetime as dt
ds = vizierdb.get_dataset("Bp")
##Check PROCESSING_TIME = APPLICATION_START_DATE - ISSUE_DATE, if null or incorrect, overwrite it
temp = 0
##Get the value of APPLICATION_START_DATE and ISSUE_DATE
for row in ds.rows:
    d1 = row.get_value('ISSUE_DATE')
    d2 = row.get_value('APPLICATION_START_DATE')
    #If ISSUE_DATE and APPLICATION_START_DATE are empty, fill in a or b value
    if d2 is None:
        row.set_value('ISSUE_DATE', d1)
    if d1 is not None and d2 is not None:
        date1 = dt.datetime.strptime(d1,"%m/%d/%Y").date()
        date2 = dt.datetime.strptime(d2,"%m/%d/%Y").date()
        day = (date1 - date2).days
        ##If verification PROCESSING_TIME is incorrect, overwrite and print the information
        if row.get_value('PROCESSING_TIME') != day:
            print(date1,date2,day,row.get_value('PROCESSING_TIME'))
            row.set_value('PROCESSING_TIME', day)

vizierdb.update_dataset('Bp', ds)
```

- Check SUBTOTAL_PAID = BUILDING_FEE_PAID + ZONING_FEE_PAID + OTHER_FEE_PAID
- Check SUBTOTAL_UNPAID = BUILDING_FEE_UNPAID + ZONING_FEE_UNPAID + OTHER_FEE_UNPAID
- Check SUBTOTAL_WAIVED = BUILDING_FEE_WAIVED + ZONING_FEE_WAIVED + OTHER_FEE_WAIVED
- Check TOTAL_FEE = SUBTOTAL_PAID + SUBTOTAL_WAIVED + SUBTOTAL_UNPAID

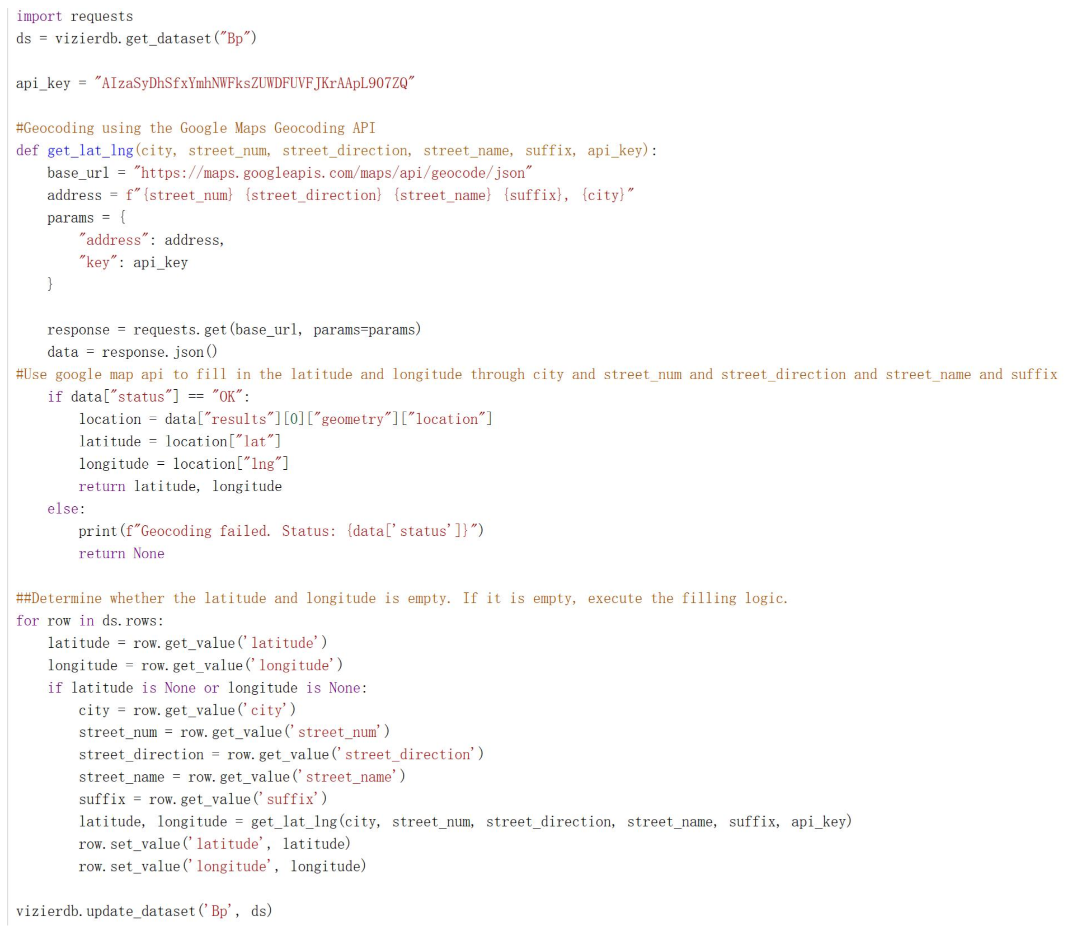


```
    ##Format Conversion
    sum_p,sum_u,sum_w = round(float(sum_p), 2),round(float(sum_u), 2),round(float(sum_w), 2)
    if sum_p != sp:
        temp = temp + 1
        #Check SUBTOTAL_PAID = BUILDING_FEE_PAID + ZONING_FEE_PAID + OTHER_FEE_PAID
        row.set_value('SUBTOTAL_PAID', sum_p)
    if sum_u != su:
        temp = temp + 1
        #Check SUBTOTAL_UNPAID = BUILDING_FEE_UNPAID + ZONING_FEE_UNPAID + OTHER_FEE_UNPAID
        row.set_value('SUBTOTAL_UNPAID', sum_u)
    if sum_w != sw:
        temp = temp + 1
        #Check SUBTOTAL_WAIVED = BUILDING_FEE_WAIVED + ZONING_FEE_WAIVED + OTHER_FEE_WAIVED
        row.set_value('SUBTOTAL_WAIVED', sum_w)
    #Check TOTAL_FEE = SUBTOTAL_PAID + SUBTOTAL_WAIVED + SUBTOTAL_UNPAID
    tf = row.get_value('TOTAL_FEE')
    tf = clean_str(tf)
    tf = round(float(tf), 2)
    if sum_p+sum_u+sum_w != tf:
        row.set_value('TOTAL_FEE', tf)

print (temp)
```

Issue

D. Handle Missing Values

- Solution
- Check the longitude and latitude. If it is empty, use the Google Map API to query the longitude and latitude through city, street and other information.



```python
import requests
ds = vizierdb.get_dataset("Bp")

api_key = "AIzaSyDhSfxYmhNWFksZUWDFUVFJKrAApL907ZQ"

#Geocoding using the Google Maps Geocoding API
def get_lat_lng(city, street_num, street_direction, street_name, suffix, api_key):
    base_url = "https://maps.googleapis.com/maps/api/geocode/json"
    address = f"{street_num} {street_direction} {street_name} {suffix}, {city}"
    params = {
        "address": address,
        "key": api_key
    }

    response = requests.get(base_url, params=params)
    data = response.json()
#Use google map api to fill in the latitude and longitude through city and street_num and street_direction and street_name and suffix
    if data["status"] == "OK":
        location = data["results"][0]["geometry"]["location"]
        latitude = location["lat"]
        longitude = location["lng"]
        return latitude, longitude
    else:
        print(f"Geocoding failed. Status: {data['status']}")
        return None

##Determine whether the latitude and longitude is empty. If it is empty, execute the filling logic.
for row in ds.rows:
    latitude = row.get_value('latitude')
    longitude = row.get_value('longitude')
    if latitude is None or longitude is None:
        city = row.get_value('city')
        street_num = row.get_value('street_num')
        street_direction = row.get_value('street_direction')
        street_name = row.get_value('street_name')
        suffix = row.get_value('suffix')
        latitude, longitude = get_lat_lng(city, street_num, street_direction, street_name, suffix, api_key)
        row.set_value('latitude', latitude)
        row.set_value('longitude', longitude)

vizierdb.update_dataset('Bp', ds)
```

Challenge

- Find relationships between attributes
- Correct format and value range of data
- Fill longitude and latitude through google map

Thank You