Converting HTML Pages to Structured Tabular Data

In [3]:
```
from bs4 import BeautifulSoup
import pandas as pd

# Step 1: Load the HTML file
with open("female-detainee-cases.html", "r", encoding="utf-8") as f:
    soup = BeautifulSoup(f, "html.parser")

# Step 2: Find all <a> tags (which include the case links)
links = soup.find_all("a")

# Step 3: Extract case number and description from the text
data = []

for link in links:
    text = link.text.strip()
    if text.lower().startswith("case"):
        parts = text.split(" ", 2)
        if len(parts) == 3:
            _, case_number, description = parts
            data.append({
                "case_number": case_number,
                "description": description
            })

# Step 4: Create DataFrame
df = pd.DataFrame(data)
df.head()
```

Out[3]:

	case_number	description
0	2657	Moy Chin See his wife
1	2917	Lee Kin Sai alias Lee Wah Chung
2	2950	Tie Yimm a woman
3	3068	Lin Kum daughter, Wye See mother
4	3100	Tarm How Yen wife

In [4]:
```
# Step 1: Load the HTML file
with open("female-detainee-cases.html", "r", encoding="utf-8") as f:
    soup = BeautifulSoup(f, "html.parser")

# Step 2: Find all <a> tags (the case links)
links = soup.find_all("a")

# Step 3: Extract case number and description
data = []

for link in links:
    text = link.text.strip()
```

```
    if text.lower().startswith("case"):
        parts = text.split(" ", 2)
        if len(parts) == 3:
            _, case_number, description = parts
            data.append({
                "case_number": case_number,
                "description": description,
                "gender": "Female",
                "detainee_type": "Civil / Habeas Corpus",
                "court": "US District Court",
                "location": "Northern District of California, San Francisco",
                "case_year_range": "1882–1892"
            })

# Step 4: Create DataFrame
df = pd.DataFrame(data)

# Step 5: Display the first few rows
df.head(10)
```

Out[4]:

	case_number	description	gender	detainee_type	court	location	case_year_range
0	2657	Moy Chin See his wife	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
1	2917	Lee Kin Sai alias Lee Wah Chung	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
2	2950	Tie Yimm a woman	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
3	3068	Lin Kum daughter, Wye See mother	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
4	3100	Tarm How Yen wife	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
5	3308	Yung Ah Chung woman	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
6	3549	Mrs. Fong Ah Chung	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
7	3644	Mrs. Ching Din	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892
8	3745	Mrs. Lee nee Chun Ah On	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892

	case_number	description	gender	detainee_type	court	location	case_year_range
9	3763	Mrs. Leong nee Lee Ah Fung	Female	Civil / Habeas Corpus	US District Court	Northern District of California, San Francisco	1882–1892

Cleaning and Processing Data

In [6]:
```
# Load the CSV file
df = pd.read_csv("female_detainee_cases.csv")

# Print column names to confirm
print(df.columns)
```

```
Index(['Case Number', 'Name / Description', 'File Name'], dtype='object')
```

In [7]:
```
df.columns = df.columns.str.strip()
```

In [8]:
```
# Clean column names just in case
df.columns = df.columns.str.strip()

# Fix file names based on case number
def fix_file_name(row):
    case_number = str(row['Case Number'])
    if case_number not in str(row['File Name']):
        return f"{case_number}.html"
    return row['File Name']

df['File Name'] = df.apply(fix_file_name, axis=1)
print(df)
```

```
     Case Number                Name / Description   File Name
0           2657              Moy Chin See his wife   2657.html
1           2917   Lee Kin Sai alias Lee Wah Chung   2917.html
2           2950                  Tie Yimm a woman   2950.html
3           3068  Lin Kum daughter, Wye See mother   3068.html
4           3100                 Tarm How Yen wife   3100.html
..           ...                               ...         ...
133        10116                    Chin Chon Loy  10116.html
134        10144                    Cha Sing Kwai  10144.html
135        10145                      Cha Tai Kim  10145.html
136        10175                  Young Choy Ling  10175.html
137        10209                          Lum Toy  10209.html

[138 rows x 3 columns]
```

In [9]:
```
# Clean column names first (in case they have extra spaces)
df.columns = df.columns.str.strip()

# List of terms to remove
remove_terms = ['Mrs.', 'mrs.', 'daughter', 'wife', 'sisters', 'Sisters','mother','

# Function to clean the name/description
```

```
def clean_name(desc):
    for word in remove_terms:
        desc = desc.replace(word, "")
    return desc.strip()

# Apply to 'Name / Description' column
df['Name / Description'] = df['Name / Description'].apply(clean_name)
print(df)
```

```
    Case Number               Name / Description   File Name
0          2657                 Moy Chin See his   2657.html
1          2917  Lee Kin Sai alias Lee Wah Chung   2917.html
2          2950                         Tie Yimm   2950.html
3          3068                Lin Kum , Wye See   3068.html
4          3100                     Tarm How Yen   3100.html
..          ...                              ...         ...
133       10116                   Chin Chon Loy  10116.html
134       10144                   Cha Sing Kwai  10144.html
135       10145                     Cha Tai Kim  10145.html
136       10175                  Young Choy Ling  10175.html
137       10209                         Lum Toy  10209.html

[138 rows x 3 columns]
```

Data Exploration

In [11]:
```
# Split names into individual words and count the most common ones
from collections import Counter

all_names = " ".join(df["Name / Description"]).split()
common_names = Counter(all_names).most_common(10)

# Turn into a DataFrame to view nicely
common_names_df = pd.DataFrame(common_names, columns=["Name Part", "Count"])
print(common_names_df)
```

```
  Name Part  Count
0        Ah     50
1      Wong     17
2        Ho     16
3       Lee     14
4     Leong     11
5      Choy     10
6      Fong      8
7      Chun      8
8     Quock      8
9       See      7
```

In [12]:
```
# Extract last word as "surname"
df['Surname'] = df['Name / Description'].apply(lambda x: x.split()[-1])

# Count the most common surnames
top_surnames = df['Surname'].value_counts().head(10)
print(top_surnames)
```

```
Surname
Ho       12
See       5
Far       5
Choy      5
Fong      4
Gim       3
Gun       3
Chung     3
Kim       3
Hee       3
Name: count, dtype: int64
```

Data Analysis

In [14]:
```python
import plotly.express as px

# Load the CSV file
df = pd.read_csv("female_detainee_cases.csv")  # adjust path if needed
df.columns = df.columns.str.strip()

# Fix file names if needed
def fix_file_name(row):
    case_number = str(row['Case Number'])
    if case_number not in str(row['File Name']):
        return f"{case_number}.html"
    return row['File Name']
df['File Name'] = df.apply(fix_file_name, axis=1)

# Clean unwanted labels from names
remove_terms = ['Mrs.', 'mrs.', 'daughter', 'wife', 'sisters', 'Sisters']
def clean_name(desc):
    for word in remove_terms:
        desc = desc.replace(word, "")
    return desc.strip()
df['Name / Description'] = df['Name / Description'].apply(clean_name)

# Extract surname (last word in name)
df['Surname'] = df['Name / Description'].apply(lambda x: x.split()[-1])

# Convert case number to numeric
df['Case Number'] = pd.to_numeric(df['Case Number'], errors='coerce')
```

In [15]:
```python
# Count surnames
surname_counts = df['Surname'].value_counts()

# Find surnames with more than 2 entries
common_surnames = surname_counts[surname_counts > 2]
print(common_surnames)
```

```
Surname
Ho       12
Far       5
Choy      5
See       4
Fong      4
Hee       3
Chung     3
Gun       3
You       3
Fung      3
Yee       3
Gim       3
Kim       3
Name: count, dtype: int64
```

In [16]:
```python
# Count and filter surnames with more than 2 entries
surname_counts = df['Surname'].value_counts()
common_surnames = surname_counts[surname_counts > 2].reset_index()
common_surnames.columns = ['Surname', 'Number of Cases']

# Plot with Plotly
fig_surnames = px.pie(
    common_surnames,
    names='Surname',
    values='Number of Cases',
    title='Distribution of Most Frequent Surnames (Appearing More Than Twice)'
)
fig_surnames.show()
```

In [17]:
```
from collections import Counter
import plotly.express as px

# Combine all names into a single string, then split into words
all_words = " ".join(df["Name / Description"]).split()

# Count the top 15 most frequent name parts
common_name_parts = Counter(all_words).most_common(15)

# Convert to a DataFrame
common_name_parts_df = pd.DataFrame(common_name_parts, columns=["Name Part", "Count

# Create horizontal bar chart with Plotly
fig_common_names = px.bar(
    common_name_parts_df.sort_values("Count", ascending=True),
    x="Count",
    y="Name Part",
    orientation="h",
    title="Most Common Name Parts Among Female Detainees",
    labels={"Count": "Frequency", "Name Part": "Name Part"}
)
fig_common_names.show()
```

Reflection

Utilizing AI to assist with this coding has several effective benefits. The first thing that worked well using AI is that I was able to communicate with the AI to better understand the code that it provided. If there was code that didn't make sense to me or I didn't know what it was doing, I could have the AI break it down for me. Unlike other code websites, such as Walsh's code, I couldn't get immediate clarification. If there were any syntax errors, I could use the AI to help correct them, and it would make corrections. Since this assignment was so free-range, a thing I struggle with at times, especially when I have trouble with what we're working with, like code, AI has helped me get inspiration. When I was exploring the data, I didn't know where to start, so I asked the AI for some inspiration and proposals to consider, which helped me gain a sense of direction. Of course, what it did best was produce new code for what I needed, which was suitable for this kind of assignment because you can do anything. Although there are many things coding did well, there were still frustrations.

There were also numerous challenges and frustrations with using AI that I had to overcome. The biggest is trying not to fall into vibe coding. I wanted to ensure I understood the code it provided, but I could have easily gone with it without doing so. One example is that it actually created graphs on JupiterLabs without importing Plotly, which surprised me.

However, since I'm more familiar with Plotly, that's what I went with. This presented another strange and frustrating problem I kept running into, which was the fact that the AI kept doing its own thing. It constantly kept trying to import all these different tools that I had never heard of. In the end, I did use one called Counter, which I never used before, but it pales in comparison to how much it wanted to use that I didn't end up importing. My biggest frustration, though, was the times when the AI and I were not on the same page, or in other words, miscommunication. If the AI didn't understand my instructions or I didn't understand the code, it felt like pulling teeth trying to get on the same page, which left me incredibly frustrated. There were times when I encountered errors in the code due to this. To be honest, I only really overcame this by keeping things simple and abandoning specific topics. However, I believe that this issue is specific to the assignment. The times I use AI, I often encounter problems with using it effectively on particular topics. I think our readings and understanding of code helped me better understand how to communicate with AI. Even with this assignment, it could have been worse. The lesson I want to take away from this assignment is to improve my communication with AI to produce more effective and comprehensible code. I also learned how frustrating AI can be, as well as to understand the associated risks. Also, knowing how tempting vibe coding can be, which can make it even more confusing. I want to know that AI is a useful tool for coding, but it must be used correctly, as it is a very powerful tool when utilized properly.

In [ ]: