Written assesment for Advertisement Number: 43680

Job Title & Classification: Leaders in Training Program - Data Science Stream (P3)

Submitted by : Harpreet Kaur Bargota

Software or analytics tool used: Jupyter Notebook, Python programming language

Libraries used :Pandas, Numpy, glob, Matplotlib, Seaborn,

In [3]:
```
# Import the libraries
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
from shapely.geometry import Point
import ast
import matplotlib.pyplot as plt
import seaborn as sns
```

API endpoint used to get the data from the link

dataset_link= "https://data.lacity.org/City-Infrastructure-Service-Requests/Building-and-Safety-Inspections/9w5z-rg2h/about_data"

data= "https://data.lacity.org/resource/9w5z-rg2h.json"

In [5]:
```
#dataset_link= "https://data.lacity.org/City-Infrastructure-Service-Requests/Buildi
data= "https://data.lacity.org/resource/9w5z-rg2h.json"
#data=r"C:\Users\harvi\Downloads\city-of-los-angeles__link_requests__last30.2025-01

#read the .json file
df = pd.read_json(data)
```

Check the first 5 and last 5 rows

In [7]:
```
print(df.head(5))  # Display first top 5 rows
print ("--------------------------------------------------------------------------
print (df.tail(5))
```

```
                    address             permit   permit_status  \
0  10000 W SANTA MONICA BLVD  14044 10000 02293          Issued
1        1000 S SANTA FE AVE  15016 10000 18196  Permit Finaled
2       3680 N BUENA PARK DR  15014 10000 04931          Issued
3     1001 N LINDENWOOD LANE  16042 90000 14712  Permit Finaled
4          2836 S ANCHOR AVE  15016 20001 17211     CofO Issued

          inspection_date         inspection inspection_result  \
0  2016-07-20T00:00:00.000  Rough-Ventilation  Partial Approval
1  2016-07-22T00:00:00.000    Smoke Detectors    Insp Cancelled
2  2016-07-18T00:00:00.000         Insulation          Approved
3  2016-07-20T00:00:00.000              Final    Permit Finaled
4  2016-07-18T00:00:00.000         Inspection    Permit Finaled

                                           lat_lon  \
0  {'latitude': '34.06364', 'longitude': '-118.41...
1  {'latitude': '34.03143', 'longitude': '-118.22...
2  {'latitude': '34.13745', 'longitude': '-118.38...
3  {'latitude': '34.07732', 'longitude': '-118.48...
4  {'latitude': '34.03878', 'longitude': '-118.39...

   :@computed_region_qz3q_ghft  :@computed_region_kqwf_mjcx  \
0                        24032                            6
1                        23082                            9
2                         8492                            5
3                        23680                           10
4                        24029                            6

   :@computed_region_k96s_3jcv  :@computed_region_tatf_ua23  \
0                          870                          739
1                          533                         1287
2                          351                         1353
3                          830                          780
4                          872                         1100

   :@computed_region_ur2y_g4cx  :@computed_region_2dna_qi2s
0                          9.0                         75.0
1                          NaN                         76.0
2                          NaN                         84.0
3                          NaN                          NaN
4                          9.0                         95.0
--------------------------------------------------------------------------------
------------------------
                   address             permit permit_status  \
995    8158 W BEVERLY BLVD  16042 10000 13656        Issued
996          215 E 85TH ST  16041 30000 15736        Issued
997  4630 N SANTA LUCIA DR  16016 20000 16867        Issued
998    12049 W JUNIETTE ST  16014 10000 02398        Issued
999   12360 W OSBORNE PL 5  13010 20000 04166        Issued

             inspection_date                 inspection inspection_result  \
995  2016-07-22T00:00:00.000               Underground    Insp Scheduled
996  2016-07-22T00:00:00.000                     Final    Insp Scheduled
997  2016-07-22T00:00:00.000               Wood Frame          Approved
998  2016-07-21T00:00:00.000            PLUMBING-Rough  Partial Approval
999  2016-07-21T00:00:00.000  Verify Sprinkler Sign Off          Approved
```

```
                                                  lat_lon  \
995  {'latitude': '34.07591', 'longitude': '-118.36...
996  {'latitude': '33.96111', 'longitude': '-118.27...
997  {'latitude': '34.15204', 'longitude': '-118.60...
998  {'latitude': '33.98364', 'longitude': '-118.40...
999  {'latitude': '34.25922', 'longitude': '-118.40...

     :@computed_region_qz3q_ghft  :@computed_region_kqwf_mjcx  \
995                        23679                            6
996                        22352                           13
997                        19346                            4
998                        24678                           10
999                        18907                            1

     :@computed_region_k96s_3jcv  :@computed_region_tatf_ua23  \
995                          629                         1095
996                          795                         1001
997                          314                          653
998                          924                         1139
999                           32                          330

     :@computed_region_ur2y_g4cx  :@computed_region_2dna_qi2s
995                          NaN                         26.0
996                          7.0                         45.0
997                         46.0                         49.0
998                         10.0                         85.0
999                          NaN                         11.0
```

Check the basic information

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   address                      1000 non-null   object
 1   permit                       1000 non-null   object
 2   permit_status                1000 non-null   object
 3   inspection_date              1000 non-null   object
 4   inspection                   1000 non-null   object
 5   inspection_result            1000 non-null   object
 6   lat_lon                      1000 non-null   object
 7   :@computed_region_qz3q_ghft  1000 non-null   int64
 8   :@computed_region_kqwf_mjcx  1000 non-null   int64
 9   :@computed_region_k96s_3jcv  1000 non-null   int64
 10  :@computed_region_tatf_ua23  1000 non-null   int64
 11  :@computed_region_ur2y_g4cx  351 non-null    float64
 12  :@computed_region_2dna_qi2s  954 non-null    float64
dtypes: float64(2), int64(4), object(7)
memory usage: 101.7+ KB
```

Check for the names of columns

In [11]:
```
df.columns
```

Out[11]:
```
Index(['address', 'permit', 'permit_status', 'inspection_date', 'inspection',
       'inspection_result', 'lat_lon', ':@computed_region_qz3q_ghft',
       ':@computed_region_kqwf_mjcx', ':@computed_region_k96s_3jcv',
       ':@computed_region_tatf_ua23', ':@computed_region_ur2y_g4cx',
       ':@computed_region_2dna_qi2s'],
      dtype='object')
```

Check for the missing values

In [13]:
```
print(df.isnull().sum())  # Count of missing values per column
```

```
address                          0
permit                           0
permit_status                    0
inspection_date                  0
inspection                       0
inspection_result                0
lat_lon                          0
:@computed_region_qz3q_ghft      0
:@computed_region_kqwf_mjcx      0
:@computed_region_k96s_3jcv      0
:@computed_region_tatf_ua23      0
:@computed_region_ur2y_g4cx    649
:@computed_region_2dna_qi2s     46
dtype: int64
```

check datatypes

In [15]:
```
print(df.dtypes)
```

```
address                         object
permit                          object
permit_status                   object
inspection_date                 object
inspection                      object
inspection_result               object
lat_lon                         object
:@computed_region_qz3q_ghft      int64
:@computed_region_kqwf_mjcx      int64
:@computed_region_k96s_3jcv      int64
:@computed_region_tatf_ua23      int64
:@computed_region_ur2y_g4cx    float64
:@computed_region_2dna_qi2s    float64
dtype: object
```

In [ ]:
```

```

Question1: Make a table and a visualization showing an interesting characteristic of the permit and

inspection dataset .

In [17]:
```
df1=df[['permit','permit_status', 'inspection_date', 'inspection', 'inspection_resu
df1.head(2)
```

Out[17]:

	permit	permit_status	inspection_date	inspection	inspection_result
0	14044 10000 02293	Issued	2016-07-20T00:00:00.000	Rough-Ventilation	Partial Approval
1	15016 10000 18196	Permit Finaled	2016-07-22T00:00:00.000	Smoke Detectors	Insp Cancelled

In [18]:
```
# Loop through each column and print the unique values
for column in df1.columns:
    unique_values = df[column].unique()
    print(f"Unique values in '{column}':")
    print(unique_values)
    print("\n")
```

Unique values in 'permit':
['14044 10000 02293' '15016 10000 18196' '15014 10000 04931'
 '16042 90000 14712' '15016 20001 17211' '16042 10000 12648'
 '16016 20000 07926' '01020 10000 02808' '16042 10000 11285'
 '16016 90000 15961' '16044 10001 00994' '15020 10000 00708'
 '16016 90000 15032' '15041 70000 16948' '16041 90000 21767'
 '16042 20000 13363' '16014 20000 02916' '16014 20000 01810'
 '12010 10000 01694' '14016 10000 19893' '15016 10000 28260'
 '15046 10000 01417' '16016 20000 01773' '15030 20000 07611'
 '15010 10000 02189' '13041 10000 34431' '15014 10000 05061'
 '15042 10000 24032' '15010 10000 03877' '16046 90000 00594'
 '16041 90000 21532' '16042 20000 13993' '15016 20002 18913'
 '13010 20000 02790' '16016 10000 13842' '16041 20000 18801'
 '14030 30000 03588' '16016 90000 16331' '16014 10000 01152'
 '14016 20000 11432' '16041 90000 17225' '16016 20000 12256'
 '15016 10001 20020' '15042 10000 13729' '15044 10000 09980'
 '15010 10000 03635' '15042 20000 01365' '16030 20000 01208'
 '16041 20000 24245' '15019 10000 02753' '16016 70000 15543'
 '14020 30000 02558' '16016 10000 09831' '15016 10000 22173'
 '16041 90000 23900' '16014 70000 02998' '14010 10000 00612'
 '14047 20000 01795' '16016 10000 09076' '14014 30000 05127'
 '16041 20000 10775' '16041 30000 23093' '16041 10000 18906'
 '16014 40000 01496' '16016 30000 07572' '12010 10000 01454'
 '16016 20000 00711' '16016 10000 10501' '15030 30000 04338'
 '14014 20000 03342' '16041 70001 18498' '16042 90000 14540'
 '13010 10005 03617' '16014 20000 01452' '16026 10000 00030'
 '16016 70000 15723' '15042 20000 01392' '16042 90000 11673'
 '14041 20000 00836' '16016 10000 14176' '15044 20000 08401'
 '15014 20000 03894' '16047 20000 00221' '14010 10000 01873'
 '16016 90000 16815' '15047 20000 01093' '14010 10000 00089'
 '14014 70000 00287' '04016 10000 21187' '15010 10000 02366'
 '14047 20000 01746' '15010 10001 03635' '15020 10000 00585'
 '13010 10000 03132' '16016 70001 06892' '15043 20000 03270'
 '14030 20000 00267' '14043 20000 04721' '16016 90000 04356'
 '16016 10000 16302' '14010 10000 03931' '15010 20000 03082'
 '16016 10000 15438' '15030 10000 06447' '15020 10001 03138'
 '13014 10006 03762' '16016 90000 16841' '16041 10000 00572'
 '13010 10000 04235' '16020 10000 00746' '16041 90000 08626'
 '16016 70000 09600' '16016 20000 09646' '15020 10000 00839'
 '16042 70001 11557' '14014 20004 03823' '10014 20000 04575'
 '13014 70000 05506' '16041 10000 01435' '16020 10000 01968'
 '15042 70000 21009' '16043 10000 03010' '15014 10002 00619'
 '15041 10000 36946' '13014 10000 01257' '15014 30000 01934'
 '15042 10000 17891' '15042 10000 25123' '15041 30000 04518'
 '14010 10000 00943' '16041 10000 23654' '16041 10000 17727'
 '15041 10000 39664' '15010 10000 02657' '14030 20000 04833'
 '16042 90000 14612' '16016 10000 12100' '13010 20000 01099'
 '14047 20000 00469' '16042 10000 14156' '16016 30000 08489'
 '16014 10000 00629' '16016 20000 16713' '16041 10000 20219'
 '16041 10000 10383' '16041 90000 25366' '13010 10000 02211'
 '12020 20001 00581' '15044 10000 12452' '13014 10000 02002'
 '13010 20000 02587' '15010 20000 01304' '16041 90000 25528'
 '16041 10000 17617' '16042 10000 13012' '16016 10000 09191'
 '16014 70000 01242' '16016 10000 15549' '16014 20000 01624'
 '15010 20000 02612' '15014 10000 05716' '16019 10001 01975'
 '16014 70000 02095' '13010 20000 04182' '16016 10000 07902'

'16016 70000 02209' '16047 30000 00651' '15014 30000 05370'
'16016 10000 07457' '16020 10000 00419' '16042 90000 14489'
'15010 10000 03058' '15016 20000 14580' '14016 10001 24095'
'15014 10000 05951' '16042 90000 14502' '13016 10000 20471'
'16042 10000 13646' '14030 30000 02656' '14010 20000 00721'
'16014 10000 02394' '16041 20000 07794' '16016 10000 12090'
'16044 10000 02850' '13042 10000 22534' '16016 10000 10179'
'15014 20000 05094' '15030 20000 03555' '15016 10000 25905'
'16041 90000 22911' '16016 30000 10301' '16041 70000 17250'
'15043 10000 04522' '14042 10000 21847' '15010 20000 04425'
'15030 20000 05881' '16042 90000 14261' '15010 10000 04061'
'16047 10001 00658' '15010 20000 03106' '15030 10000 03316'
'16016 30000 07161' '16042 70000 05582' '15047 30000 01404'
'16041 90000 22888' '16043 10000 03185' '15014 20000 01338'
'15014 20000 03832' '15014 30000 03756' '14010 20000 03732'
'15010 20000 03869' '15020 10000 03137' '14010 10000 03535'
'16044 90000 04421' '15010 20000 00974' '16016 20000 14349'
'15010 70000 02108' '13010 10000 04232' '16014 70000 02560'
'16041 90000 21588' '14014 30000 02370' '16041 10001 21289'
'15010 20000 02687' '15010 20000 03315' '14010 20000 00513'
'16042 10001 11346' '15014 30000 01210' '16041 20000 04546'
'13041 10000 34677' '16014 30000 00391' '16042 40000 14514'
'16042 20001 13087' '16016 90000 15042' '13010 10000 01500'
'15041 90000 34707' '16042 90000 14565' '13020 10000 03224'
'15041 40000 36672' '16042 90000 14362' '16042 30000 03257'
'16016 20000 13185' '15043 10000 02480' '14010 10000 02426'
'16043 90000 03334' '16010 20000 01346' '15014 20000 05919'
'14014 30000 02852' '16041 90000 18027' '16047 20000 00217'
'16041 30000 24828' '15014 20000 03781' '16047 30000 00286'
'16016 30000 09382' '14014 70000 03660' '16041 90000 23968'
'16014 30000 01701' '16016 10000 11407' '16016 20000 12692'
'16016 90000 11740' '16014 10000 00171' '16014 10000 00028'
'12010 10000 02799' '12010 20000 02815' '15016 10000 06690'
'15014 10000 01689' '15030 20000 00779' '16041 20000 25399'
'15047 20000 01929' '16041 90000 22965' '15010 10000 03070'
'15010 70000 03959' '14014 10000 04762' '16014 30000 01204'
'16016 90000 15988' '16044 10001 01163' '14010 30000 03803'
'16014 10000 00169' '16014 20000 02564' '15010 10000 03344'
'14014 20000 01551' '16041 10000 04802' '15041 90000 37065'
'16016 10000 09007' '08016 10000 09349' '14014 10000 05475'
'16014 20000 02482' '16016 20000 07247' '16042 90000 14358'
'15010 20000 01445' '15043 10000 04049' '14010 10000 02920'
'16016 20000 06305' '15010 20000 04270' '11014 20000 02435'
'12014 20000 02291' '16016 10000 05541' '13016 10000 04809'
'16014 30000 00273' '16041 40000 07174' '15041 10001 00524'
'13010 20000 04120' '15016 10000 20299' '16014 20000 02115'
'15010 20001 04253' '15016 20000 19101' '15016 10000 28617'
'16042 20000 11043' '15014 30002 00654' '16044 20000 07972'
'16042 20000 14148' '16043 20000 00209' '15043 20000 05871'
'15010 20000 03111' '15010 20000 01215' '16042 20000 08755'
'16041 90000 05114' '13014 20000 02234' '16042 10000 14413'
'16014 10000 02538' '16020 20000 01858' '16016 10000 15797'
'14043 10000 01740' '14042 20000 22587' '14016 10000 13466'
'15010 10000 00649' '16041 30000 20836' '15010 20000 03108'
'15041 10000 42820' '14010 10000 01202' '16042 20000 08586'
'16014 10001 00123' '13010 20000 04368' '14010 30000 02808'

'13010 20000 00214' '15020 10001 02152' '15047 20000 01053'
'12010 30004 01165' '16016 20000 15237' '16041 10000 21627'
'15020 20000 00377' '15041 10000 40698' '16043 90000 03327'
'14030 30000 07519' '16041 20000 23539' '15014 30000 02924'
'16041 90000 23874' '14043 20000 04611' '16014 20000 01817'
'14041 10000 16357' '15014 20000 04744' '16044 20000 07653'
'16030 20000 00402' '16041 20000 20483' '16043 10000 01195'
'16041 10000 14780' '13010 20000 04163' '10010 10000 02370'
'15010 20000 02689' '15041 20000 18948' '15041 10000 41294'
'16016 20000 15250' '15016 10000 26357' '16030 10000 04877'
'15016 10000 12193' '13041 10000 30631' '15030 10000 08505'
'16043 20000 03358' '16016 20000 15523' '15014 10001 00544'
'14010 30000 00467' '16047 20000 00866' '16046 10000 00255'
'14047 20000 01621' '16041 90000 07964' '16014 20000 02745'
'15010 10000 01752' '16030 20000 05062' '16016 40000 16282'
'14014 20000 04930' '16016 20001 10014' '15010 20000 04632'
'10010 10000 01979' '16014 70000 01876' '16042 20000 11262'
'16044 20000 03875' '15014 30000 00751' '15043 20000 04003'
'14044 10000 01894' '16016 20000 04386' '13010 10000 02721'
'14010 10000 01004' '16041 30000 11271' '16016 20000 13449'
'14030 30000 02534' '15010 10000 01371' '14010 10000 00445'
'16041 10000 13013' '13010 10000 03617' '16042 10000 13702'
'15041 10000 39880' '16041 90000 22081' '14010 20000 03114'
'14016 10000 22616' '16016 70000 15408' '13044 10000 11159'
'15043 10000 02838' '14010 20000 00994' '14010 20000 00657'
'16041 90000 25337' '16014 10000 02348' '15016 20001 25709'
'16042 20000 14493' '16014 20000 02020' '16047 20000 00648'
'15043 10000 05224' '16041 20000 10896' '14014 10000 03687'
'16014 30000 01563' '16016 20000 08272' '16016 10000 08821'
'16014 20000 00844' '16042 10003 03067' '16041 90000 04245'
'16041 10000 11949' '15030 10000 01152' '16041 30000 25183'
'15014 20000 00584' '14030 10000 04479' '16026 20000 00245'
'16016 10000 01282' '16044 20000 07650' '16042 90000 13870'
'16043 10000 00427' '15014 20001 04909' '16042 90000 12274'
'15043 20000 05826' '13010 10000 03296' '16047 20000 00760'
'16041 90000 23325' '16016 90000 11865' '16041 10000 13806'
'14016 30000 10605' '15042 70000 25536' '16043 10000 02027'
'16016 10000 11219' '16016 10000 11422' '14016 10000 11297'
'16016 10000 06026' '16041 90000 24756' '16016 90000 15447'
'16014 70000 02197' '12010 10000 00981' '13010 10000 04345'
'15016 10000 22842' '15047 20000 02024' '16014 20003 01757'
'16041 90000 24052' '16014 20000 00740' '13010 10000 03431'
'16042 90000 13740' '16016 90000 10096' '16016 20000 15909'
'16042 90000 14357' '14041 20000 08644' '15030 20000 05847'
'16010 20000 01120' '16014 10000 00572' '14010 20000 00881'
'16041 70000 22499' '14042 10000 12208' '15016 10000 08427'
'16041 20000 23833' '16016 20000 16344' '16042 20000 07009'
'16041 10000 23171' '16041 10000 25312' '15042 10000 12100'
'16014 10000 02792' '16047 20000 00477' '13042 70000 06434'
'16014 30000 02417' '15046 20000 00657' '12016 10000 24073'
'13014 20000 02241' '13010 20000 04166' '16016 10001 09598'
'16042 20000 14490' '16041 90000 20884' '16041 90000 24682'
'16016 20000 10071' '12014 20000 03082' '16016 20000 11016'
'15014 30000 05326' '16020 10000 00393' '16030 20000 00123'
'15014 20000 05221' '14010 10000 00184' '16016 10000 13266'
'15014 20000 01969' '13042 10004 09282' '16043 10000 00002'

'16041 90000 20830' '15010 20000 03701' '16042 10000 07936'
'15043 10000 03803' '16047 30000 00225' '16041 90000 25221'
'16016 10000 05664' '15016 20000 26673' '16016 90000 11498'
'16042 90000 13519' '14041 10000 31990' '16047 20000 00504'
'16014 30000 01851' '06014 10000 03154' '15048 20000 03043'
'15016 20000 28469' '15016 20000 24785' '16014 30000 03274'
'16047 20000 00649' '12010 10000 01270' '16047 20000 01096'
'16014 30000 00402' '16016 10000 06920' '15047 30000 01360'
'16047 20000 00070' '13030 20000 00594' '15010 10000 03266'
'15016 30000 18913' '16047 20000 00635' '14010 10000 00301'
'15014 10000 03177' '15047 20000 00660' '16044 30001 03192'
'15010 70000 03229' '16016 90000 10070' '16047 10000 00016'
'16041 10000 25158' '14010 10000 03714' '16044 90000 03184'
'14044 90000 06028' '16041 20000 18233' '16410 90000 01405'
'16047 20000 00826' '15010 20000 03642' '16016 20000 13578'
'14016 10000 17305' '16041 20000 24799' '14016 10000 12165'
'16041 20000 25543' '14010 30000 00584' '16016 70000 11628'
'16041 10000 24948' '16016 30000 07898' '16041 10002 07723'
'15044 10000 13165' '16041 20000 15225' '16044 90000 07922'
'13010 10005 03024' '16041 10000 19143' '16016 10000 03764'
'16041 90000 06731' '13042 10000 24926' '15043 10000 04850'
'16014 20000 01774' '14016 20001 13338' '16014 20000 02798'
'16042 10000 07541' '13010 10000 03499' '15016 10000 08908'
'16016 90000 11842' '16041 20000 25049' '16041 90000 25188'
'15010 10000 02816' '16014 20000 00511' '16041 20000 14416'
'16016 10000 04909' '16014 20000 01821' '15014 20000 03817'
'15020 10000 00251' '15010 20000 03081' '15030 10000 01851'
'14030 10000 07455' '16030 10000 02477' '15010 20000 02802'
'14010 10000 03716' '16042 90000 10013' '15041 10000 12192'
'15016 10000 21094' '14041 10000 22825' '16016 20000 07390'
'15030 20000 01100' '13016 10000 26033' '10014 20000 04336'
'16042 20000 04542' '14014 10000 05225' '14010 10000 01878'
'16020 20001 00449' '15041 10000 22038' '16016 20000 11056'
'16042 10000 12688' '15030 30000 02192' '14014 10001 02160'
'16016 20000 11702' '16047 20000 00279' '16014 20000 00194'
'16014 20000 02432' '13010 10000 03895' '16041 90000 22462'
'16041 10000 03575' '16043 10000 03082' '16016 10000 01808'
'13016 20000 06242' '15047 20001 01545' '16016 90000 16402'
'13010 10000 02346' '14042 20000 08685' '15014 30000 03699'
'14010 20000 00722' '16047 20000 00577' '15010 10000 03120'
'16042 20000 14153' '15010 30000 04374' '12010 20002 02815'
'15014 10000 05245' '16041 20000 25627' '16041 90000 01384'
'16041 10000 11394' '16016 20000 12008' '16043 10000 03324'
'16041 10000 18206' '15030 10000 08659' '15014 40000 04712'
'15041 40000 36705' '16016 10000 16332' '15010 10000 03068'
'15014 30000 01067' '16014 20000 00549' '16041 90000 25042'
'14010 10000 00103' '15042 10000 17301' '16030 20000 00046'
'16041 10000 17620' '15010 20000 04239' '08020 20000 03695'
'14014 30000 05575' '16041 10000 11037' '16016 30000 08073'
'16044 10000 07606' '16016 20000 16493' '14042 90000 11387'
'16041 10000 24259' '15014 20000 03493' '16016 10000 12216'
'15014 20000 02642' '14041 10000 18601' '10016 10000 18244'
'16016 10000 13231' '16047 30001 00184' '15014 20000 04761'
'16016 10000 03175' '16041 90000 23672' '10014 20003 01449'
'15014 10000 00012' '16041 90000 21165' '14020 30000 01175'
'14010 10000 01324' '16016 10000 14158' '16016 30000 15391'

'14020 30000 02513' '16041 10000 25420' '14010 10003 01812'
'15014 20000 05176' '16044 90000 06106' '16042 20000 14694'
'13047 10000 01450' '13030 20000 01280' '15041 20001 42177'
'16020 70000 01691' '16041 20000 25367' '16042 20000 07591'
'16016 10000 15195' '15016 20001 21875' '16014 40000 01557'
'16043 10000 03312' '15042 10000 06948' '14014 20001 02523'
'15047 20000 00510' '16016 30000 11897' '16016 10000 05905'
'16016 40000 08658' '16016 90000 12404' '14016 10000 06857'
'16047 20000 00701' '16044 90000 07042' '13010 10000 03472'
'15014 10000 02854' '14016 10013 11587' '16016 10000 06717'
'16041 10000 01334' '15041 20000 39588' '14016 10000 03732'
'15010 10000 02658' '16010 10000 01274' '16016 10000 07480'
'14014 10002 05225' '16044 70000 04738' '15044 20000 09578'
'16020 10000 01710' '16044 40000 04751' '13010 10001 04301'
'13010 10008 03617' '16041 90000 17468' '13014 20000 03977'
'16042 90000 14017' '16016 20000 16106' '16041 10000 04239'
'16042 90000 14577' '16016 10000 04822' '15044 30000 06863'
'14010 10000 02021' '16042 90000 14292' '15014 20000 04709'
'16016 20000 08277' '16044 20000 07153' '16016 10000 09050'
'15010 10000 03261' '16016 40000 11215' '16016 10000 10587'
'16030 20000 01689' '16042 10000 10101' '16030 10000 01411'
'15020 10000 02152' '16014 20000 00339' '15014 10000 04334'
'16014 20000 00729' '16016 20000 14428' '14042 30000 02472'
'06010 20000 01635' '16016 20000 15241' '15016 20000 02114'
'14044 10000 05850' '16041 20000 24193' '15047 20002 01545'
'15010 10000 03135' '15041 20002 05476' '14043 10000 02441'
'16044 10000 06217' '15016 10000 23312' '14030 10000 05657'
'14016 10000 17809' '16043 20000 03087' '16044 90000 05897'
'16041 40000 25088' '16014 20000 02100' '15016 20000 23782'
'16041 90000 24986' '16042 20000 14264' '16016 20000 13569'
'14016 10000 21080' '15016 10000 12977' '16041 90000 18440'
'16042 70000 09841' '16020 10000 00870' '16014 20000 01382'
'15016 20000 21710' '15010 10000 03878' '15042 90000 25421'
'09010 70000 01037' '14044 20000 04023' '16014 20000 00776'
'16020 20000 01477' '16014 20001 02595' '14010 10000 01949'
'15010 10000 00312' '15047 20000 01357' '16016 30001 04926'
'16042 90000 07661' '16016 70000 15216' '15010 10000 02269'
'16016 20000 07490' '16043 10000 01196' '14041 20000 12859'
'16010 20000 01347' '16041 10000 22841' '16041 20000 12444'
'15014 10000 02993' '16041 10000 05333' '15043 20000 03100'
'16041 90000 19739' '16042 20000 14741' '16016 70000 01233'
'16042 10000 13921' '15014 30000 03686' '15041 10000 31814'
'16047 40000 00503' '16041 90000 25431' '16016 20000 13326'
'16042 90000 08261' '13014 20000 05411' '13016 10000 26609'
'16042 10000 13972' '11014 10000 04281' '16016 10000 08813'
'15043 10000 00107' '16042 90000 14328' '16042 90000 14181'
'16010 20000 01324' '15041 90000 33347' '13010 20000 03116'
'14030 10000 07770' '16010 10001 00970' '14010 30000 03238'
'16041 20000 24050' '14042 10000 23415' '15010 20000 02962'
'16041 30000 13451' '16014 20000 03264' '15014 20002 05401'
'15014 20000 02179' '14030 30001 08146' '15030 20000 01567'
'15016 10000 25722' '07010 10000 00202' '16014 20000 01103'
'15043 10000 03468' '16041 70000 24914' '16041 90000 20982'
'16016 40000 11214' '14010 10000 03191' '16046 10000 00730'
'16044 90000 07533' '16041 90000 14259' '14014 10000 03708'
'16014 10000 00624' '16014 20000 01444' '15047 20001 01138'

```
 '15041 90000 31954' '15016 20000 23157' '14042 10003 00421'
 '15044 10002 08012' '14010 70000 01334' '16041 40000 24564'
 '13042 10000 24550' '11014 20000 00849' '15016 10000 04019'
 '15014 30000 01353' '15016 30001 23769' '15041 70001 35617'
 '13016 10007 16108' '14041 10000 14621' '16030 10000 04717'
 '16016 20000 11501' '15043 10000 05579' '16041 20000 23752'
 '16020 10000 00507' '14014 10000 05409' '15016 10000 14954'
 '16016 30000 08421' '13014 20000 01876' '16016 20000 07238'
 '14041 10000 29651' '15041 20001 36091' '16041 20000 10600'
 '16042 10000 13354' '16016 10000 15725' '14014 20000 04786'
 '14030 10000 02126' '16041 20000 12306' '15016 30000 00513'
 '16014 30000 01845' '15042 20000 22272' '15010 10000 02273'
 '16014 20000 02595' '16044 90000 06207' '16042 70000 13341'
 '15043 10000 02451' '16044 90000 04942' '15043 10000 04415'
 '16047 20000 01212' '16010 10000 01260' '16042 10000 14708'
 '16016 20000 08899' '13041 10000 29904' '16041 90000 16695'
 '16014 10000 03058' '16016 90000 16439' '14020 20000 02989'
 '16041 10000 24796' '16044 30000 03718' '15043 10000 02550'
 '16042 90000 14589' '16044 90000 06527' '16042 20000 14833'
 '15042 10001 15427' '16047 20000 00847' '15042 10001 13271'
 '16014 20000 00091' '16042 10000 13656' '16041 30000 15736'
 '16016 20000 16867' '16014 10000 02398']


Unique values in 'permit_status':
['Issued' 'Permit Finaled' 'CofO Issued' 'CofO in Progress'
 'Permit Expired' 'Permit Closed' 'CofO Corrected']


Unique values in 'inspection_date':
['2016-07-20T00:00:00.000' '2016-07-22T00:00:00.000'
 '2016-07-18T00:00:00.000' '2016-07-21T00:00:00.000'
 '2016-07-19T00:00:00.000' '2016-07-23T00:00:00.000']


Unique values in 'inspection':
['Rough-Ventilation' 'Smoke Detectors' 'Insulation' 'Final' 'Inspection'
 'Drywall Nailing' 'Plumbing Verification' 'Rough' 'Masonry Wall/Backfill'
 'ELECTRICAL-Rough' 'BUILDING-Rough-Frame' 'Wood Frame'
 'Water Piping or Service' 'Service/Power Release'
 'TCO Plumbing Verification' 'T-Bar Ceiling' 'Bottom/Toe' 'CofO Issuance'
 'Footing/Foundation/Slab' 'Fire Damper Framing'
 'Floor/Roof Diaphrgm/Shear Wall' 'Deputy Shotcrete' 'Pre-Inspection'
 'Gas Test' 'Life Safety' 'HVAC-Rough' 'SGSOV-Seismic Gas S/O Valve'
 'PLUMBING-Rough' 'Special/Order Compliance' 'Rough-A/C and Heating'
 'Green Building Rough' 'Pool Equipment Noise Test'
 'Verify Sprinkler Sign Off' 'Enclosure/Fence' 'Deputy Grading'
 'Deputy Steel/Welding' 'ELECTRICAL-Final'
 'Excavation/Setback/Form/Re-Bar' 'Piling/Pier/Caisson' 'Shower Pan'
 'Interior/Exterior Lathing' 'Bathtub Test' 'HVAC-Final'
 'Reinforced Masonry Frame' 'Construction Power(CTS)'
 'Deputy Reinf. Concrete' 'Rough-Elec/Plmb/HVAC' 'Sewer or Sewer Cap'
 'Grounding or Bonding' 'Underground' 'Irrigation/Landscape'
 'Partition/T-Bar Ceiling' 'Weld' 'Grading Verification' 'Overhead Hydro'
 'METHANE-Rough' 'Deputy Wood Construction' 'Deputy Reinf. Masonry'
 'TCO Sprinkler Verification' 'Underground Mechanical'
```

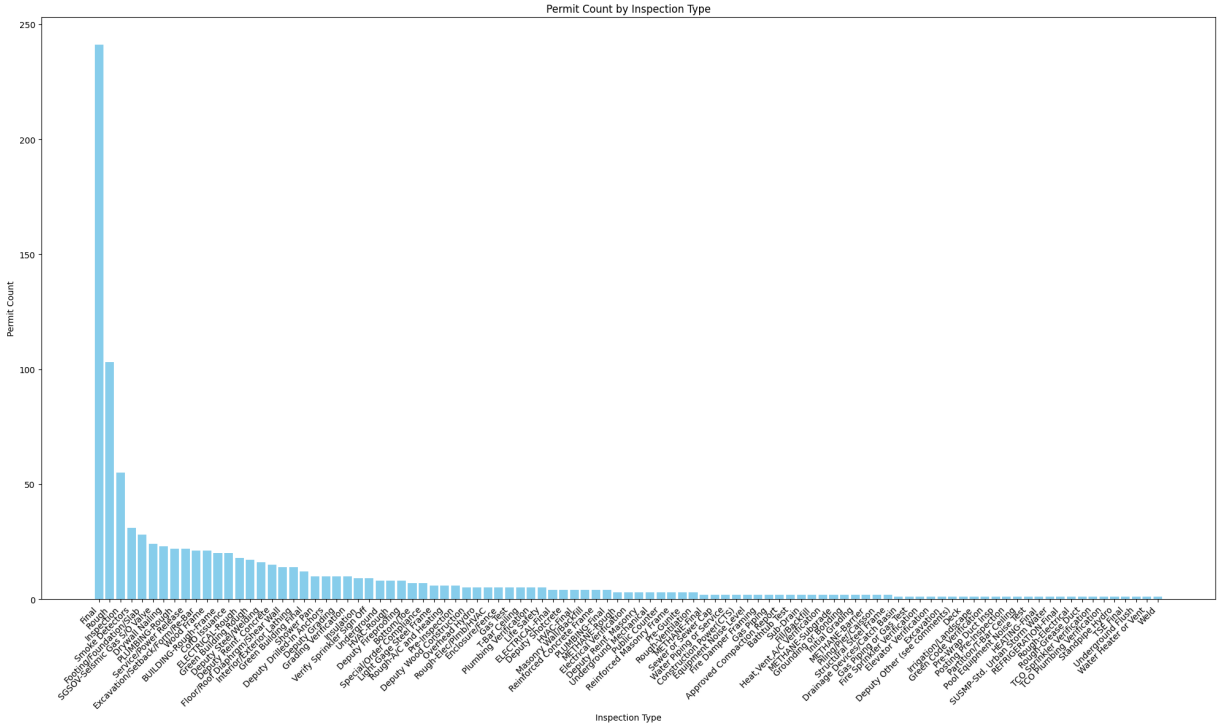
```
 'Green Building Final' 'Equipment Noise Level' 'Elevator Verification'
 'METHANE-Barrier' 'Light Gage Steel Frame' 'Reinforced Concrete Frame'
 'PLUMBING-Final' 'Underground Flush' 'Deputy Fireproofing'
 'Green Code Verification' 'Deputy Drilled-In Anchors' 'Gas Piping'
 'Public Counter' 'Initial Grading' 'Pre-Gunite' 'Electrical Verification'
 'METHANE-Subgrade' 'Heat,Vent,A/C Verification'
 'Approved Compaction Report' 'Posting Pre-Inspection'
 'Structural Steel Frame' 'Standpipe Hydro' 'Rough-Electrical' 'TSE Final'
 'METHANE-Final' 'REFRIGERATION-Final' 'SUSMP-Std. Urban Storm Water'
 'Sub-Drain' 'Water Heater or Vent' 'Deputy Other (see comments)'
 'Fill/Backfill' 'HEATING-Final' 'Deck' 'Rough-Grease Duct'
 'Pre-Wrap Duct Insp' 'Drainage Devices/Catch Basin' 'Excavation'
 'Gas Piping or Gas Test' 'Fire Sprinkler Verification']


Unique values in 'inspection_result':
['Partial Approval' 'Insp Cancelled' 'Approved' 'Permit Finaled'
 'Insp Scheduled' 'Not Ready for Inspection' 'OK for CofO'
 'Completed (special insp)' 'Partial Inspection' 'Corrections Issued'
 'CofO in Progress' 'Cancelled' 'Conditional Approval' 'CofO Issued'
 'SGSOV Approved' 'SGSOV Not Ready' 'CofO on Hold' 'Not Applicable'
 'No Access for Inspection' 'Completed' 'Permit Closed'
 'SGSOV Not Required' 'OK to Expire Permit' 'Permit Expired'
 'CofO Corrected' 'OK to Issue CofO' 'Off-Hour Fees Due']
```

In [19]:

```python
# Group by 'inspection' and count the occurrences of 'permit'
inspection_counts = df1.groupby('inspection')['permit'].count().reset_index(name='p

# Sort the data by 'permit_count' in descending order
inspection_counts_sorted = inspection_counts.sort_values(by='permit_count', ascendi

# Display the table
print(inspection_counts_sorted)

# Create the bar plot
plt.figure(figsize=(20, 12))
plt.bar(inspection_counts_sorted['inspection'], inspection_counts_sorted['permit_co
plt.xlabel('Inspection Type')
plt.ylabel('Permit Count')
plt.title('Permit Count by Inspection Type')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

	inspection	permit_count
27	Final	241
71	Rough	103
45	Inspection	55
82	Smoke Detectors	31
31	Footing/Foundation/Slab	28
..	...	...
84	Standpipe Hydro	1
90	TSE Final	1
92	Underground Flush	1
95	Water Heater or Vent	1
97	Weld	1

[99 rows x 2 columns]



Key Observations: Final Inspections: This type of inspection has the highest permit cou (241).s. It's a crucial inspection type in the construction process, likely marking the final approval stages of the work. Rough Inspections: Coming in secits, rough inspect (103)ions are typically done during the early stages of construction (after framing and before drywall), showing a significant amount of ongoing construction activit InspecThese occupy the third position (55) as rmits, this category likely includes regular inspections throughout various stages of construction to ensure complia

> Smoke DetS5.34%, smoke detectors are essential for safety inspections, often required by building codes, though they represent a smaller portion of the overall inspecpections.

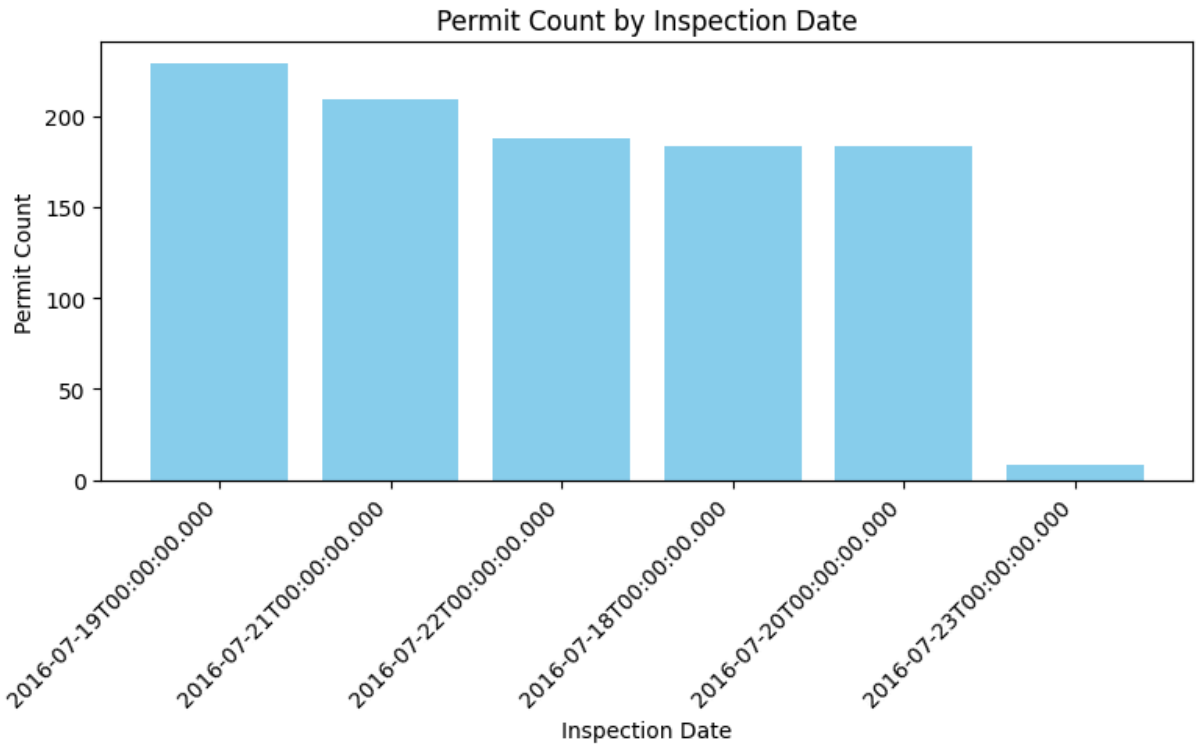
Interesting Insight:

Final inspections are by far the most frequent type of inspection in the dataset, suggesting that most of the inspections recorded are related to ensuring that a building or structure meets all requirements before it is considered complete. It could indicate that the dataset focuses more on the concluding stages of construction work rather than on ongoing or preliminary assessments.

The distribution shows a high concentration of activity around certain inspection types like Final and Rough, with a sharp drop-off for others such as Smoke Detectors and Footing/Foundation/Slab. This could indicate an area where most building projects are at a finalizing stage rather than ongoing or early-phase inspections.

In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Group by 'inspection_date' and count the occurrences of 'permit_status'
inspection_counts = df1.groupby('inspection_date')['permit_status'].count().reset_i

# Add a new column for day names
inspection_counts['day_name'] = pd.to_datetime(inspection_counts['inspection_date']

# Sort the data by 'permit_count' in descending order
inspection_counts_sorted = inspection_counts.sort_values(by='permit_count', ascendi

# Display the table with day names
print(inspection_counts_sorted)

# Create the bar plot
plt.figure(figsize=(8, 5))
plt.bar(inspection_counts_sorted['inspection_date'], inspection_counts_sorted['perm
plt.xlabel('Inspection Date')
plt.ylabel('Permit Count')
plt.title('Permit Count by Inspection Date')
plt.xticks(rotation=45, ha='right')

# Add day names to the x-axis labels
for i, label in enumerate(plt.gca().get_xticklabels()):
    label.set_text(f"{label.get_text()} ({inspection_counts_sorted['day_name'].iloc

plt.tight_layout()
plt.show()
```

	inspection_date	permit_count	day_name
1	2016-07-19T00:00:00.000	229	Tuesday
3	2016-07-21T00:00:00.000	209	Thursday
4	2016-07-22T00:00:00.000	188	Friday
0	2016-07-18T00:00:00.000	183	Monday
2	2016-07-20T00:00:00.000	183	Wednesday
5	2016-07-23T00:00:00.000	8	Saturday



Peak Inspection Day The inspection ocuured only for 6 days (2016-07-19 to 2016-07-23) Peak Inspection Days: The highest permit count occurs on 2016-07-19 (Tuesday) with 229 permits, followed closely by 2016-07-21 (Thursday) with 209 permits. These two mid-week days appear to have the most inspection activity T This could suggest that inspections tend to peak on business days, specifically Tuesdays and Thursdays, possibly due to the increased construction work during the middle of the wee

```
Consistent Permit C
    2016-07-18 (Monday) and 2016-07-20 (Wednesday) both have 183
permits. While the numbers are slightly lower than the peak, these
two days show steady inspection activity, indicative of a regular
schedule of inspections. Monday and Wednesday are likely common
inspection days, as the start and mid-week often have planned
inspectiork.

Significant Decline on thd:
    2016-07-23 (Saturday) stands out as the day with the lowest
permit count at only 8 permits. This significant drop could be due
to fewer inspections being conducted on the weekend, as most
inspections are likely to be scheduled on weekdays. This suggests
that inspections are generally slower or less frequent on weekends,
possibly due to reduced staffing or prioritization of weekday
schedules for intions.

DatCoe Comparison:
    2016-07-21 and 2016-07-22 (Thursday and Friday) show a moderate
decrease in inspections compared to earlier days (Tuesday, 2016-07-
19), but still maintain a fairly high number, especially Friday
(188). However, by 2016-07-23 (Saturday), the number drops
```

drastically, suggesting a weekend slowdown in activity.

Insights:

Weekday Activity: The majority of inspections are concentrated in the early part of the week, specifically Tuesday and Thursday, with a noticeable drop towards the weekend. This could be attributed to construction schedules, staffing availability, or operational priorities that skew inspections toward business days.

Weekend Effect: The dramatic drop in permit counts on Saturday (2016-07-23) indicates that inspections are much less frequent or possibly halted on weekends, aligning with standard business operations where construction inspections are more common during working hours.

Mid-week Peak: The mid-week surge (Tuesday and Thursday) could reflect a common pattern in inspection schedules, where most projects reach a stage that requires inspection around the middle of the week, thus leading to higher counts on those specific days.

Conclusion:

This dataset shows that permit inspections are heavily concentrated on weekdays, with a clear peak in the middle of the week (Tuesday and Thursday), while Saturday shows a stark decline in inspections. The weekend effect and the consistent mid-week activity suggest that inspection schedules are likely optimized around working days, and a reduced schedule is implemented over the weekend.emented over the weekend.

Question2: Make a table and a visualization showing the number of inspections by geography. In a sentence or two describe any patterns you observe.

Make a table and a visualization showing the results of inspections across geographies.

In a sentence or two describe any patterns you observe.

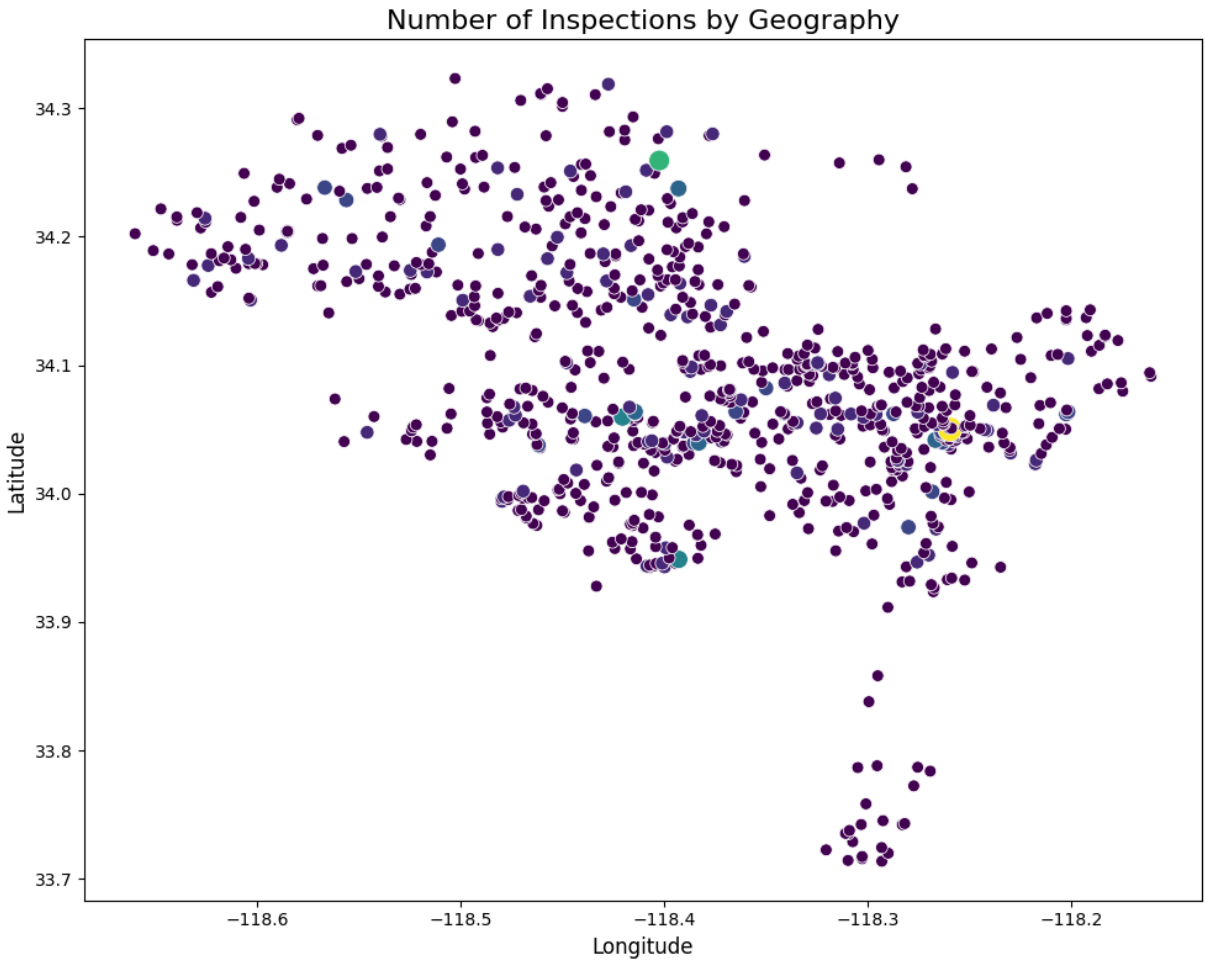
Were there any permits that did not get an inspection?

In [24]:
```
# Convert lat_lon column to actual dictionary and extract latitude and longitude
df['lat_lon'] = df['lat_lon'].apply(lambda x: ast.literal_eval(x) if isinstance(x,
df['latitude'] = df['lat_lon'].apply(lambda x: float(x.get('latitude')) if isinstan
df['longitude'] = df['lat_lon'].apply(lambda x: float(x.get('longitude')) if isinst

# Remove rows with missing latitudes or longitudes
```

```
df_cleaned = df.dropna(subset=['latitude', 'longitude'])

# Create a GeoDataFrame using latitude and longitude
geometry = [Point(lon, lat) for lon, lat in zip(df_cleaned['longitude'], df_cleaned
gdf = gpd.GeoDataFrame(df_cleaned, geometry=geometry)

# 1. Number of Inspections by Geography:
# Group by geographical locations (using Point geometry) and count inspections
inspection_by_geography = gdf.groupby(['latitude', 'longitude']).size().reset_index

# Table: Number of Inspections by Geography
print("Number of Inspections by Geography:")
print(inspection_by_geography)

# Visualization for Inspections by Geography
plt.figure(figsize=(10, 8))
sns.scatterplot(data=inspection_by_geography, x='longitude', y='latitude', size='in
plt.title('Number of Inspections by Geography', fontsize=16)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.tight_layout()
plt.show()

# 2. Inspection Results Across Geographies (Using computed region columns)
# List of computed region columns to analyze
region_columns = [
    ':@computed_region_qz3q_ghft', ':@computed_region_kqwf_mjcx', ':@computed_regio
    ':@computed_region_tatf_ua23', ':@computed_region_ur2y_g4cx', ':@computed_regio
]

# Clean the dataset by removing rows with missing region or inspection result data
df_cleaned_regions = df.dropna(subset=region_columns + ['inspection_result'])

# Group by computed region columns and inspection result
inspection_by_region = df_cleaned_regions.groupby(region_columns + ['inspection_res

# Table: Inspection Results Across Geographies
print("Inspection Results Across Geographies:")
print(inspection_by_region)

# Visualization: Heatmap showing inspection results across different geographies
inspection_pivot = inspection_by_region.pivot_table(index=region_columns, columns='

# Visualize using a heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(inspection_pivot, annot=True, cmap='Blues', fmt='d', cbar_kws={'label':
plt.title('Inspection Results Across Geographies', fontsize=16)
plt.xlabel('Inspection Result', fontsize=12)
plt.ylabel('Geographical Region', fontsize=12)
plt.tight_layout()
plt.show()

# 3. Permits Without Inspections
# Identify permits with inspections by checking if 'inspection_result' is not null
inspected_permits = df.dropna(subset=['inspection_result'])['permit'].unique()
```

```python
# Get all unique permits
all_permits = df['permit'].unique()

# Find permits that did not have inspections
permits_without_inspection = [permit for permit in all_permits if permit not in ins

# Create a table to show permits without inspections
permits_without_inspection_df = df[df['permit'].isin(permits_without_inspection)].d

# Table: Permits Without Inspections
print("Permits Without Inspections:")
print(permits_without_inspection_df[['permit', 'permit_status', 'inspection_result'
```

```
Number of Inspections by Geography:
     latitude  longitude  inspection_count
0    33.71374 -118.29306                 1
1    33.71437 -118.30965                 1
2    33.71575 -118.30272                 1
3    33.71741 -118.30272                 1
4    33.72002 -118.28990                 1
..        ...        ...               ...
825  34.31068 -118.46037                 1
826  34.31128 -118.46070                 1
827  34.31509 -118.45733                 1
828  34.31859 -118.42740                 2
829  34.32306 -118.50270                 1

[830 rows x 3 columns]
```


Number of Inspections by Geography
Latitude
Longitude
34.3
34.2
34.1
34.0
33.9
33.8
33.7
−118.6
−118.5
−118.4
−118.3
−118.2

Inspection Results Across Geographies:

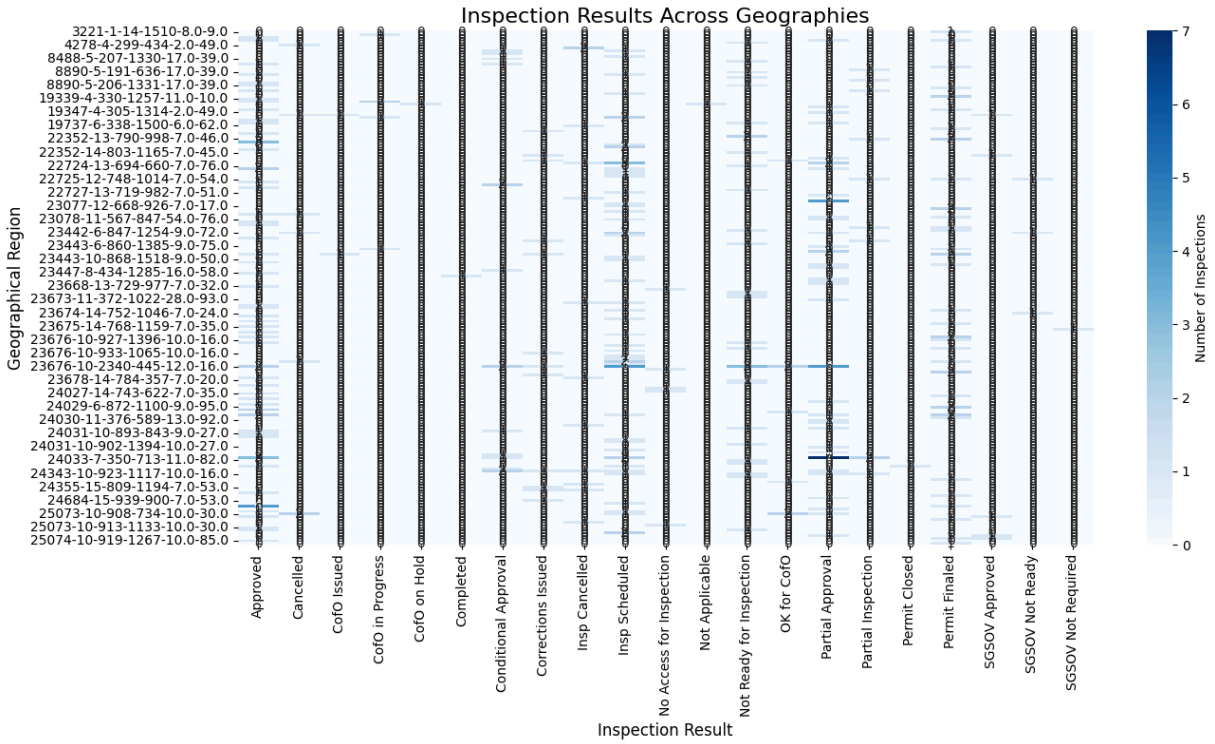
	:@computed_region_qz3q_ghft	:@computed_region_kqwf_mjcx	\
0	3221	1	
1	3222	1	
2	3342	15	
3	3342	15	
4	3342	15	
..	...	...	
268	25074	10	
269	25074	10	
270	25074	10	
271	25074	10	
272	25075	10	

	:@computed_region_k96s_3jcv	:@computed_region_tatf_ua23	\
0	14	1510	
1	6	541	
2	985	831	
3	985	940	
4	985	940	
..	...	...	
268	915	952	
269	919	240	
270	919	240	
271	919	1267	
272	930	1440	

	:@computed_region_ur2y_g4cx	:@computed_region_2dna_qi2s	\
0	8.0	9.0	
1	8.0	7.0	
2	15.0	33.0	
3	15.0	33.0	
4	15.0	33.0	
..	...	...	
268	10.0	30.0	
269	10.0	85.0	
270	10.0	85.0	
271	10.0	85.0	
272	10.0	16.0	

	inspection_result	inspection_count
0	Permit Finaled	1
1	CofO in Progress	1
2	Approved	1
3	Approved	1
4	Partial Approval	1
..	...	...
268	SGSOV Approved	1
269	Permit Finaled	1
270	SGSOV Approved	1
271	Approved	1
272	Permit Finaled	1

[273 rows x 8 columns]




```
Permits Without Inspections:
Empty DataFrame
Columns: [permit, permit_status, inspection_result]
Index: []
```

In [36]:
```
# Find permits with no inspection
permits_without_inspections = df.isna().sum()
permits_without_inspections
```

Out[36]:
```
address                        0
permit                         0
permit_status                  0
inspection_date                0
inspection                     0
inspection_result              0
lat_lon                        0
:@computed_region_qz3q_ghft    0
:@computed_region_kqwf_mjcx    0
:@computed_region_k96s_3jcv    0
:@computed_region_tatf_ua23    0
:@computed_region_ur2y_g4cx    649
:@computed_region_2dna_qi2s    46
latitude                       0
longitude                      0
dtype: int64
```

Result for Question 2: Every geographical location characterized by lat and log has inspection.

The regions such as io:@computed_region_ur2y_g4cx and

:@computed_region_2dna_qi2shave missing values, which means they didnt have inspections.

Question3: Your manager is convinced 'out of town' contractors are not as invested in the success

of their projects and so are the main culprits when it comes to violations. You are asked to complete an analysis to test this hypothesis. Produce a model that quantifies the relationship between a contractor's place of origin and their inspection outcome history. Investigate any other relevant factors as necessary. Interpret your results and produce a clear response for your manager.

In [27]:
```
# Import required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import xgboost as xgb
from geopy.distance import geodesic
import json
import matplotlib.pyplot as plt
import seaborn as sns

# Convert inspection_date to datetime
df['inspection_date'] = pd.to_datetime(df['inspection_date'])

# Extract location information from lat_lon
df['latitude'] = df['lat_lon'].apply(lambda x: float(json.loads(x)['latitude']) if
df['longitude'] = df['lat_lon'].apply(lambda x: float(json.loads(x)['longitude']) i

# Determine if contractor is local (within 50 miles of LA center)
la_center = (34.0522, -118.2437)
df['distance_from_la'] = df.apply(
    lambda row: geodesic(la_center, (row['latitude'], row['longitude'])).miles
    if pd.notnull(row['latitude']) and pd.notnull(row['longitude'])
    else None, axis=1
)
df['is_local'] = df['distance_from_la'].apply(lambda x: 1 if x <= 50 else 0 if pd.n

# Create time-based features
df['month'] = df['inspection_date'].dt.month
df['day_of_week'] = df['inspection_date'].dt.dayofweek

# Encode categorical variables
le = LabelEncoder()
df['permit_status_encoded'] = le.fit_transform(df['permit_status'])
```

```
df['inspection_encoded'] = le.fit_transform(df['inspection'])

# Create binary target variable for inspection outcome
success_outcomes = ['Approved', 'Permit Finaled']
df['inspection_success'] = df['inspection_result'].isin(success_outcomes).astype(in

# Fill NA values in computed regions
region_cols = [col for col in df.columns if ':@computed_region' in col]
df[region_cols] = df[region_cols].fillna(-1)

# Prepare features for the model
feature_cols = [
    'is_local',
    'distance_from_la',
    'month',
    'day_of_week',
    'permit_status_encoded',
    'inspection_encoded'
] + region_cols

X = df[feature_cols].fillna(-1)
y = df['inspection_success']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Initialize and train XGBoost classifier
model = xgb.XGBClassifier(
    max_depth=3,
    learning_rate=0.1,
    n_estimators=100,
    objective='binary:logistic',
    random_state=42
)

# Train the model
model.fit(
    X_train,
    y_train,
    eval_set=[(X_test, y_test)],
    eval_metric='auc',
    early_stopping_rounds=10,
    verbose=False
)

# Make predictions
y_pred = model.predict(X_test)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Get and plot feature importance
importance = model.feature_importances_
feature_importance = pd.DataFrame({
    'feature': feature_cols,
```

```
    'importance': importance
}).sort_values('importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importance)
plt.title('Feature Importance in Predicting Inspection Success')
plt.show()

# Analyze local vs out-of-town contractors
local_success_rate = y_test[X_test['is_local'] == 1].mean()
outoftown_success_rate = y_test[X_test['is_local'] == 0].mean()

print("\nSuccess Rates:")
print(f"Local Contractors: {local_success_rate:.2%}")
print(f"Out-of-town Contractors: {outoftown_success_rate:.2%}")

# Statistical Analysis
from scipy import stats

# Create contingency table
contingency_table = pd.crosstab(X_test['is_local'], y_test)

# Perform chi-square test
chi2, p_value = stats.chi2_contingency(contingency_table)[:2]

print("\nStatistical Analysis:")
print(f"Chi-square statistic: {chi2:.2f}")
print(f"p-value: {p_value:.4f}")

# Calculate effect size (Cramer's V)
n = contingency_table.sum().sum()
min_dim = min(contingency_table.shape) - 1
cramer_v = np.sqrt(chi2 / (n * min_dim))
print(f"Effect size (Cramer's V): {cramer_v:.4f}")

# Visualize success rates by location
plt.figure(figsize=(8, 6))
success_rates = pd.DataFrame({
    'Contractor Type': ['Local', 'Out-of-town'],
    'Success Rate': [local_success_rate, outoftown_success_rate]
})
sns.barplot(x='Contractor Type', y='Success Rate', data=success_rates)
plt.title('Inspection Success Rates by Contractor Location')
plt.ylim(0, 1)
plt.show()

# Additional analysis: Inspection outcomes by distance
plt.figure(figsize=(10, 6))
sns.boxplot(x='inspection_success', y='distance_from_la', data=df)
plt.title('Distance from LA Center by Inspection Outcome')
plt.xlabel('Inspection Success')
plt.ylabel('Distance (miles)')
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[27], line 4
      2 import pandas as pd
      3 import numpy as np
----> 4 from sklearn.model_selection import train_test_split
      5 from sklearn.preprocessing import LabelEncoder
      6 from sklearn.metrics import classification_report, confusion_matrix

File ~\anaconda3\lib\site-packages\sklearn\__init__.py:73
     62 # `_distributor_init` allows distributors to run custom init code.
     63 # For instance, for the Windows wheel, this is used to pre-load the
     64 # vcomp shared library runtime for OpenMP embedded in the sklearn/.libs
   (...)
     67 # later is linked to the OpenMP runtime to make it possible to introspect
     68 # it and importing it first would fail if the OpenMP dll cannot be found.
     69 from . import (  # noqa: F401 E402
     70     __check_build,
     71     _distributor_init,
     72 )
---> 73 from .base import clone  # noqa: E402
     74 from .utils._show_versions import show_versions  # noqa: E402
     76 _submodules = [
     77     "calibration",
     78     "cluster",
   (...)
    114     "compose",
    115 ]

File ~\anaconda3\lib\site-packages\sklearn\base.py:19
     17 from ._config import config_context, get_config
     18 from .exceptions import InconsistentVersionWarning
---> 19 from .utils._estimator_html_repr import _HTMLDocumentationLinkMixin, estimat
or_html_repr
     20 from .utils._metadata_requests import _MetadataRequester, _routing_enabled
     21 from .utils._param_validation import validate_parameter_constraints

File ~\anaconda3\lib\site-packages\sklearn\utils\__init__.py:15
     13 from . import _joblib, metadata_routing
     14 from ._bunch import Bunch
---> 15 from ._chunking import gen_batches, gen_even_slices
     16 from ._estimator_html_repr import estimator_html_repr
     18 # Make _safe_indexing importable from here for backward compat as this parti
cular
     19 # helper is considered semi-private and typically very useful for third-part
y
     20 # libraries that want to comply with scikit-learn's estimator API. In partic
ular,
     21 # _safe_indexing was included in our public API documentation despite the le
ading
     22 # `_` in its name.

File ~\anaconda3\lib\site-packages\sklearn\utils\_chunking.py:11
      8 import numpy as np
     10 from ..._config import get_config
---> 11 from ._param_validation import Interval, validate_params
```

```
     14 def chunk_generator(gen, chunksize):
     15     """Chunk generator, ``gen`` into lists of length ``chunksize``. The last
     16     chunk may have a length less than ``chunksize``."""

File ~\anaconda3\lib\site-packages\sklearn\utils\_param_validation.py:17
     14 from scipy.sparse import csr_matrix, issparse
     16 from .._config import config_context, get_config
---> 17 from .validation import _is_arraylike_not_scalar
     20 class InvalidParameterError(ValueError, TypeError):
     21     """Custom exception to be raised when the parameter of a class/method/fu
nction
     22     does not have a valid type or value.
     23     """

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:21
     19 from .. import get_config as _get_config
     20 from ..exceptions import DataConversionWarning, NotFittedError, PositiveSpec
trumWarning
---> 21 from ..utils._array_api import _asarray_with_order, _is_numpy_namespace, get
_namespace
     22 from ..utils.deprecation import _deprecate_force_all_finite
     23 from ..utils.fixes import ComplexWarning, _preserve_dia_indices_dtype

File ~\anaconda3\lib\site-packages\sklearn\utils\_array_api.py:17
     14 import scipy.special as special
     16 from .._config import get_config
---> 17 from .fixes import parse_version
     19 _NUMPY_NAMESPACE_NAMES = {"numpy", "array_api_compat.numpy"}
     22 def yield_namespaces(include_numpy_namespaces=True):

File ~\anaconda3\lib\site-packages\sklearn\utils\fixes.py:25
     22     pd = None
     24 from ..externals._packaging.version import parse as parse_version
---> 25 from .parallel import _get_threadpool_controller
     27 _IS_32BIT = 8 * struct.calcsize("P") == 32
     28 _IS_WASM = platform.machine() in ["wasm32", "wasm64"]

File ~\anaconda3\lib\site-packages\sklearn\utils\parallel.py:13
     10 from functools import update_wrapper
     12 import joblib
---> 13 from threadpoolctl import ThreadpoolController
     15 from .._config import config_context, get_config
     17 # Global threadpool controller instance that can be used to locally limit th
e number of
     18 # threads without looping through all shared libraries every time.
     19 # It should not be accessed directly and _get_threadpool_controller should b
e used
     20 # instead.

ModuleNotFoundError: No module named 'threadpoolctl'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: