In [2]:
```
import os
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC


def scrape_health_department(url):
    # Modern Selenium automatically detects Chrome and manages the driver
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)

    driver.get(url)
    os.makedirs('Health/files', exist_ok=True)

    global_index = 0

    try:
        # Wait until the main container table/rows load
        wait.until(EC.presence_of_element_located((By.CSS_SELECTOR, ".identifier-row")))

        while True:
            # Re-fetch row COUNT on every iteration to avoid stale DOM references
            rows_count = len(driver.find_elements(By.CSS_SELECTOR, ".identifier-row"))
            print(f"Found {rows_count} items on this page.")

            for i in range(rows_count):
                success = scrape_popup_data(driver, wait, i, global_index)
                if success:
                    global_index += 1

            # --- PAGINATION ---
            try:
                # Target the Next button
                next_btn_xpath = '//label[@data-bind="click: identifierSearch.plusPage, disable: (identifi
                next_button = driver.find_element(By.XPATH, next_btn_xpath)

                # Check Knockout 'disabled' state
                class_attr = next_button.get_attribute("class") or ""
```

```
                if "disabled" in class_attr or next_button.get_attribute("disabled") is not None:
                    print("Reached the last page. Stopping.")
                    break

                # Store first row reference to wait for staleness (page transition)
                first_row = driver.find_element(By.CSS_SELECTOR, ".identifier-row")

                driver.execute_script("arguments[0].scrollIntoView(true);", next_button)
                driver.execute_script("arguments[0].click();", next_button)

                # Wait until old page rows disappear to ensure new page has loaded
                wait.until(EC.staleness_of(first_row))
                wait.until(EC.presence_of_element_located((By.CSS_SELECTOR, ".identifier-row")))
                time.sleep(1)  # Brief pause for Knockout binding to finish rendering

            except Exception as e:
                print(f"Pagination finished or failed: {e}")
                break

    finally:
        driver.quit()


def scrape_popup_data(driver, wait, row_index, global_index):
    try:
        # Re-fetch row by index right before clicking
        rows = driver.find_elements(By.CSS_SELECTOR, ".identifier-row")
        row = rows[row_index]

        driver.execute_script("arguments[0].scrollIntoView(true);", row)
        driver.execute_script("arguments[0].click();", row)

        # 1. Wait for modal to become visible
        modal_selector = ".modal-dialog, .modal-content, #inspectionModal"
        modal = wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, modal_selector)))

        # 2. Extract ONLY the modal HTML container (isolates the popup data)
        modal_html = modal.get_attribute('outerHTML')

        filename = f"Health/files/popup_{global_index}.html"
        with open(filename, 'w', encoding='utf-8') as f:
            f.write(modal_html)
```

```
        print(f"Saved pop-up {global_index} -> {filename}")

        # 3. Close modal cleanly
        close_btn = wait.until(EC.element_to_be_clickable((By.CSS_SELECTOR, "button[data-dismiss='modal'],
        driver.execute_script("arguments[0].click();", close_btn)

        # 4. Wait for modal to completely disappear before proceeding
        wait.until(EC.invisibility_of_element_located((By.CSS_SELECTOR, modal_selector)))
        return True

    except Exception as e:
        print(f"Error on pop-up index {global_index} (row {row_index}): {e}")
        # If modal gets stuck open, attempt JS cleanup so the loop can recover
        try:
            driver.execute_script("$('.modal').modal('hide');")
        except Exception:
            pass
        return False


if __name__ == "__main__":
    url = 'https://webapp.sanswrite.com/mo_doh/pulaski/'
    scrape_health_department(url)
```

```
Found 1 items on this page.
Error on pop-up index 0 (row 0): Message:

Found 25 items on this page.
Saved pop-up 0 -> Health/files/popup_0.html
Saved pop-up 1 -> Health/files/popup_1.html
Saved pop-up 2 -> Health/files/popup_2.html
Saved pop-up 3 -> Health/files/popup_3.html
Saved pop-up 4 -> Health/files/popup_4.html
Saved pop-up 5 -> Health/files/popup_5.html
Saved pop-up 6 -> Health/files/popup_6.html
Saved pop-up 7 -> Health/files/popup_7.html
Saved pop-up 8 -> Health/files/popup_8.html
Saved pop-up 9 -> Health/files/popup_9.html
Saved pop-up 10 -> Health/files/popup_10.html
Saved pop-up 11 -> Health/files/popup_11.html
Saved pop-up 12 -> Health/files/popup_12.html
Saved pop-up 13 -> Health/files/popup_13.html
Saved pop-up 14 -> Health/files/popup_14.html
Saved pop-up 15 -> Health/files/popup_15.html
Saved pop-up 16 -> Health/files/popup_16.html
Saved pop-up 17 -> Health/files/popup_17.html
Saved pop-up 18 -> Health/files/popup_18.html
Saved pop-up 19 -> Health/files/popup_19.html
Saved pop-up 20 -> Health/files/popup_20.html
Saved pop-up 21 -> Health/files/popup_21.html
Saved pop-up 22 -> Health/files/popup_22.html
Saved pop-up 23 -> Health/files/popup_23.html
Saved pop-up 24 -> Health/files/popup_24.html
Found 25 items on this page.
Saved pop-up 25 -> Health/files/popup_25.html
Saved pop-up 26 -> Health/files/popup_26.html
Saved pop-up 27 -> Health/files/popup_27.html
Saved pop-up 28 -> Health/files/popup_28.html
Saved pop-up 29 -> Health/files/popup_29.html
Saved pop-up 30 -> Health/files/popup_30.html
Saved pop-up 31 -> Health/files/popup_31.html
Saved pop-up 32 -> Health/files/popup_32.html
Saved pop-up 33 -> Health/files/popup_33.html
Saved pop-up 34 -> Health/files/popup_34.html
Saved pop-up 35 -> Health/files/popup_35.html
Saved pop-up 36 -> Health/files/popup_36.html
```

```
Saved pop-up 37 -> Health/files/popup_37.html
Saved pop-up 38 -> Health/files/popup_38.html
Saved pop-up 39 -> Health/files/popup_39.html
Saved pop-up 40 -> Health/files/popup_40.html
Saved pop-up 41 -> Health/files/popup_41.html
Saved pop-up 42 -> Health/files/popup_42.html
Saved pop-up 43 -> Health/files/popup_43.html
Saved pop-up 44 -> Health/files/popup_44.html
Saved pop-up 45 -> Health/files/popup_45.html
Saved pop-up 46 -> Health/files/popup_46.html
Saved pop-up 47 -> Health/files/popup_47.html
Saved pop-up 48 -> Health/files/popup_48.html
Saved pop-up 49 -> Health/files/popup_49.html
Found 25 items on this page.
Saved pop-up 50 -> Health/files/popup_50.html
Saved pop-up 51 -> Health/files/popup_51.html
Saved pop-up 52 -> Health/files/popup_52.html
Saved pop-up 53 -> Health/files/popup_53.html
Saved pop-up 54 -> Health/files/popup_54.html
Saved pop-up 55 -> Health/files/popup_55.html
Saved pop-up 56 -> Health/files/popup_56.html
Saved pop-up 57 -> Health/files/popup_57.html
Saved pop-up 58 -> Health/files/popup_58.html
Saved pop-up 59 -> Health/files/popup_59.html
Saved pop-up 60 -> Health/files/popup_60.html
Saved pop-up 61 -> Health/files/popup_61.html
Saved pop-up 62 -> Health/files/popup_62.html
Saved pop-up 63 -> Health/files/popup_63.html
Saved pop-up 64 -> Health/files/popup_64.html
Saved pop-up 65 -> Health/files/popup_65.html
Saved pop-up 66 -> Health/files/popup_66.html
Saved pop-up 67 -> Health/files/popup_67.html
Saved pop-up 68 -> Health/files/popup_68.html
Saved pop-up 69 -> Health/files/popup_69.html
Saved pop-up 70 -> Health/files/popup_70.html
Saved pop-up 71 -> Health/files/popup_71.html
Saved pop-up 72 -> Health/files/popup_72.html
Saved pop-up 73 -> Health/files/popup_73.html
Saved pop-up 74 -> Health/files/popup_74.html
Found 25 items on this page.
Saved pop-up 75 -> Health/files/popup_75.html
Saved pop-up 76 -> Health/files/popup_76.html
```

```
Saved pop-up 77 -> Health/files/popup_77.html
Saved pop-up 78 -> Health/files/popup_78.html
Saved pop-up 79 -> Health/files/popup_79.html
Saved pop-up 80 -> Health/files/popup_80.html
Saved pop-up 81 -> Health/files/popup_81.html
Saved pop-up 82 -> Health/files/popup_82.html
Saved pop-up 83 -> Health/files/popup_83.html
Saved pop-up 84 -> Health/files/popup_84.html
Saved pop-up 85 -> Health/files/popup_85.html
Saved pop-up 86 -> Health/files/popup_86.html
Saved pop-up 87 -> Health/files/popup_87.html
Saved pop-up 88 -> Health/files/popup_88.html
Saved pop-up 89 -> Health/files/popup_89.html
Saved pop-up 90 -> Health/files/popup_90.html
Saved pop-up 91 -> Health/files/popup_91.html
Saved pop-up 92 -> Health/files/popup_92.html
Saved pop-up 93 -> Health/files/popup_93.html
Saved pop-up 94 -> Health/files/popup_94.html
Saved pop-up 95 -> Health/files/popup_95.html
Saved pop-up 96 -> Health/files/popup_96.html
Saved pop-up 97 -> Health/files/popup_97.html
Saved pop-up 98 -> Health/files/popup_98.html
Saved pop-up 99 -> Health/files/popup_99.html
Found 25 items on this page.
Saved pop-up 100 -> Health/files/popup_100.html
Saved pop-up 101 -> Health/files/popup_101.html
Saved pop-up 102 -> Health/files/popup_102.html
Saved pop-up 103 -> Health/files/popup_103.html
Saved pop-up 104 -> Health/files/popup_104.html
Saved pop-up 105 -> Health/files/popup_105.html
Saved pop-up 106 -> Health/files/popup_106.html
Saved pop-up 107 -> Health/files/popup_107.html
Saved pop-up 108 -> Health/files/popup_108.html
Saved pop-up 109 -> Health/files/popup_109.html
Saved pop-up 110 -> Health/files/popup_110.html
Saved pop-up 111 -> Health/files/popup_111.html
Saved pop-up 112 -> Health/files/popup_112.html
Saved pop-up 113 -> Health/files/popup_113.html
Saved pop-up 114 -> Health/files/popup_114.html
Saved pop-up 115 -> Health/files/popup_115.html
Saved pop-up 116 -> Health/files/popup_116.html
Saved pop-up 117 -> Health/files/popup_117.html
```

```
Saved pop-up 118 -> Health/files/popup_118.html
Saved pop-up 119 -> Health/files/popup_119.html
Saved pop-up 120 -> Health/files/popup_120.html
Saved pop-up 121 -> Health/files/popup_121.html
Saved pop-up 122 -> Health/files/popup_122.html
Saved pop-up 123 -> Health/files/popup_123.html
Saved pop-up 124 -> Health/files/popup_124.html
Found 25 items on this page.
Saved pop-up 125 -> Health/files/popup_125.html
Saved pop-up 126 -> Health/files/popup_126.html
Saved pop-up 127 -> Health/files/popup_127.html
Saved pop-up 128 -> Health/files/popup_128.html
Saved pop-up 129 -> Health/files/popup_129.html
Saved pop-up 130 -> Health/files/popup_130.html
Saved pop-up 131 -> Health/files/popup_131.html
Saved pop-up 132 -> Health/files/popup_132.html
Saved pop-up 133 -> Health/files/popup_133.html
Saved pop-up 134 -> Health/files/popup_134.html
Saved pop-up 135 -> Health/files/popup_135.html
Saved pop-up 136 -> Health/files/popup_136.html
Saved pop-up 137 -> Health/files/popup_137.html
Saved pop-up 138 -> Health/files/popup_138.html
Saved pop-up 139 -> Health/files/popup_139.html
Saved pop-up 140 -> Health/files/popup_140.html
Saved pop-up 141 -> Health/files/popup_141.html
Saved pop-up 142 -> Health/files/popup_142.html
Saved pop-up 143 -> Health/files/popup_143.html
Saved pop-up 144 -> Health/files/popup_144.html
Saved pop-up 145 -> Health/files/popup_145.html
Saved pop-up 146 -> Health/files/popup_146.html
Saved pop-up 147 -> Health/files/popup_147.html
Saved pop-up 148 -> Health/files/popup_148.html
Saved pop-up 149 -> Health/files/popup_149.html
Found 25 items on this page.
Saved pop-up 150 -> Health/files/popup_150.html
Saved pop-up 151 -> Health/files/popup_151.html
Saved pop-up 152 -> Health/files/popup_152.html
Saved pop-up 153 -> Health/files/popup_153.html
Saved pop-up 154 -> Health/files/popup_154.html
Saved pop-up 155 -> Health/files/popup_155.html
Saved pop-up 156 -> Health/files/popup_156.html
Saved pop-up 157 -> Health/files/popup_157.html
```

```
Saved pop-up 158 -> Health/files/popup_158.html
Saved pop-up 159 -> Health/files/popup_159.html
Saved pop-up 160 -> Health/files/popup_160.html
Saved pop-up 161 -> Health/files/popup_161.html
Saved pop-up 162 -> Health/files/popup_162.html
Saved pop-up 163 -> Health/files/popup_163.html
Saved pop-up 164 -> Health/files/popup_164.html
Saved pop-up 165 -> Health/files/popup_165.html
Saved pop-up 166 -> Health/files/popup_166.html
Saved pop-up 167 -> Health/files/popup_167.html
Saved pop-up 168 -> Health/files/popup_168.html
Saved pop-up 169 -> Health/files/popup_169.html
Saved pop-up 170 -> Health/files/popup_170.html
Saved pop-up 171 -> Health/files/popup_171.html
Saved pop-up 172 -> Health/files/popup_172.html
Saved pop-up 173 -> Health/files/popup_173.html
Saved pop-up 174 -> Health/files/popup_174.html
Found 25 items on this page.
Saved pop-up 175 -> Health/files/popup_175.html
Saved pop-up 176 -> Health/files/popup_176.html
Saved pop-up 177 -> Health/files/popup_177.html
Saved pop-up 178 -> Health/files/popup_178.html
Saved pop-up 179 -> Health/files/popup_179.html
Saved pop-up 180 -> Health/files/popup_180.html
Saved pop-up 181 -> Health/files/popup_181.html
Saved pop-up 182 -> Health/files/popup_182.html
Saved pop-up 183 -> Health/files/popup_183.html
Saved pop-up 184 -> Health/files/popup_184.html
Saved pop-up 185 -> Health/files/popup_185.html
Saved pop-up 186 -> Health/files/popup_186.html
Saved pop-up 187 -> Health/files/popup_187.html
Saved pop-up 188 -> Health/files/popup_188.html
Saved pop-up 189 -> Health/files/popup_189.html
Saved pop-up 190 -> Health/files/popup_190.html
Saved pop-up 191 -> Health/files/popup_191.html
Saved pop-up 192 -> Health/files/popup_192.html
Saved pop-up 193 -> Health/files/popup_193.html
Saved pop-up 194 -> Health/files/popup_194.html
Saved pop-up 195 -> Health/files/popup_195.html
Saved pop-up 196 -> Health/files/popup_196.html
Saved pop-up 197 -> Health/files/popup_197.html
Saved pop-up 198 -> Health/files/popup_198.html
```

```
Saved pop-up 199 -> Health/files/popup_199.html
Found 25 items on this page.
Saved pop-up 200 -> Health/files/popup_200.html
Saved pop-up 201 -> Health/files/popup_201.html
Saved pop-up 202 -> Health/files/popup_202.html
Saved pop-up 203 -> Health/files/popup_203.html
Saved pop-up 204 -> Health/files/popup_204.html
Saved pop-up 205 -> Health/files/popup_205.html
Saved pop-up 206 -> Health/files/popup_206.html
Saved pop-up 207 -> Health/files/popup_207.html
Saved pop-up 208 -> Health/files/popup_208.html
Saved pop-up 209 -> Health/files/popup_209.html
Saved pop-up 210 -> Health/files/popup_210.html
Saved pop-up 211 -> Health/files/popup_211.html
Saved pop-up 212 -> Health/files/popup_212.html
Saved pop-up 213 -> Health/files/popup_213.html
Saved pop-up 214 -> Health/files/popup_214.html
Saved pop-up 215 -> Health/files/popup_215.html
Saved pop-up 216 -> Health/files/popup_216.html
Saved pop-up 217 -> Health/files/popup_217.html
Saved pop-up 218 -> Health/files/popup_218.html
Saved pop-up 219 -> Health/files/popup_219.html
Saved pop-up 220 -> Health/files/popup_220.html
Saved pop-up 221 -> Health/files/popup_221.html
Saved pop-up 222 -> Health/files/popup_222.html
Saved pop-up 223 -> Health/files/popup_223.html
Saved pop-up 224 -> Health/files/popup_224.html
Found 25 items on this page.
Saved pop-up 225 -> Health/files/popup_225.html
Saved pop-up 226 -> Health/files/popup_226.html
Saved pop-up 227 -> Health/files/popup_227.html
Saved pop-up 228 -> Health/files/popup_228.html
Saved pop-up 229 -> Health/files/popup_229.html
Saved pop-up 230 -> Health/files/popup_230.html
Saved pop-up 231 -> Health/files/popup_231.html
Saved pop-up 232 -> Health/files/popup_232.html
Saved pop-up 233 -> Health/files/popup_233.html
Saved pop-up 234 -> Health/files/popup_234.html
Saved pop-up 235 -> Health/files/popup_235.html
Saved pop-up 236 -> Health/files/popup_236.html
Saved pop-up 237 -> Health/files/popup_237.html
Saved pop-up 238 -> Health/files/popup_238.html
```

```
Saved pop-up 239 -> Health/files/popup_239.html
Saved pop-up 240 -> Health/files/popup_240.html
Saved pop-up 241 -> Health/files/popup_241.html
Saved pop-up 242 -> Health/files/popup_242.html
Saved pop-up 243 -> Health/files/popup_243.html
Saved pop-up 244 -> Health/files/popup_244.html
Saved pop-up 245 -> Health/files/popup_245.html
Saved pop-up 246 -> Health/files/popup_246.html
Saved pop-up 247 -> Health/files/popup_247.html
Saved pop-up 248 -> Health/files/popup_248.html
Saved pop-up 249 -> Health/files/popup_249.html
Reached the last page. Stopping.
```

In [3]:

```python
import os
import re
import pandas as pd
from bs4 import BeautifulSoup

def parse_popup_files(directory):
    inspections = []

    if not os.path.exists(directory):
        print(f"Directory '{directory}' does not exist.")
        return pd.DataFrame()

    # Pre-compile regex for performance
    # Matches patterns like "05/12/2024 - Routine Inspection (Routine)" or simple dates
    title_pattern = re.compile(r"^(.*?)(?:\s*-\s*(.*?))?(?:\s*\((.*?)\))?$")

    for filename in os.listdir(directory):
        if not filename.endswith(".html"):
            continue

        file_path = os.path.join(directory, filename)

        with open(file_path, 'r', encoding='utf-8') as f:
            # Using 'lxml' for significantly faster HTML parsing
            soup = BeautifulSoup(f, 'lxml')

        # Helper to extract text safely without repetitive try/except or if/else
        def get_text(element, selector, attr='data-bind'):
            node = element.find(selector, {attr: re.compile(r'\b' + re.escape(attr_value) + r'\b')}) if is
```

```
            return node.text.strip() if node else None

        # --- Establishment Details ---
        name_node = soup.find('span', {'data-bind': 'text: name'})
        street_node = soup.find('span', {'data-bind': 'text: street'})
        csz_node = soup.find('span', {'data-bind': 'text: city + \', \' + state + \' \' + zip'})
        phone_node = soup.find('span', {'data-bind': 'text: phone'})

        name = name_node.text.strip() if name_node else None
        street = street_node.text.strip() if street_node else None
        phone = phone_node.text.strip() if phone_node else None

        # Parse City, State, and Zip separately
        city, state, zip_code = None, None, None
        if csz_node and csz_node.text.strip():
            csz_text = csz_node.text.strip()
            # Expecting "City, ST 12345"
            match_csz = re.match(r"^([^,]+),\s*([A-Z]{2})\s*(\d{5}(?:-\d{4})?)$", csz_text)
            if match_csz:
                city, state, zip_code = match_csz.group(1), match_csz.group(2), match_csz.group(3)
            else:
                city = csz_text  # Fallback to full string if format deviates

        # --- Inspections ---
        # Look for panel containers rather than isolated title blocks to scoped search
        panels = soup.find_all('div', class_=re.compile(r'panel')) or [soup]

        for panel in panels:
            title_node = panel.find('div', {'data-bind': 'html: displayTitle'}) or panel.find(attrs={'data-
            if not title_node:
                continue

            inspection_date_full = title_node.text.strip()

            # Parse inspection date & type
            match_title = title_pattern.match(inspection_date_full)
            if match_title:
                inspection_date = match_title.group(1).strip() if match_title.group(1) else inspection_dat
                inspection_type = match_title.group(3).strip() if match_title.group(3) else (match_title.g
            else:
                inspection_date = inspection_date_full
                inspection_type = None
```

```
            # Scoped lookup inside panel to avoid grabbing the first item repeatedly
            start_time_node = panel.find('span', {'data-bind': 'html: inspectionTime'})
            end_time_node = panel.find('span', {'data-bind': 'html: inspectionTimeout'})
            inspector_node = panel.find('span', {'data-bind': 'html: inspectionInspector'})

            inspections.append({
                'Source File': filename,
                'Name': name,
                'Street': street,
                'City': city,
                'State': state,
                'Zip': zip_code,
                'Phone': phone,
                'Inspection Date': inspection_date,
                'Inspection Type': inspection_type,
                'Start Time': start_time_node.text.strip() if start_time_node else None,
                'End Time': end_time_node.text.strip() if end_time_node else None,
                'Inspector': inspector_node.text.strip() if inspector_node else None
            })

    return pd.DataFrame(inspections)


if __name__ == "__main__":
    directory = "Health/files"
    df = parse_popup_files(directory)

    if not df.empty:
        # Deduplicate records if files overlap
        df.drop_duplicates(subset=['Name', 'Inspection Date', 'Start Time'], inplace=True)

        output_path = 'Health/health_department_inspections_JUL_2026.csv'
        df.to_csv(output_path, index=False)
        print(f"Successfully processed {len(df)} records -> saved to {output_path}")
    else:
        print("No inspection records were extracted.")
```

```
Successfully processed 7346 records -> saved to Health/health_department_inspections_JUL_2026.csv
```

In [1]:
```
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
```

In [4]:
```
# Load the data
df = pd.read_csv('Health/health_department_inspections_JUL_2026.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	Source File	Name	Street	City	State	Zip	Phone	Inspection Date	Inspection Type	Start Time	End Time	Inspector
0	popup_35.html	Crocker VFW Post 4956	13165 Hwy 17	Crocker	MO	65452.0	(573) 736-2455	07/01/2026	Routine	2:06 PM	3:10 PM	Zachary Marlow
1	popup_35.html	Crocker VFW Post 4956	13165 Hwy 17	Crocker	MO	65452.0	(573) 736-2455	07/01/2026	Routine	NaN	NaN	NaN
2	popup_35.html	Crocker VFW Post 4956	13165 Hwy 17	Crocker	MO	65452.0	(573) 736-2455	08/27/2025	Follow-up	2:40 PM	3:17 PM	Amy Osborn
3	popup_35.html	Crocker VFW Post 4956	13165 Hwy 17	Crocker	MO	65452.0	(573) 736-2455	08/27/2025	Follow-up	NaN	NaN	NaN
4	popup_35.html	Crocker VFW Post 4956	13165 Hwy 17	Crocker	MO	65452.0	(573) 736-2455	07/02/2025	Routine	3:10 PM	4:10 PM	Amy Osborn

In [6]:
```
df.dtypes
```

Out[6]:
```
Source File         object
Name                object
Street              object
City                object
State               object
Zip                float64
Phone               object
Inspection Date     object
Inspection Type     object
Start Time          object
End Time            object
Inspector           object
dtype: object
```

In [8]:
```
# List of columns to cast to string
columns_to_string = ['Name', 'Street', 'City', 'State', 'Zip', 'Inspection Type', 'Inspector']

# Cast the columns to string
df[columns_to_string] = df[columns_to_string].astype(str)
```

In [18]:
```
# Convert to Pandas datetime format (datetime64[ns])
df['Inspection Date'] = pd.to_datetime(df['Inspection Date'], errors='coerce')
```

In [9]:
```
df.isna().sum()
```

Out[9]:
```
Source File          0
Name                 0
Street               0
City                 0
State                0
Zip                  0
Phone               77
Inspection Date      0
Inspection Type      0
Start Time        3659
End Time          5080
Inspector            0
dtype: int64
```

In [10]:
```
# 1. Sort so that rows with actual values (non-NaN) come before rows with NaNs
df = df.sort_values(
```

```
    by=['Name', 'Inspection Date', 'Start Time', 'Inspector'],
    na_position='last'
)

# 2. Drop duplicates based on Establishment Name & Inspection Date, keeping the first (non-NaN) row
df = df.drop_duplicates(
    subset=['Name', 'Inspection Date'],
    keep='first'
)

# 3. Reset index for clean numbering
df = df.reset_index(drop=True)
```

In [11]:
```
df.head()
```

Out[11]:

	Source File	Name	Street	City	State	Zip	Phone	Inspection Date	Inspection Type	Start Time	End Time	Inspector
0	popup_0.html	BUDGET INN (Lodging)	127 FINA DRIVE	ST ROBERT	MO	65584.0	(573) 336-5212	02/26/2024	Initial	9:30 AM	10:30 AM	Zachary Marlow
1	popup_0.html	BUDGET INN (Lodging)	127 FINA DRIVE	ST ROBERT	MO	65584.0	(573) 336-5212	03/01/2021	Follow-up	12:00 PM	1:00 PM	Israel Doba
2	popup_0.html	BUDGET INN (Lodging)	127 FINA DRIVE	ST ROBERT	MO	65584.0	(573) 336-5212	03/02/2026	Initial	9:30 AM	10:30 AM	Zachary Marlow
3	popup_0.html	BUDGET INN (Lodging)	127 FINA DRIVE	ST ROBERT	MO	65584.0	(573) 336-5212	04/06/2026	Follow-up	11:10 AM	11:45 AM	Zachary Marlow
4	popup_0.html	BUDGET INN (Lodging)	127 FINA DRIVE	ST ROBERT	MO	65584.0	(573) 336-5212	05/05/2025	Initial	10:15 AM	11:15 PM	Amy Osborn

In [12]:
```
df.isna().sum()
```

Out[12]:
```
Source File           0
Name                  0
Street                0
City                  0
State                 0
Zip                   0
Phone                39
Inspection Date       0
Inspection Type       0
Start Time            0
End Time           1421
Inspector             0
dtype: int64
```

In [13]:
```
# Check unique values in 'Start Time'
unique_start_times = df['Start Time'].unique()
print("Unique Start Times:")
print(unique_start_times)
```

```
Unique Start Times:
['9:30 AM' '12:00 PM' '11:10 AM' '10:15 AM' '1:00 PM' '2:30 PM' '00:00 AM'
 '12:30 PM' '10:00 AM' '11:50 AM' '1:02 PM' '11:45 AM' '10:48 AM'
 '7:00 PM' '10:50 AM' '3:38 PM' '2:02 PM' '10:30 AM' '11:00 AM' '12:45 PM'
 '10:58 AM' '1:30 PM' '1:59 PM' '12:33 PM' '11:35 AM' '2:10 PM' '12:05 PM'
 '2:00 PM' '1:29 PM' '1:10 PM' '12:15 PM' '2:40 PM' '10:40 AM' '10:59 AM'
 '11:25 AM' '9:03 AM' '1:45 PM' '11:15 AM' '10:45 AM' '1:20 PM' '10:08 AM'
 '2:20 PM' '11:41 AM' '2:47 PM' '1:15 PM' '8:49 AM' '11:03 AM' '10:04 AM'
 '2:50 PM' '10:12 AM' '10:54 AM' '12:50 PM' '11:02 AM' '12:43 PM'
 '1:50 PM' '11:20 AM' '1:25 PM' '1:08 PM' '11:22 AM' '2:36 PM' '1:36 PM'
 '12:45 AM' '9:00 AM' '12:16 PM' '11:19 AM' '11:05 AM' '1:35 PM'
 '12:31 PM' '9:38 AM' '9:45 AM' '1:21 PM' '12:11 PM' '3:30 PM' '12:14 PM'
 '10:49 AM' '11:40 AM' '11:30 AM' '1:16 PM' '7:41 AM' '12:51 PM'
 '10:05 AM' '9:50 AM' '9:55 AM' '1:01 PM' '9:20 AM' '1:40 PM' '12:35 PM'
 '8:02 AM' '9:56 AM' '12:10 PM' '10:35 AM' '3:45 PM' '10:07 AM' '2:05 PM'
 '1:34 PM' '7:40 AM' '8:48 AM' '1:28 PM' '5:10 PM' '3:14 PM' '12:55 PM'
 '11:18 AM' '2:32 PM' '3:25 PM' '12:22 PM' '12:20 PM' '3:15 PM' '10:39 AM'
 '9:15 AM' '12:40 PM' '11:45 PM' '9:57 AM' '9:14 AM' '12:58 PM' '3:01 PM'
 '6:30 PM' '11:38 AM' '12:53 PM' '4:04 PM' '1:19 PM' '8:30 AM' '7:50 AM'
 '12:28 PM' '9:53 AM' '1:12 PM' '1:32 PM' '2:17 PM' '7:20 AM' '1:14 PM'
 '12:12 PM' '10:25 AM' '7:37 AM' '9:25 AM' '1:37 PM' '11:12 AM' '12:46 PM'
 '10:10 AM' '11:07 AM' '2:41 PM' '9:40 AM' '1:03 PM' '10:20 AM' '4:30 PM'
 '5:45 PM' '2:45 AM' '10:02 AM' '12:02 PM' '8:45 AM' '1:26 PM' '8:24 AM'
 '12:28 AM' '9:08 AM' '10:31 AM' '10:01 AM' '10:33 AM' '10:21 AM'
 '11:06 AM' '2:06 PM' '3:10 PM' '2:45 PM' '3:41 PM' '12:42 PM' '10:42 AM'
 '10:09 AM' '1:30 AM' '1:31 PM' '10:53 AM' '9:27 AM' '3:35 PM' '2:40 AM'
 '8:15 AM' '10:38 AM' '8:31 AM' '8:20 AM' '10:26 AM' '8:00 AM' '9:51 AM'
 '9:35 AM' '10:51 AM' '9:52 AM' '9:47 AM' '9:28 AM' '10:37 AM' '8:55 AM'
 '9:06 AM' '9:34 AM' '2:15 PM' '4:01 PM' '12:15 AM' '10:28 AM' '9:10 AM'
 '11:30 PM' '11:31 AM' '2:01 PM' '12:01 PM' '8:33 AM' '3:48 PM' '5:15 PM'
 '12:56 PM' '2:55 PM' '11:01 AM' '1:06 PM' '11:47 AM' '11:33 AM'
 '11:04 AM' '11:42 AM' '11:46 PM' '3:40 PM' '12:52 PM' '4:07 PM'
 '12:26 PM' '2:09 PM' '1:11 PM' '1:05 PM' '3:00 PM' '12:23 PM' '2:16 PM'
 '11:57 AM' '2:26 PM' '12:04 PM' '10:47 AM' '11:16 AM' '10:24 AM'
 '10:17 AM' '11:55 AM' '12:00 AM' '12:13 PM' '7:45 AM' '7:35 AM'
 '12:39 PM' '6:10 PM' '2:33 PM' '11:24 AM' '9:07 AM' '11:55 PM' '8:35 AM'
 '1:55 PM' '1:57 PM' '12:35 AM' '2:11 PM' '3:02 PM' '8:40 AM' '3:32 PM'
 '12:06 PM' '10:32 AM' '12:24 PM' '2:29 PM' '1:48 PM' '8:10 AM' '12:03 PM'
 '11:08 AM' '1:23 PM' '11:51 AM' '11:29 AM' '12:57 PM' '12:08 PM'
 '11:00 PM' '11:32 AM' '1:02 AM' '1:49 PM' '7:30 AM' '10:27 AM' '4:20 PM'
 '11:20 PM' '1:22 PM' '10:30 PM' '12:36 PM' '2:31 PM' '3:03 AM' '4:27 PM'
 '12:09 PM' '10:29 AM' '11:28 AM' '1:07 PM' '1:17 PM' '11:46 AM'
```

```
'12:17 PM' '1:18 PM' '12:37 PM' '1:27 PM' '4:33 PM' '4:35 PM' '1:04 PM'
'12:19 PM' '3:03 PM' '1:43 PM' '3:23 PM' '9:05 AM' '3:20 PM' '2:43 PM'
'11:09 AM' '11:43 AM' '2:03 PM' '4:40 PM' '10:19 AM' '11:15 PM' '5:31 PM'
'10:52 AM' '9:19 AM' '5:38 PM' '3:11 PM' '12:44 PM' '12:05 AM' '11:17 AM'
'12:21 PM' '2:22 PM' '2:59 PM' '10:03 AM' '8:18 AM' '10:43 AM' '10:46 AM'
'2:23 PM' '12:47 PM' '10:57 AM' '11:53 AM' '11:26 AM' '11:44 AM'
'2:49 PM' '6:00 AM' '3:08 PM' '4:15 PM' '8:58 AM' '9:31 AM' '10:55 AM'
'8:19 AM' '8:14 AM' '2:29 AM' '2:13 PM' '12:29 PM' '1:46 PM' '11:50 PM'
'9:16 AM' '3:36 PM' '8:43 AM' '11:54 PM' '1:38 PM' '7:05 AM' '3:26 PM'
'12:38 PM' '12:59 PM' '1:51 PM' '11:13 AM' '2:48 PM' '11:37 AM' '2:07 PM'
'5:13 PM' '4:09 PM' '10:06 AM' '1:53 PM' '2:25 PM' '11:40 PM' '1:54 PM'
'3:50 PM' '3:21 PM' '9:31 PM' '1:41 PM' '12:25 PM' '2:12 PM' '9:26 AM'
'8:50 AM' '3:54 PM' '1:58 PM' '12:49 PM' '9:29 AM' '8:11 AM' '11:59 AM'
'9:32 AM' '1:33 PM' '2:35 PM' '8:22 AM' '9:01 AM' '4:25 PM' '3:05 PM'
'11:27 AM' '10:14 AM' '8:39 AM' '9:21 AM' '4:59 PM' '10:45 PM' '4:00 PM'
'11:48 AM' '3:33 PM' '1:09 PM' '2:54 PM' '10:41 AM' '10:36 AM' '9:59 AM'
'11:36 AM' '12:41 PM' '10:44 AM' '9:44 AM' '10:00 PM' '9:09 AM' '9:54 AM'
'6:30 AM' '3:24 PM' '4:29 PM' '12:07 PM' '11:21 AM' '2:21 PM' '8:05 AM'
'2:04 PM' '1:24 PM' '3:29 PM' '7:11 PM' '4:20 AM' '2:42 PM' '4:06 PM'
'8:25 AM' '2:18 PM' '3:46 PM' '11:39 AM' '1:39 PM' '10:11 AM' '4:02 PM'
'2:44 PM' '9:58 AM' '7:00 AM' '2:28 PM' '3:28 PM' '10:16 AM' '2:57 PM'
'2:14 PM' '1:13 PM' '11:14 AM' '2:08 PM' '1:44 PM' '4:12 PM' '4:18 PM'
'3:06 PM' '5:00 PM' '10:22 AM' '9:48 AM' '9:46 AM' '11:11 AM' '9:12 AM'
'9:43 AM' '9:49 AM' '2:34 PM' '11:58 AM' '8:38 AM' '10:13 AM' '4:19 PM'
'5:33 PM' '5:32 PM' '11:54 AM']
```

In [14]:

```python
# Check unique values in 'End Time'
unique_end_times = df['End Time'].unique()
print("\nUnique End Times:")
print(unique_end_times)
```

```
Unique End Times:
['10:30 AM' '1:00 PM' '11:45 AM' '11:15 PM' '2:15 PM' '3:00 PM' nan
 '2:09 PM' '11:15 AM' '1:48 PM' '1:12 PM' '1:30 PM' '12:21 AM' '10:31 AM'
 '1:08 PM' '11:50 AM' '8:01 PM' '3:50 AM' '4:02 PM' '11:30 AM' '12:30 PM'
 '1:45 PM' '2:00 PM' '12:41 PM' '12:50 PM' '2:30 PM' '2:37 PM' '12:53 PM'
 '11:40 AM' '11:54 PM' '11:19 AM' '3:12 PM' '1:13 PM' '3:05 PM' '1:34 PM'
 '1:23 PM' '1:15 PM' '3:15 PM' '12:00 AM' '1:28 PM' '11:37 AM' '12:15 PM'
 '9:30 AM' '10:00 AM' '1:20 PM' '2:10 PM' '2:02 PM' '12:08 PM' '12:45 PM'
 '11:00 AM' '12:40 PM' '2:45 AM' '3:58 PM' '12:21 PM' '11:05 AM' '1:40 PM'
 '3:46 PM' '1:35 PM' '4:00 PM' '12:12 PM' '1:47 PM' '11:26 AM' '11:36 AM'
 '1:25 PM' '12:38 PM' '1:24 PM' '12:00 PM' '1:05 PM' '12:05 PM' '2:54 PM'
 '11:55 AM' '3:23 PM' '2:39 PM' '2:21 PM' '10:09 AM' '1:43 PM' '2:00 AM'
 '1:17 PM' '12:10 PM' '11:25 AM' '1:50 PM' '2:35 PM' '10:43 AM' '10:35 AM'
 '2:04 PM' '5:00 PM' '2:20 PM' '2:40 PM' '10:52 AM' '12:26 PM' '8:23 AM'
 '10:40 AM' '10:55 AM' '1:19 PM' '10:25 AM' '2:45 PM' '9:05 AM' '1:10 PM'
 '11:35 AM' '1:22 PM' '11:53 PM' '4:11 PM' '10:45 AM' '11:10 AM'
 '11:59 AM' '3:50 PM' '2:05 PM' '9:47 AM' '9:34 AM' '12:15 AM' '6:15 PM'
 '2:48 PM' '3:43 PM' '1:58 PM' '12:17 PM' '12:40 AM' '12:37 PM' '4:15 PM'
 '11:17 AM' '12:35 PM' '10:20 AM' '12:20 PM' '1:16 PM' '10:33 AM'
 '10:06 AM' '3:41 PM' '8:20 PM' '12:31 PM' '1:27 PM' '2:23 PM' '4:30 PM'
 '2:22 PM' '3:30 PM' '11:34 AM' '8:38 AM' '11:01 AM' '12:04 AM' '12:43 PM'
 '11:44 AM' '11:13 AM' '2:43 PM' '8:54 AM' '11:03 AM' '12:49 PM'
 '10:41 AM' '8:40 AM' '12:55 PM' '3:59 PM' '11:45 PM' '1:49 PM' '8:53 AM'
 '3:29 PM' '9:45 AM' '11:11 AM' '12:06 PM' '11:30 PM' '5:30 PM' '5:52 PM'
 '3:15 AM' '10:50 AM' '10:13 AM' '9:15 AM' '10:32 AM' '2:44 PM' '9:00 AM'
 '11:25 PM' '10:56 AM' '9:29 AM' '10:33 PM' '12:54 PM' '11:35 PM'
 '11:42 AM' '3:10 PM' '4:10 PM' '3:45 PM' '3:17 PM' '2:25 PM' '3:20 PM'
 '4:19 PM' '1:21 PM' '11:08 AM' '2:50 PM' '3:35 PM' '11:33 AM' '12:46 PM'
 '10:20 PM' '10:10 AM' '11:41 AM' '11:20 AM' '10:30 PM' '12:11 PM'
 '1:36 PM' '9:20 AM' '9:33 AM' '8:50 AM' '11:05 PM' '11:43 AM' '10:05 AM'
 '8:40 PM' '11:23 AM' '10:39 AM' '11:54 AM' '9:45 PM' '10:29 AM'
 '10:51 PM' '10:14 AM' '10:15 AM' '9:53 AM' '11:39 AM' '1:55 AM'
 '10:03 AM' '12:58 PM' '10:35 PM' '3:24 PM' '9:30 PM' '10:10 PM' '3:02 PM'
 '3:22 PM' '4:41 PM' '12:07 AM' '2:12 PM' '1:57 PM' '3:09 PM' '10:00 PM'
 '1:38 PM' '12:25 PM' '11:31 AM' '4:34 PM' '4:20 PM' '6:37 PM' '2:14 PM'
 '1:42 PM' '3:00 AM' '2:16 PM' '12:44 PM' '11:47 AM' '3:42 PM' '3:31 PM'
 '4:52 PM' '5:14 PM' '1:55 PM' '1:02 PM' '3:51 PM' '1:29 PM' '11:21 AM'
 '1:18 PM' '12:18 PM' '1:52 PM' '11:02 AM' '4:22 PM' '4:27 PM' '1:26 PM'
 '8:30 AM' '3:27 PM' '9:15 PM' '8:55 AM' '8:35 AM' '4:57 PM' '6:00 PM'
 '7:20 PM' '3:49 PM' '1:14 PM' '1:53 PM' '3:44 PM' '2:53 PM' '12:04 PM'
 '3:33 PM' '10:55 PM' '9:40 AM' '5:34 PM' '9:55 PM' '9:50 AM' '11:13 PM'
 '10:38 PM' '2:07 PM' '1:37 PM' '2:36 PM' '11:12 AM' '2:18 PM' '1:09 PM'
```

```
'12:01 PM' '1:07 PM' '10:12 AM' '1:41 PM' '11:12 PM' '12:19 PM' '9:55 AM'
'12:27 PM' '12:07 PM' '4:05 PM' '10:45 PM' '4:58 PM' '4:46 PM' '10:58 AM'
'2:58 PM' '11:20 PM' '1:06 PM' '11:56 AM' '5:18 PM' '11:41 PM' '12:33 PM'
'2:27 PM' '2:41 PM' '3:03 PM' '2:31 PM' '2:56 PM' '2:32 PM' '8:45 AM'
'1:46 PM' '5:24 PM' '1:31 PM' '2:29 PM' '10:53 AM' '3:55 PM' '11:00 PM'
'2:13 PM' '5:20 PM' '10:24 AM' '3:40 PM' '4:45 PM' '2:47 PM' '4:08 PM'
'4:44 PM' '10:42 AM' '5:50 PM' '11:48 AM' '5:42 PM' '9:52 AM' '4:47 PM'
'1:11 PM' '2:42 PM' '3:16 PM' '11:14 AM' '10:34 AM' '10:46 AM' '11:23 PM'
'2:34 PM' '2:49 PM' '10:05 PM' '3:37 PM' '12:57 PM' '12:29 PM' '10:27 AM'
'12:47 PM' '2:46 PM' '11:22 AM' '7:00 AM' '12:36 PM' '11:28 AM' '4:37 PM'
'1:39 PM' '12:03 PM' '10:16 AM' '9:25 AM' '9:17 AM' '10:36 AM' '12:51 PM'
'10:07 AM' '9:40 PM' '11:34 PM' '11:10 PM' '11:52 AM' '9:49 AM'
'12:02 PM' '3:25 PM' '12:35 AM' '3:26 PM' '12:48 PM' '2:24 PM' '12:59 PM'
'4:20 AM' '9:50 PM' '8:00 AM' '1:51 PM' '5:05 PM' '1:03 PM' '7:35 AM'
'4:36 PM' '11:29 AM' '1:00 AM' '3:11 PM' '6:16 PM' '5:07 PM' '11:40 PM'
'12:14 PM' '2:19 PM' '12:22 PM' '5:09 PM' '4:07 PM' '3:47 PM' '1:56 PM'
'11:51 AM' '12:05 AM' '2:52 PM' '11:46 AM' '10:44 AM' '9:09 AM'
'11:32 AM' '10:50 PM' '11:31 PM' '11:03 PM' '10:09 PM' '11:09 AM'
'3:29 AM' '1:54 PM' '12:28 PM' '11:55 PM' '2:33 PM' '2:38 PM' '4:50 PM'
'4:55 PM' '1:59 PM' '4:51 PM' '5:15 PM' '9:07 AM' '3:30 AM' '3:07 PM'
'10:04 AM' '5:10 PM' '11:49 PM' '10:59 AM' '11:29 PM' '9:12 AM'
'12:10 AM' '9:32 AM' '11:49 AM' '4:18 PM' '4:39 PM' '11:16 AM' '12:03 AM'
'3:04 PM' '12:20 AM' '10:37 AM' '11:07 AM' '9:31 PM' '4:06 PM' '1:32 PM'
'11:38 AM' '3:45 AM' '7:30 AM' '10:57 AM' '12:24 PM' '4:43 PM' '5:01 PM'
'2:03 PM' '9:58 AM' '10:26 AM' '1:01 PM' '6:30 PM' '3:08 PM' '2:55 PM'
'10:25 PM' '7:50 AM' '3:38 AM' '11:58 AM' '4:25 PM' '11:44 PM' '3:19 PM'
'12:30 AM' '3:21 PM' '3:06 PM' '11:51 PM' '12:39 PM' '12:09 PM' '1:04 PM'
'3:28 PM' '10:02 AM' '9:48 AM' '11:04 AM' '8:37 AM' '4:01 PM' '2:28 PM'
'11:33 PM' '1:33 PM' '5:23 PM' '4:14 PM' '5:21 PM' '4:40 PM' '11:06 AM'
'10:48 AM' '2:01 PM' '11:48 PM' '9:10 AM' '10:31 PM' '10:08 AM'
'10:17 AM' '2:30 AM' '5:16 PM' '9:35 AM' '8:30 PM' '4:21 PM' '5:37 PM'
'11:27 PM' '9:57 PM' '4:49 PM' '6:20 PM']
```

In [24]:
```
# Convert string times to Python time objects (datetime.time)
df['Start Time'] = pd.to_datetime(df['Start Time'], format='%I:%M %p', errors='coerce').dt.time
df['End Time'] = pd.to_datetime(df['End Time'], format='%I:%M %p', errors='coerce').dt.time
```

In [15]:
```
columns_to_check = ['Inspector', 'Inspection Type']

for column in columns_to_check:
    print(f"Unique values in '{column}':")
```

```
    print(df[column].unique())
    print("\n")
```

```
Unique values in 'Inspector':
['Zachary  Marlow' 'Israel Doba' 'Zachary Marlow' 'Amy Osborn'
 'Karen Wall' 'Hector Silva' 'Elaine Clark' 'Isreal Doba'
 'Old Inspections' 'Elaine E. Clark' 'Dan Cordova' 'Karen K. Wall'
 'Hunter Barnett']


Unique values in 'Inspection Type':
['Initial' 'Follow-up' 'Pre-opening' 'Routine' 'Inspection' 'Follow-Up'
 'Pre-Opening' 'Annual' 'Re-inspection' 'Complaint' 'Renewal' 'Other'
 'Special Circumstances']
```

In [16]:

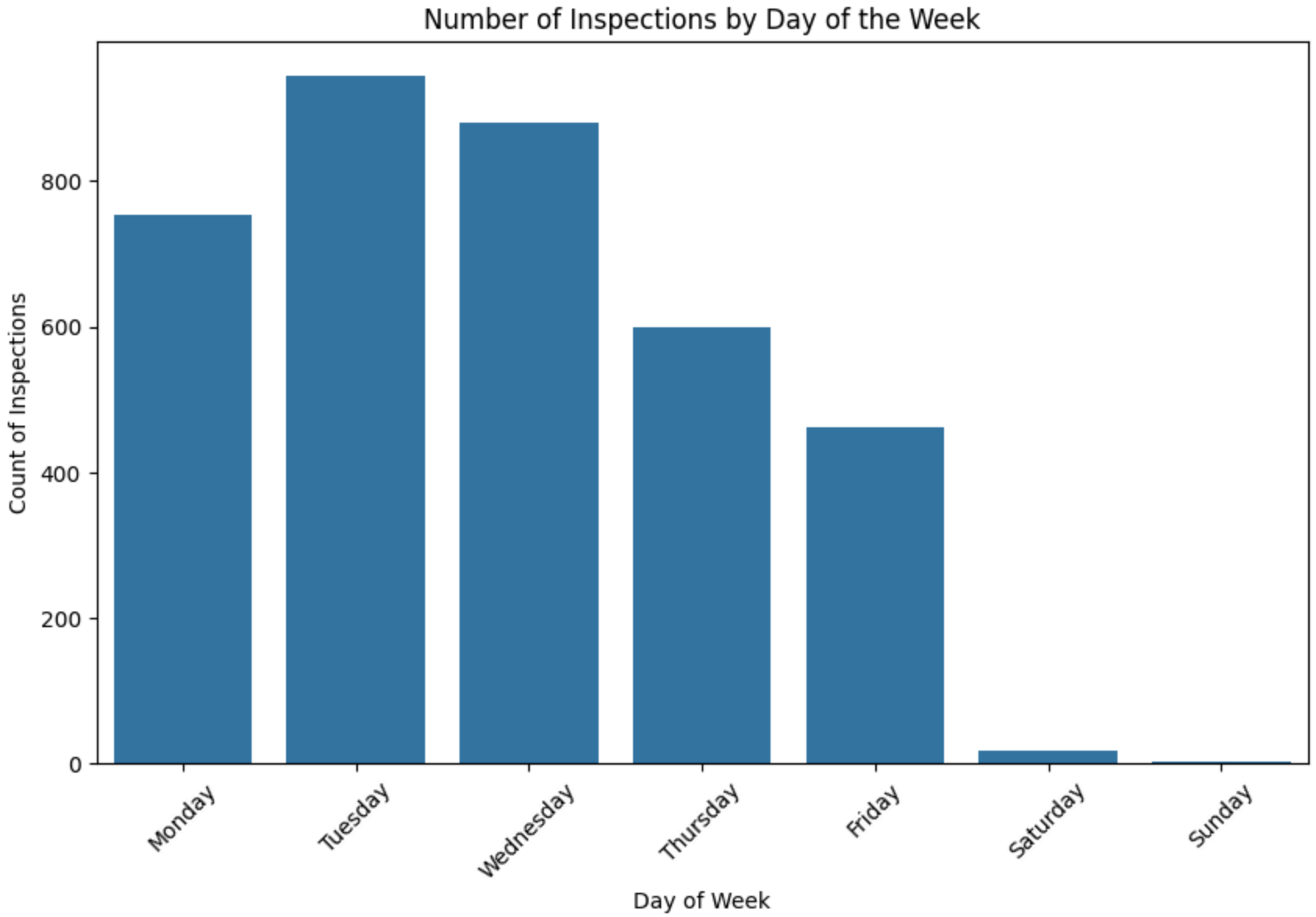
```
# 1. Standardize Inspector Names
inspector_map = {
    # Fix typos & extra spaces
    'Zachary  Marlow': 'Zachary Marlow',
    'Isreal Doba': 'Israel Doba',

    # Standardize full names / middle initials
    'Elaine E. Clark': 'Elaine Clark',
    'Karen K. Wall': 'Karen Wall',

    # Remove non-person values
    'Old Inspections': None  # Or 'Unknown'
}

# 2. Standardize Inspection Types
type_map = {
    # Fix casing & hyphenation inconsistencies
    'Follow-Up': 'Follow-up',
    'Pre-Opening': 'Pre-opening',

    # Standardize synonyms
    'Re-inspection': 'Follow-up',
    'Inspection': 'Routine',
    'Annual': 'Routine'
}
```

```
# Apply mappings
df['Inspector'] = df['Inspector'].replace(inspector_map)
df['Inspection Type'] = df['Inspection Type'].replace(type_map)

# Optional: Trim trailing whitespace across all string columns
df['Inspector'] = df['Inspector'].str.strip()
df['Inspection Type'] = df['Inspection Type'].str.strip()
```

In [19]:
```
# Calculate the cutoff date
cutoff_date = datetime.now() - timedelta(days=18 * 30)  # Approximate 18 months as 30 days/month

# Find the most recent inspection date for each establishment
recent_inspections = df.groupby('Name', as_index=False)['Inspection Date'].max()

# Filter establishments not inspected within the last 18 months
not_recently_inspected = recent_inspections[recent_inspections['Inspection Date'] < cutoff_date]

# Save the filtered data to a CSV or display it
#not_recently_inspected.to_csv('not_inspected_last_18_months.csv', index=False)
print(len(not_recently_inspected))
print(not_recently_inspected)
```

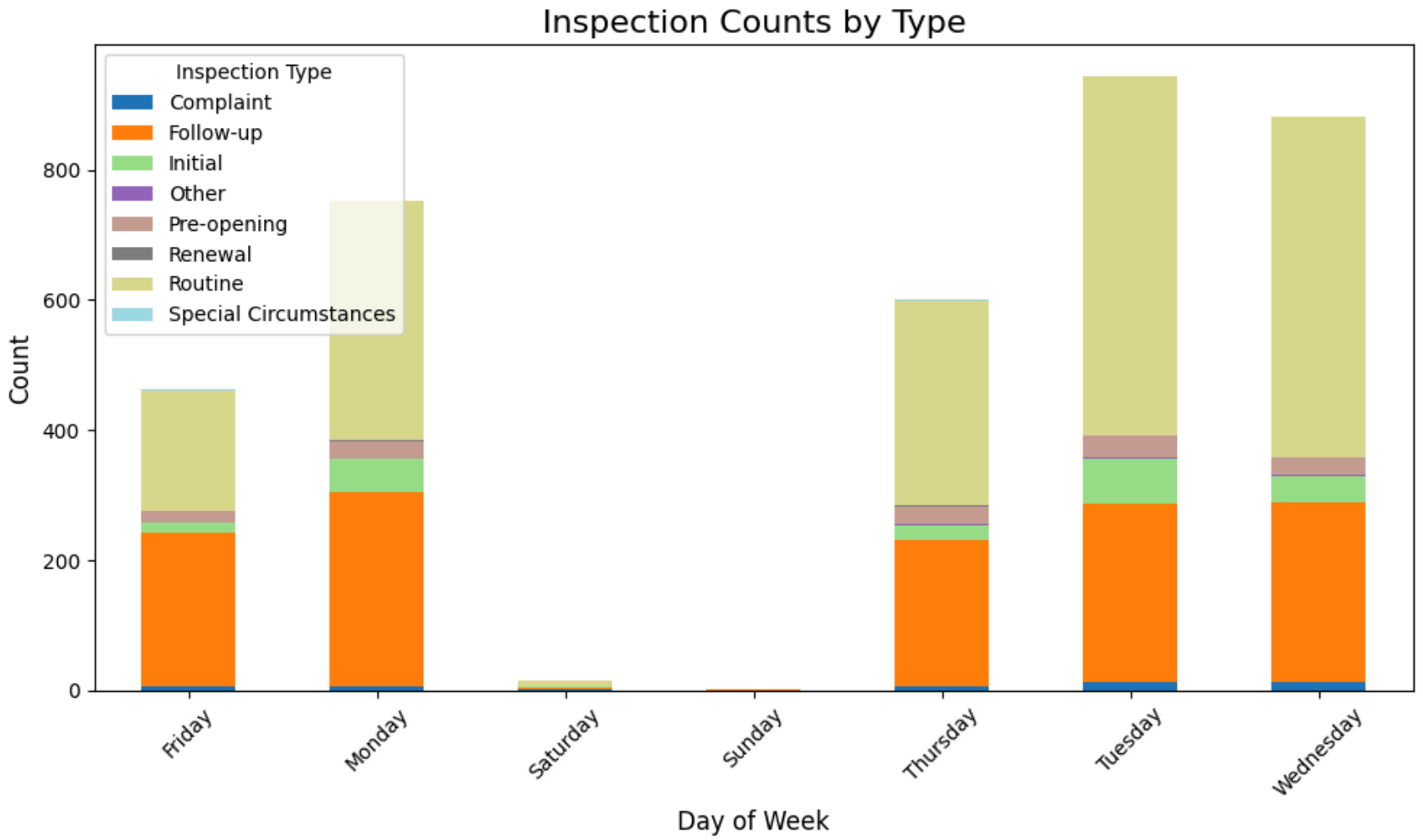
```
2
            Name Inspection Date
28     Crepescape      2024-12-17
153  Prep Kitchen      2024-10-25
```

In [20]:
```
# Add a column for the day of the week
df['Day of Week'] = df['Inspection Date'].dt.day_name()

# Countplot for inspections by day of the week
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Day of Week', order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Sa
plt.title('Number of Inspections by Day of the Week')
plt.xlabel('Day of Week')
plt.ylabel('Count of Inspections')
plt.xticks(rotation=45)
plt.show()
```



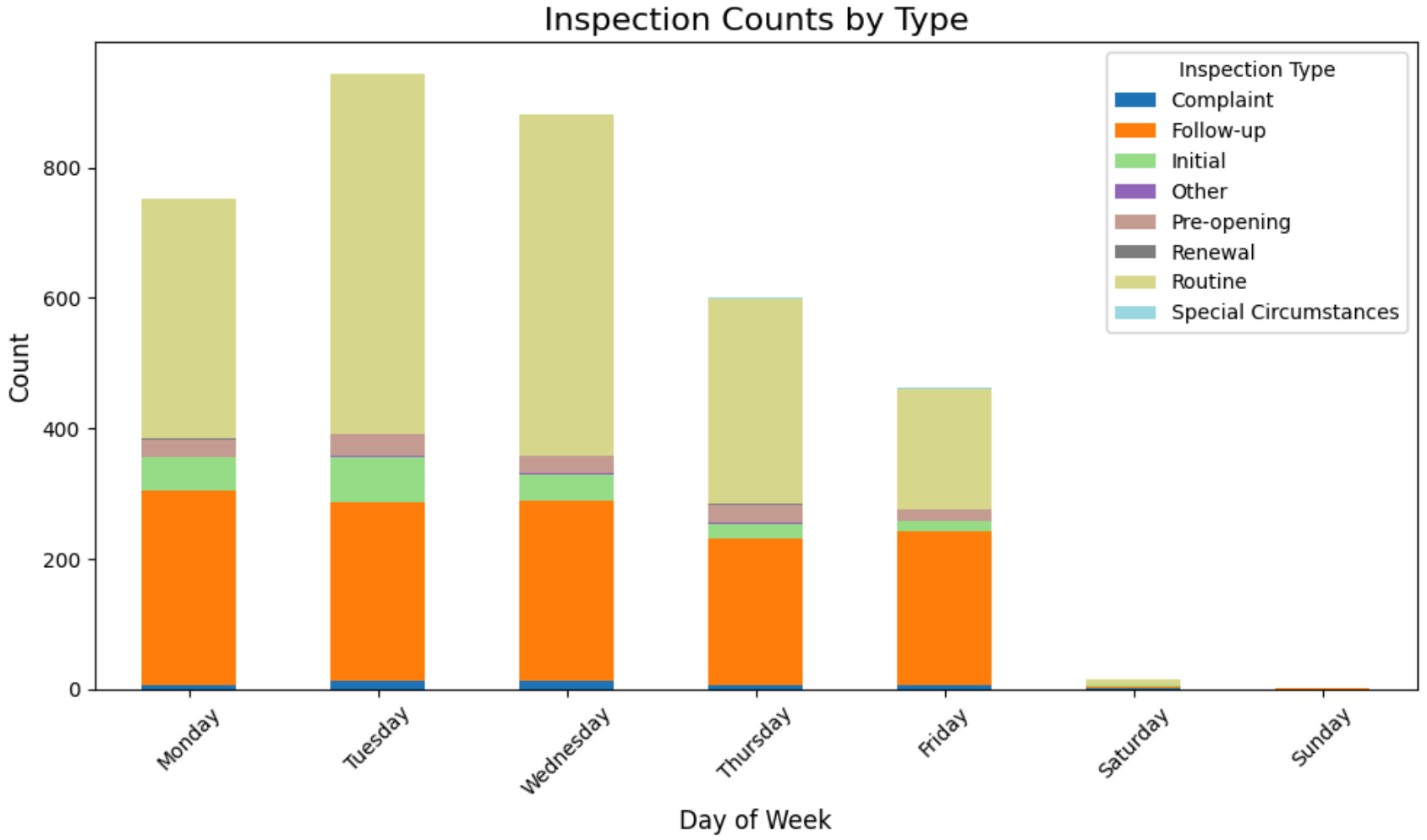
In [21]:
```
# Group by 'Day of Week' and 'Inspection Type' to calculate counts
aggregated_df = df.groupby(['Day of Week', 'Inspection Type']).size().reset_index(name='Count')

pivot_df = aggregated_df.pivot(index='Day of Week', columns='Inspection Type', values='Count').fillna(0)
```

```
# Step 3: Plot a stacked column chart
ax = pivot_df.plot(kind='bar', stacked=True, figsize=(10, 6), colormap='tab20')

# Step 4: Add labels and title
plt.title("Inspection Counts by Type", fontsize=16)
plt.xlabel("Day of Week", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Inspection Type', fontsize=10)
plt.tight_layout()

# Step 5: Show the chart
plt.show()
```

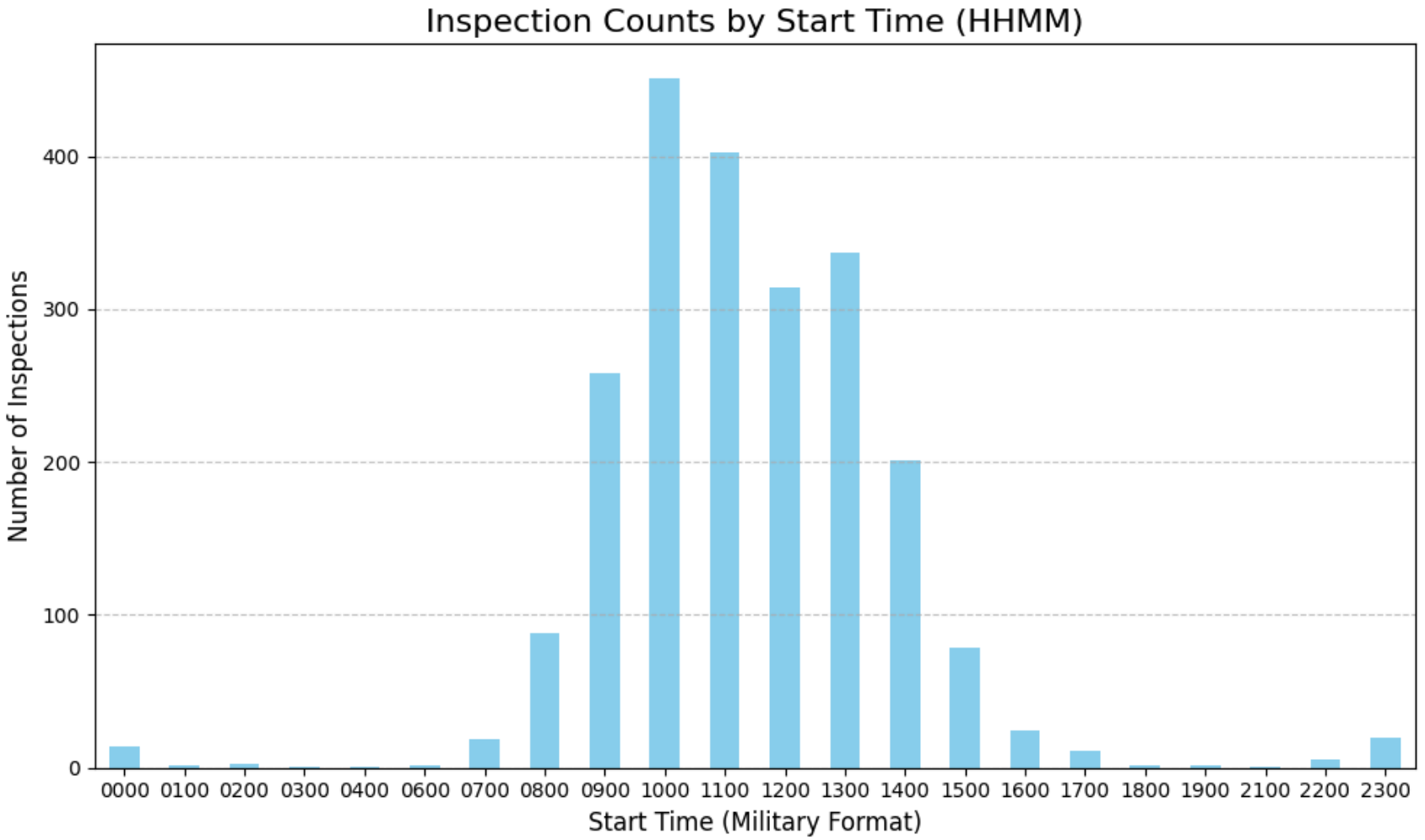


In [22]:
```
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df['Day of Week'] = pd.Categorical(df['Day of Week'], categories=day_order, ordered=True)

# Group by 'Day of Week' and 'Inspection Type' to calculate counts
aggregated_df = df.groupby(['Day of Week', 'Inspection Type'], observed=False).size().reset_index(name='Cou

# Step 3: Pivot the data
pivot_df = aggregated_df.pivot(index='Day of Week', columns='Inspection Type', values='Count').fillna(0)

# Step 4: Plot a stacked column chart
ax = pivot_df.plot(kind='bar', stacked=True, figsize=(10, 6), colormap='tab20')
```

```
# Step 5: Add labels and title
plt.title("Inspection Counts by Type", fontsize=16)
plt.xlabel("Day of Week", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Inspection Type', fontsize=10)
plt.tight_layout()

# Step 6: Show the chart
plt.show()
```

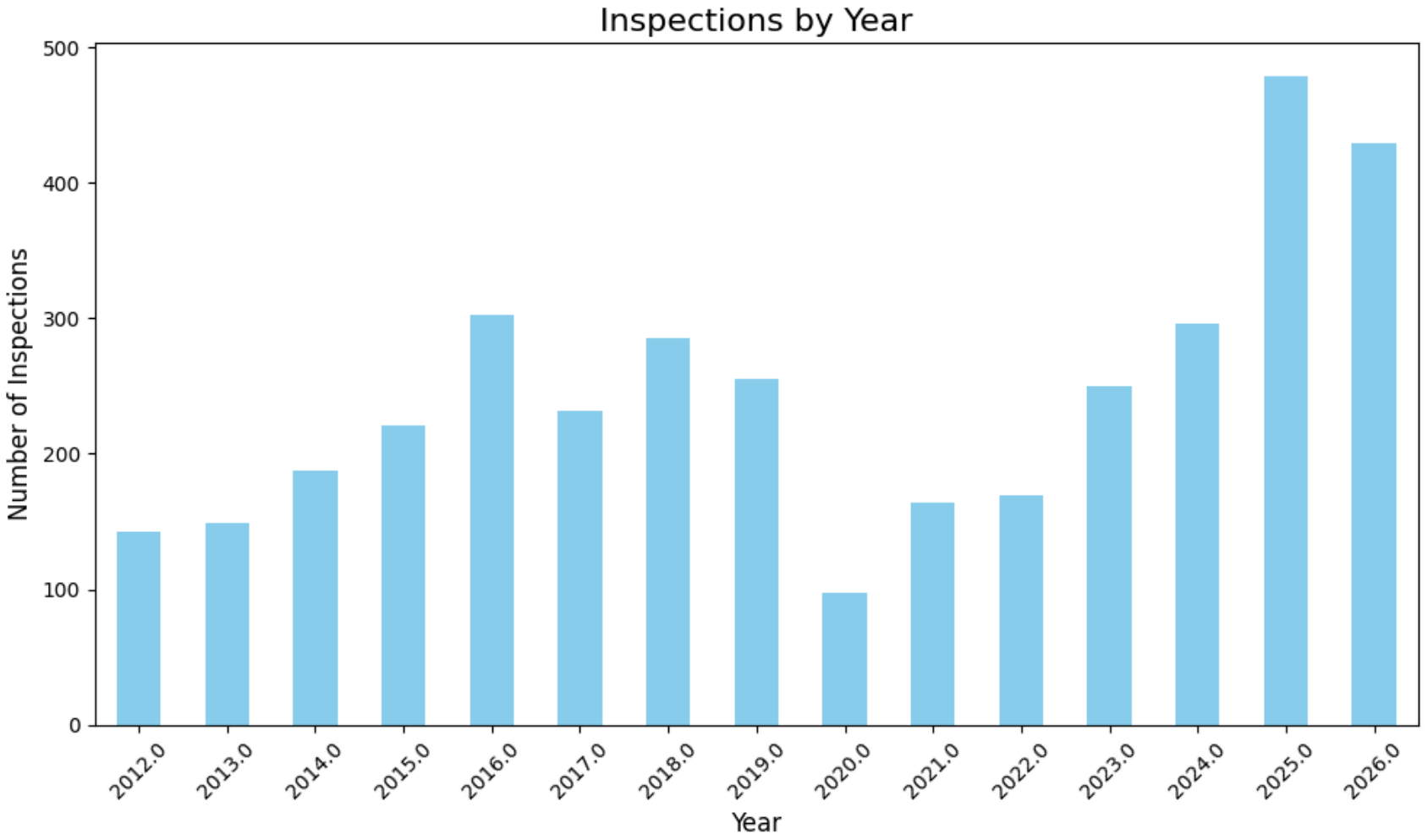


In [27]:
```
# Extracts the .hour property from Python time objects directly
df['Start Hour'] = df['Start Time'].apply(lambda x: x.hour if pd.notnull(x) else None)

# Step 3: Count the number of inspections for each hour
start_time_counts = df['Start Hour'].value_counts().sort_index()

# Step 4: Plot the bar chart
plt.figure(figsize=(10, 6))
ax = start_time_counts.plot(kind='bar', color='skyblue')

# Step 5: Customize x-axis to show HHMM (military time)
ax.set_xticks(range(len(start_time_counts.index)))
ax.set_xticklabels([f"{int(hour):02d}00" for hour in start_time_counts.index], rotation=0)

# Step 6: Add labels and title
plt.title("Inspection Counts by Start Time (HHMM)", fontsize=16)
plt.xlabel("Start Time (Military Format)", fontsize=12)
plt.ylabel("Number of Inspections", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Step 7: Show the chart
plt.tight_layout()
plt.show()
```



In [28]:
```
# Step 2: Extract the year from the 'Inspection Date' column
df['Year'] = df['Inspection Date'].dt.year

# Step 3: Group by the year and calculate the count of inspections
inspection_by_year = df.groupby('Year').size()

# Step 4: Plot the data
inspection_by_year.plot(kind='bar', figsize=(10, 6), color='skyblue')
plt.title("Inspections by Year", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Number of Inspections", fontsize=12)
```

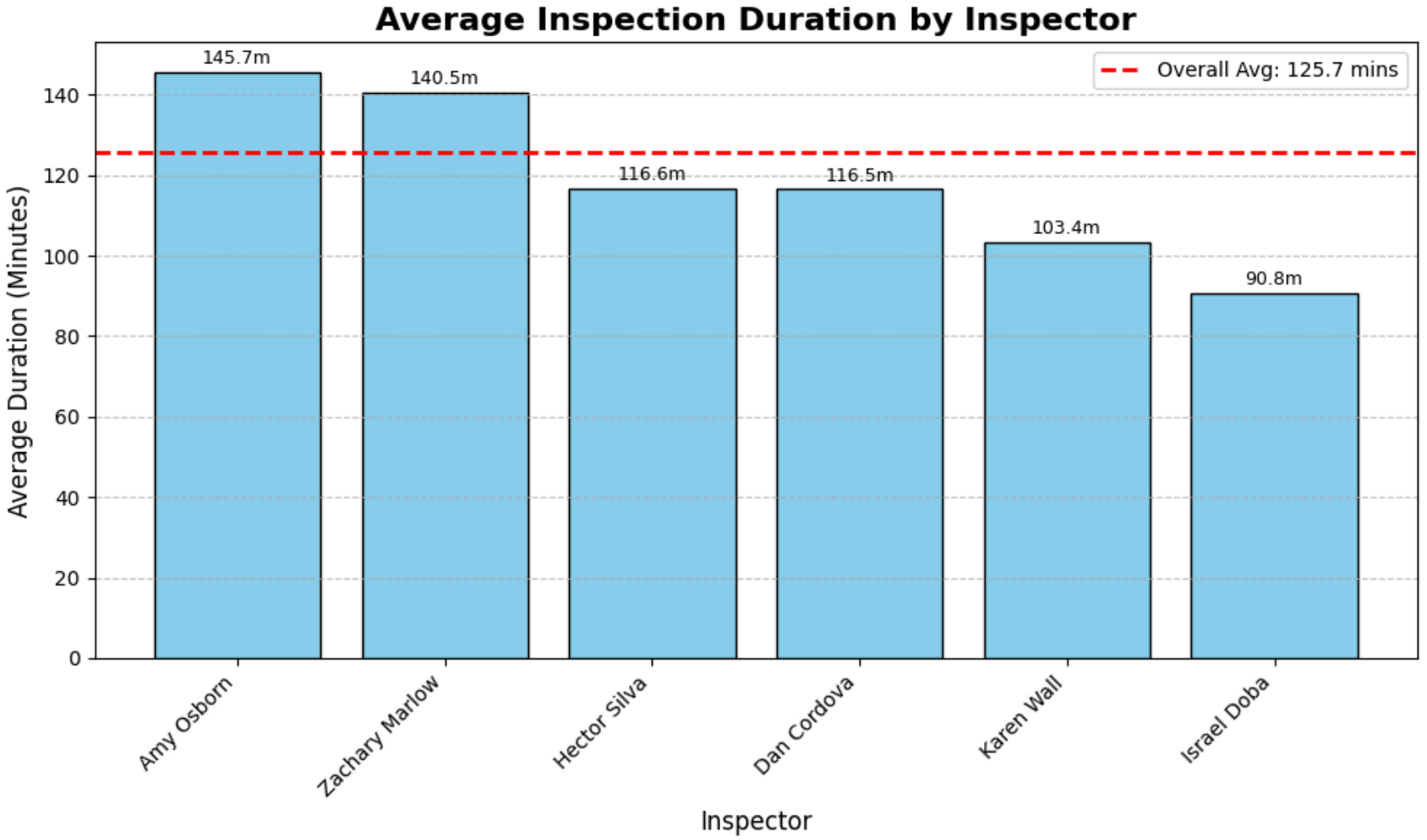
```
plt.xticks(rotation=45)
plt.tight_layout()

# Step 5: Show the chart
plt.show()
```



In [29]:
```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Combine 'Inspection Date' with times to create complete datetime timestamps
start_dt = pd.to_datetime(
```

```
    df['Inspection Date'].astype(str) + ' ' + df['Start Time'].astype(str),
    errors='coerce'
)
end_dt = pd.to_datetime(
    df['Inspection Date'].astype(str) + ' ' + df['End Time'].astype(str),
    errors='coerce'
)

# 2. Calculate duration in minutes
df['Inspection Duration'] = (end_dt - start_dt).dt.total_seconds() / 60

# Adjust for overnight inspections (if end time is past midnight/smaller than start time)
df.loc[df['Inspection Duration'] < 0, 'Inspection Duration'] += 24 * 60

# Filter out invalid or zero/negative durations
valid_df = df[df['Inspection Duration'] > 0].copy()

if valid_df.empty:
    print("Could not calculate durations. Please check 'Start Time' and 'End Time' data.")
else:
    # 3. Calculate average inspection time by inspector
    avg_by_inspector = valid_df.groupby('Inspector')['Inspection Duration'].mean().sort_values(ascending=Fa

    # 4. Calculate overall average
    overall_avg = valid_df['Inspection Duration'].mean()

    # 5. Plot results using Matplotlib directly
    plt.figure(figsize=(10, 6))
    bars = plt.bar(avg_by_inspector.index, avg_by_inspector.values, color='skyblue', edgecolor='black')

    # 6. Add horizontal benchmark line for overall average
    plt.axhline(
        y=overall_avg,
        color='red',
        linestyle='--',
        linewidth=2,
        label=f'Overall Avg: {overall_avg:.1f} mins'
    )

    # 7. Annotate bar values on top of each bar
    for bar in bars:
        yval = bar.get_height()
```

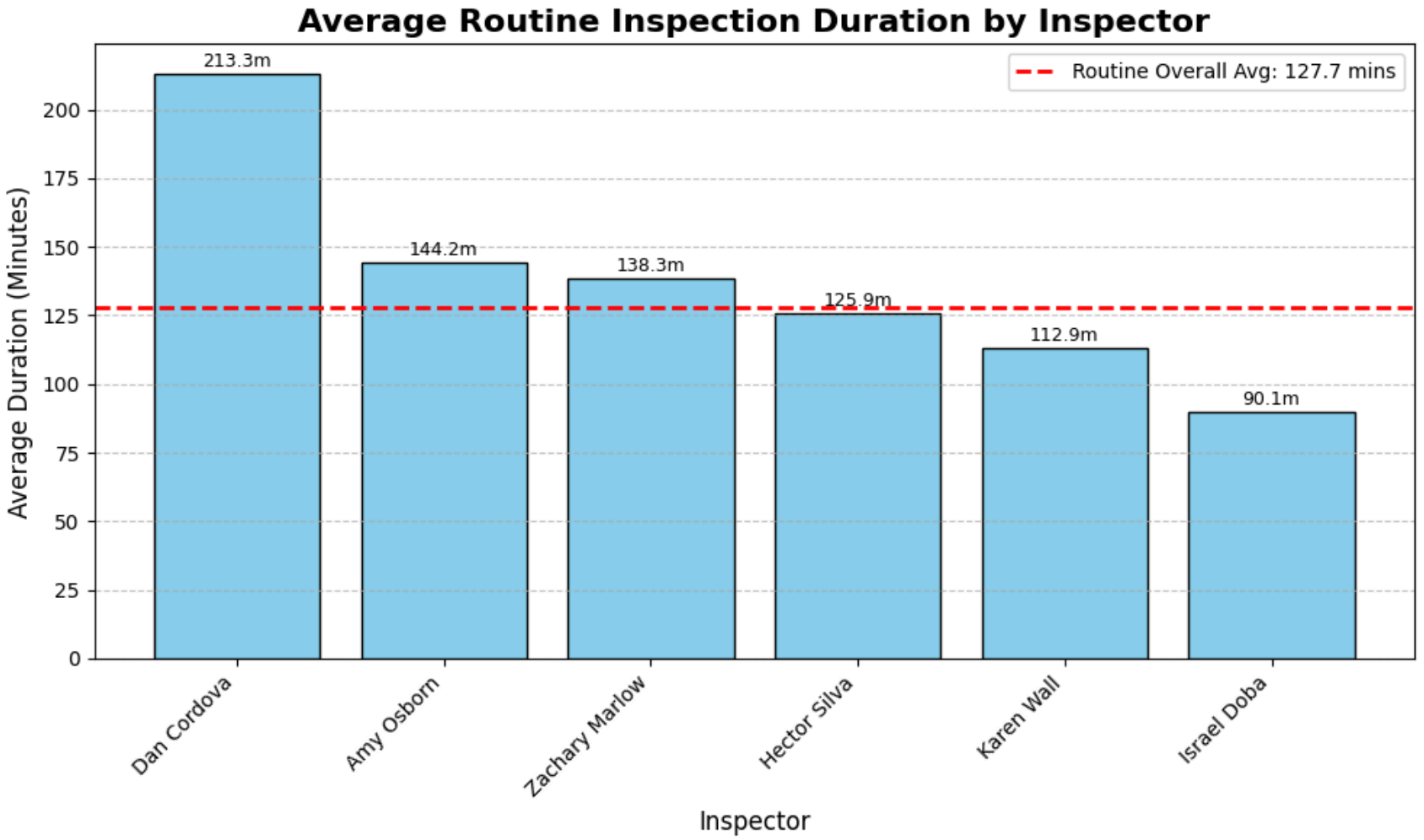
```
        plt.text(
            bar.get_x() + bar.get_width() / 2.0,
            yval + 1,
            f'{yval:.1f}m',
            ha='center',
            va='bottom',
            fontsize=9
        )

    # 8. Formatting
    plt.title("Average Inspection Duration by Inspector", fontsize=16, fontweight='bold')
    plt.xlabel("Inspector", fontsize=12)
    plt.ylabel("Average Duration (Minutes)", fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.legend(loc='upper right')

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



In [30]:
```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Ensure full datetime objects for duration calculation
start_dt = pd.to_datetime(
    df['Inspection Date'].astype(str) + ' ' + df['Start Time'].astype(str),
    errors='coerce'
)
end_dt = pd.to_datetime(
    df['Inspection Date'].astype(str) + ' ' + df['End Time'].astype(str),
    errors='coerce'
```

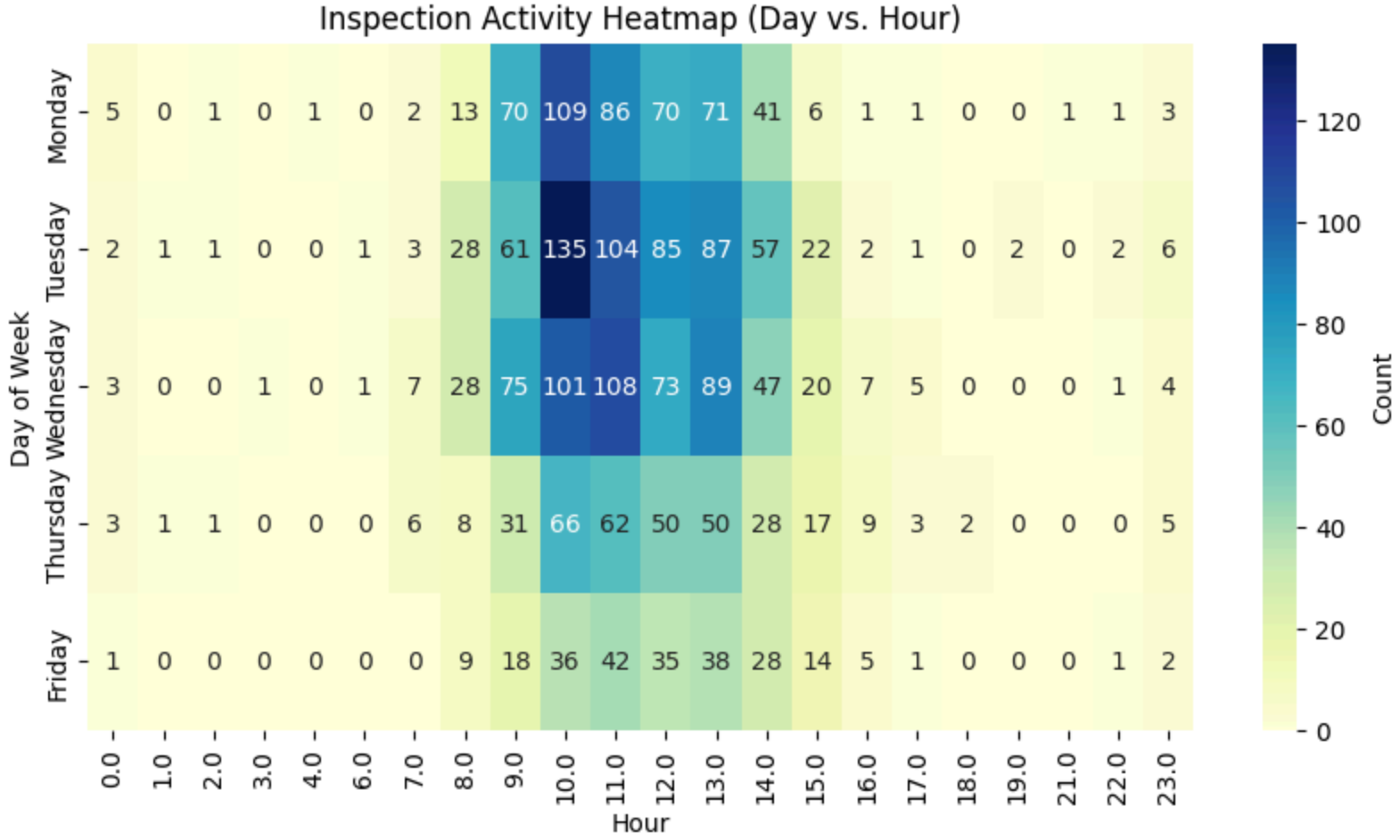
```
)

# 2. Calculate duration in minutes
df['Inspection Duration'] = (end_dt - start_dt).dt.total_seconds() / 60
df.loc[df['Inspection Duration'] < 0, 'Inspection Duration'] += 24 * 60  # Overnight shift fix

# 3. Filter strictly for 'Routine' inspections with valid durations
routine_df = df[
    (df['Inspection Type'] == 'Routine') &
    (df['Inspection Duration'] > 0)
].copy()

if routine_df.empty:
    print("No 'Routine' inspections found with valid start/end times.")
else:
    # 4. Group by Inspector for routine inspections only
    routine_avg_by_inspector = (
        routine_df.groupby('Inspector')['Inspection Duration']
        .mean()
        .sort_values(ascending=False)
    )

    # 5. Overall average for routine inspections
    overall_routine_avg = routine_df['Inspection Duration'].mean()

    # 6. Plotting
    plt.figure(figsize=(10, 6))
    bars = plt.bar(
        routine_avg_by_inspector.index,
        routine_avg_by_inspector.values,
        color='skyblue',
        edgecolor='black'
    )

    # Horizontal benchmark line for Routine overall average
    plt.axhline(
        y=overall_routine_avg,
        color='red',
        linestyle='--',
        linewidth=2,
        label=f'Routine Overall Avg: {overall_routine_avg:.1f} mins'
    )
```

```
    # Add numeric labels above each bar
    for bar in bars:
        yval = bar.get_height()
        plt.text(
            bar.get_x() + bar.get_width() / 2.0,
            yval + 1,
            f'{yval:.1f}m',
            ha='center',
            va='bottom',
            fontsize=9
        )

    # Formatting
    plt.title("Average Routine Inspection Duration by Inspector", fontsize=16, fontweight='bold')
    plt.xlabel("Inspector", fontsize=12)
    plt.ylabel("Average Duration (Minutes)", fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.legend(loc='upper right')

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



In [31]:
```
# Quick Example: Code snippet for the Workload Heatmap
import seaborn as sns
import matplotlib.pyplot as plt

# Prepare Day of Week and Hour
df['Day of Week'] = df['Inspection Date'].dt.day_name()
df['Hour'] = pd.to_datetime(df['Start Time'].astype(str), format='%H:%M:%S', errors='coerce').dt.hour

# Pivot table for heatmap
heatmap_data = df.pivot_table(index='Day of Week', columns='Hour', values='Name', aggfunc='count').fillna(
days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

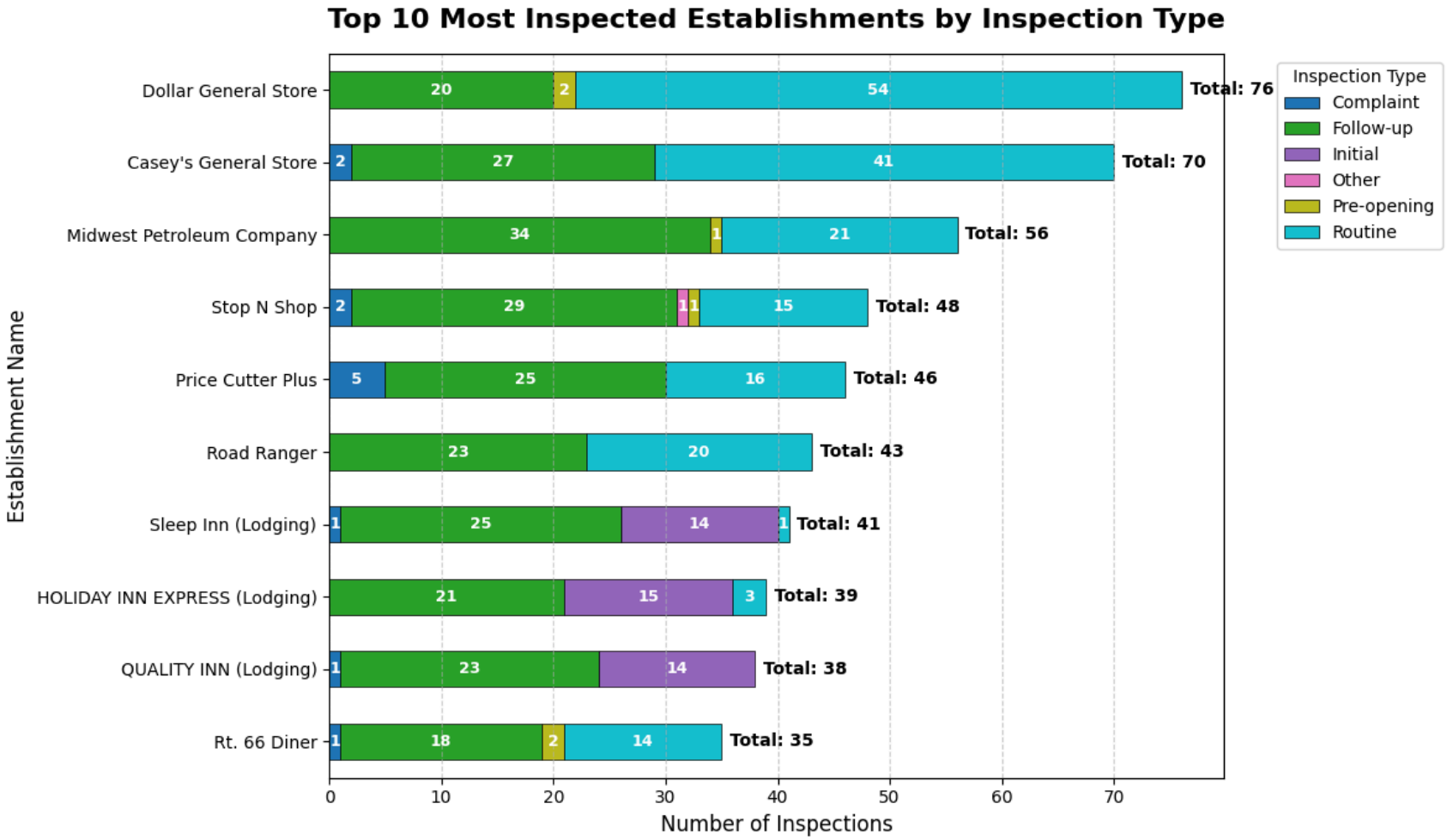
```
heatmap_data = heatmap_data.reindex(days_order)

plt.figure(figsize=(10, 5))
sns.heatmap(heatmap_data, cmap='YlGnBu', annot=True, fmt='g', cbar_kws={'label': 'Count'})
plt.title('Inspection Activity Heatmap (Day vs. Hour)')
plt.show()
```



In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Identify the Top 10 most inspected establishment names
top_10_names = df['Name'].value_counts().head(10).index
```

```
# 2. Filter the dataframe to only include these top 10 establishments
top_10_df = df[df['Name'].isin(top_10_names)].copy()

# 3. Create a pivot table: Rows = Establishment Name, Columns = Inspection Type
pivot_df = top_10_df.pivot_table(
    index='Name',
    columns='Inspection Type',
    aggfunc='size',
    fill_value=0
)

# 4. Sort by total overall inspections so the most inspected is at the top
pivot_df['Total'] = pivot_df.sum(axis=1)
pivot_df = pivot_df.sort_values(by='Total', ascending=True)  # Ascending for horizontal bar
pivot_df = pivot_df.drop(columns=['Total'])  # Drop total column before plotting

# 5. Plot a horizontal stacked bar chart
fig, ax = plt.subplots(figsize=(12, 7))

# Choose a distinct color palette
pivot_df.plot(
    kind='barh',
    stacked=True,
    ax=ax,
    colormap='tab10',
    edgecolor='black',
    linewidth=0.5
)

# 6. Add value labels inside/on top of bars for clarity
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    if width > 0:  # Only label segments with actual values
        x, y = p.get_xy()
        ax.text(
            x + width / 2,
            y + height / 2,
            f'{int(width)}',
            ha='center',
            va='center',
            color='white',
```

```
            fontweight='bold',
            fontsize=9
        )

# 7. Add totals at the end of each bar
totals = pivot_df.sum(axis=1)
for idx, (name, total) in enumerate(totals.items()):
    ax.text(
        total + 0.2,
        idx,
        f' Total: {int(total)}',
        va='center',
        fontweight='bold',
        fontsize=10
    )

# 8. Chart Formatting
plt.title("Top 10 Most Inspected Establishments by Inspection Type", fontsize=16, fontweight='bold', pad=15
plt.xlabel("Number of Inspections", fontsize=12)
plt.ylabel("Establishment Name", fontsize=12)
plt.legend(title="Inspection Type", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(axis='x', linestyle='--', alpha=0.6)

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


Top 10 Most Inspected Establishments by Inspection Type
Establishment Name
Number of Inspections
Inspection Type
Complaint
Follow-up
Initial
Other
Pre-opening
Routine
Dollar General Store
Casey's General Store
Midwest Petroleum Company
Stop N Shop
Price Cutter Plus
Road Ranger
Sleep Inn (Lodging)
HOLIDAY INN EXPRESS (Lodging)
QUALITY INN (Lodging)
Rt. 66 Diner
Total: 76
Total: 70
Total: 56
Total: 48
Total: 46
Total: 43
Total: 41
Total: 39
Total: 38
Total: 35
0
10
20
30
40
50
60
70