import pandas as pd ## 1. “This submission is my work alone and complies with the 30538 integrity policy.” # Add your initials to indicate your agreement: **JL_ *# # 2. “I have uploaded the names of anyone I worked with on the problem set here” none*** (1 point) # # 3. Late coins used this pset: ***1*** Late coins left after submission: ***idk*** # # 4. Knit your `ps1.qmd` to an HTML file to make `ps1.html`. # # 5. Save your HTML file as a PDF and upload your `ps1.pdf` to Gradescope. # - The PDF should not exceed 25 pages. Use `head()` and resize figures when appropriate. # # 6. Push `ps1.qmd` and `ps1.html` to your GitHub repo. It is fine to use GitHub Desktop. # # 7. Submit `ps1.pdf` via Gradescope (4 points). # # 8. Tag your submission in Gradescope.

file_path = ‘/Users/doudouliu/Desktop/Snake 2/ppha30538_fall2024/problem_sets/ps1/data/parking_tickets_one_percent.csv’

#Q1.1 import pandas as pd df_1 = pd.read_csv(file_path) df_1.head() import time as t begin = t.time()

#record the ending time end = t.time()

#get the difference time_used = end - begin print (time_used)

#time used is about 1.08 seconds +/- 0.001

Assert statement

assert df_1.shape[0] == 287458, “Row count is off”

#1.2 import os data_size = os.path.getsize(file_path)

#change the unit data_size_mb = data_size / (1024 * 1024) print (data_size_mb) #The size in mb is 79.78

#1.3 #eye ball the columns, are they sorted df_1.head() #issued date is sorted

Check if columns with numeric values are sorted

def check_sorted_numeric_columns(df): numeric_cols = df.select_dtypes(include=[‘number’]).columns # Select only numeric columns sorted_columns = {}

```
for col in numeric_cols:
    sorted_columns[col] = df[col].is_monotonic_increasing  # Check if the column is sorte

return sorted_columns
```

Run the function

sorted_col = check_sorted_numeric_columns(df_1)

Print the result

for col, is_sorted in sorted_col.items(): print(f"Is '{col}' sorted? {is_sorted}") #just making sure df_1.head()

```
1.5020370483398438e-05
79.77995681762695
Is 'Unnamed: 0' sorted? True
Is 'ticket_number' sorted? False
Is 'unit' sorted? False
Is 'fine_level1_amount' sorted? False
Is 'fine_level2_amount' sorted? False
Is 'current_amount_due' sorted? False
Is 'total_payments' sorted? False
Is 'notice_number' sorted? False
Number of tickets issued in 2017 (sample): 22364
Number of tickets issued in 2017 (population): 2236400
```

2.1

Convert ‘issue_date’ to datetime format

df_1[‘issue_date’] = pd.to_datetime(df_1[‘issue_date’])

distill year

df_1[‘year’] = df_1[‘issue_date’].dt.year

subset tickets from 2017

ticket_17 = df_1[df_1[‘year’] == 2017]

Count the number of tickets

ticket_17_count = ticket_17.shape[0] print(f”Number of tickets issued in 2017 (sample): {ticket_17_count}“) # The number of tickets issued in 2017 (sample) is 22,364 # Since this is a 1% sample of the population, we scale the number up by 100 print(f”Number of tickets issued in 2017 (population): {ticket_17_count * 100}“) # The number of tickets issued in 2017 (population) is estimated to be 2,236,400 # According to the ProPublica web, the total number of tickets issued in fiscal year 2017 was over 3 million, # which is about 50% more than our estimation based on the sample. # This suggests that our sample might not be representative and could be downwardly biased due to randomization failures.

#2.2 import matplotlib.pyplot as plt

v_cards = df_1[‘violation_description’].value_counts()

choose the big 20

plot them

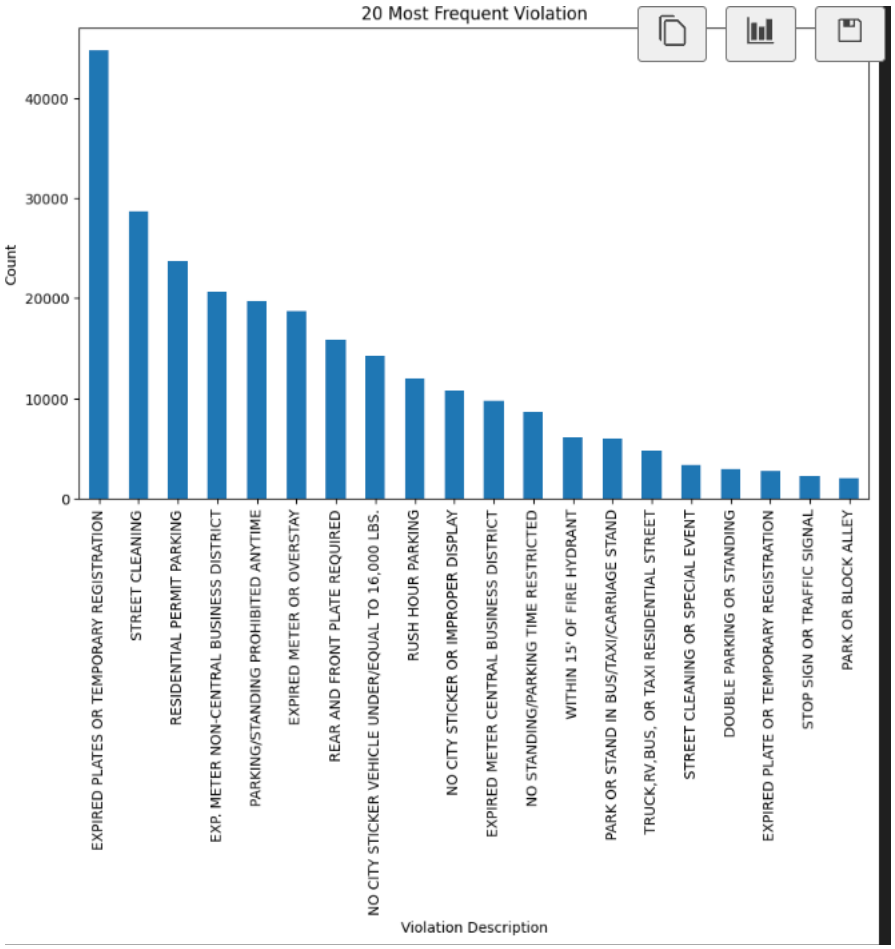
plt.figure(figsize=(10, 6))
big_20.plot(kind='bar')

Add titles and labels

plt.title('20 Most Frequent Violation') plt.ylabel('Count') plt.xlabel('Violation Description')

Show the plot

plt.show()



Format the markdown table

header = “| Name | Type |” separator = “|———————|———————|” rows = [f”| {var:<25} | {dtype:<16} |” for var, dtype in zip(data[‘Variable Name’], data[‘Data Type’])]

Combine the header, separator, and rows into a list

markdown_table = “”.join([header, separator] + rows)

Print the markdown table

print(markdown_table)

Explanation of the columns

explanation = “”” ## Explanation of Multiple Types:

`zipcode`: - While denoted in numbers, `zipcode` can also be treated as a nominal variable because it indicates regional information and doesn’t imply anything in itself

”””

Print the explanation

print(explanation)

Name	Type
ticket_number	Quantitative (Q)
issue_date	Temporal (T)
violation_location	Nominal (N)
license_plate_number	Nominal (N)
license_plate_state	Nominal (N)
license_plate_type	Nominal (N)
zipcode	Quantitative (Q)
violation_code	Nominal (N)
violation_description	Nominal (N)
fine_level2_amount	Quantitative (Q)
current_amount_due	Quantitative (Q)
total_payments	Quantitative (Q)
ticket_queue	Nominal (N)
ticket_queue_date	Temporal (T)
notice_level	Ordinal (O)
hearing_disposition	Nominal (N)
notice_number	Quantitative (Q)
officer	Nominal (N)
address	Nominal (N)

Explanation of Multiple Types:

`zipcode`:
- While denoted in numbers, `zipcode` can also be treated as a nominal variable because it indicates regional information and doesn't imply anything in itself

Plot the data using a heatmap

import matplotlib.pyplot as plt import seaborn as sns

plt.figure(figsize=(12, 6)) # Set the figure size for the plot sns.heatmap(pivot_table_36.T, cmap='coolwarm', annot=False) # Create a heatmap to show the number of tickets over time

Add element

plt.title('Number of Tickets Issued Over Time for Top 5 Violation Types') # Title of the plot plt.ylabel('Violation Type') # Label for y-axis plt.xlabel('Year-Month') # Label for x-axis

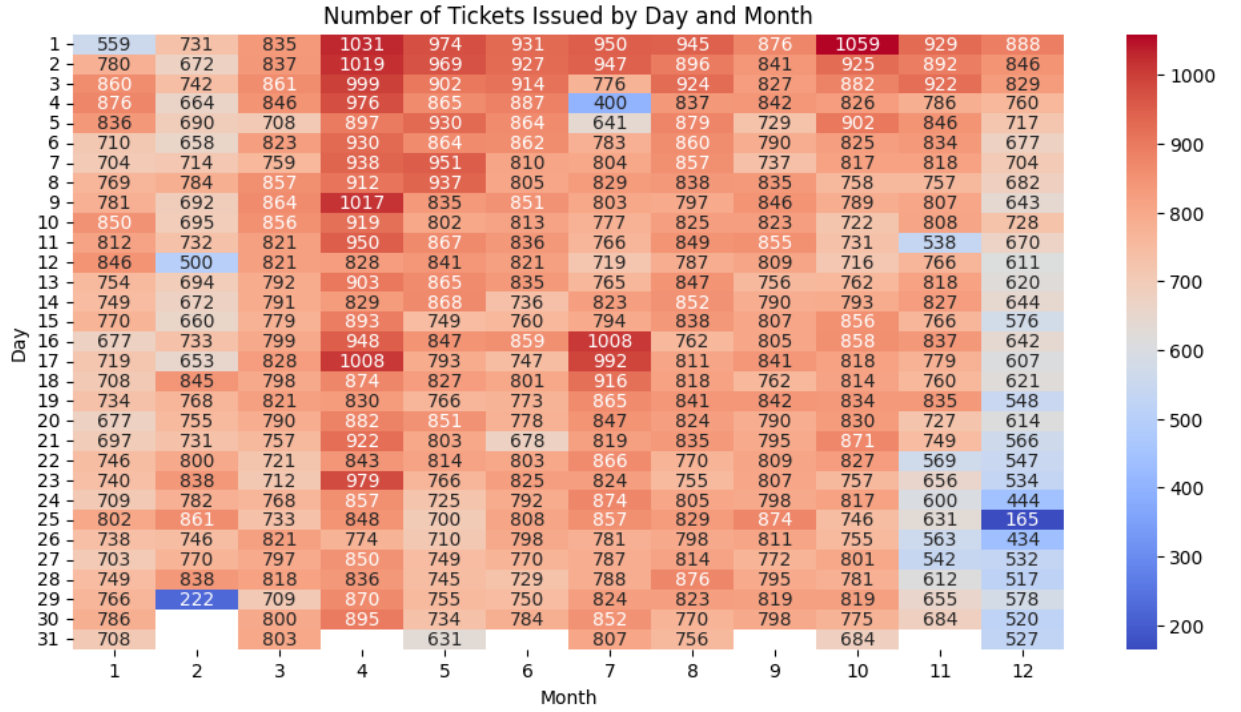
plot them

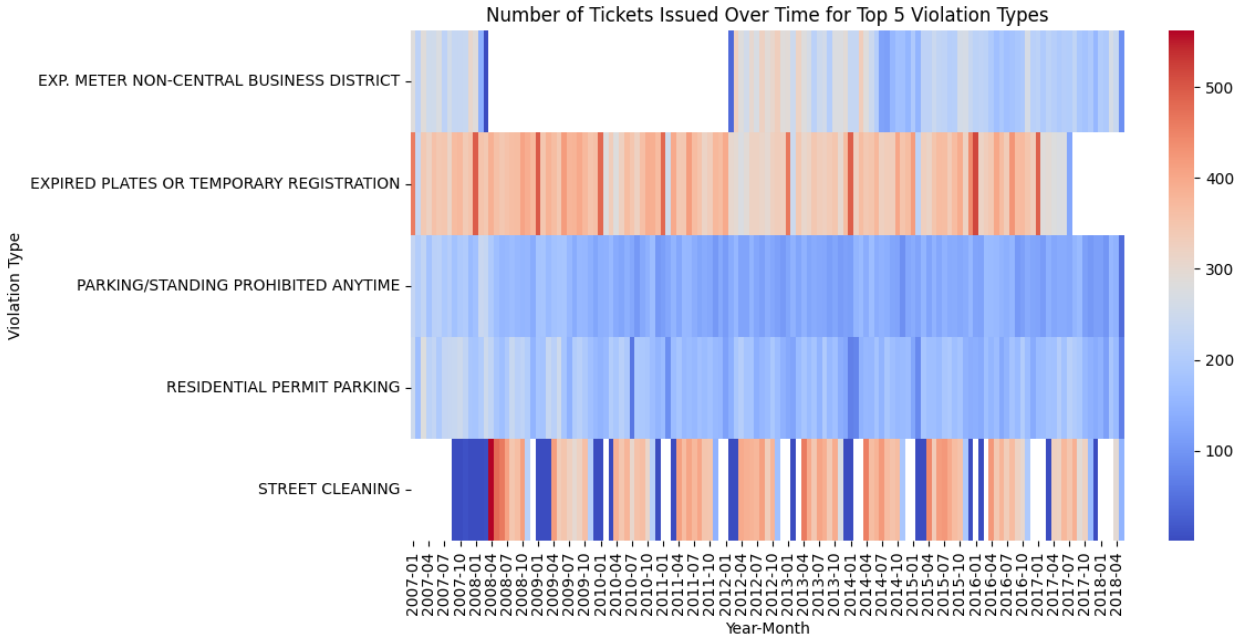
plt.tight_layout() # Adjust layout to prevent clipping plt.show()

#3.6 #for heatmap #One advantage of visualizing ticket issuance across different times of the year is that it helps identify trends, such as whether more tickets are issued during certain seasons, months, or days. However, a drawback is that it doesn't offer detailed information about the types of violations or specific vehicles involved, and it may be challenging to interpret for those unfamiliar with heatmaps or seasonal patterns.

#for lasagna #One advantage is that it effectively shows how ticket issuance varies over time for different violation types, allowing you to visualize trends and identify spikes in enforcement for specific violations. However, a drawback is that it can be harder to interpret compared to simpler bar plots or line charts and may become overwhelming if there are too many categories or time points to analyze.

#3.7 #The **Lasagna Plot** effectively shows uneven enforcement over time, highlighting changes in ticket issuance for different violations. Its heatmap design makes it easy to spot trends, such as peaks and dips, clearly displaying variations across time periods.





#3.6

#for heatmap

#One advantage of visualizing ticket issuance across different times of the year is that it helps identify trends, such as whether more tickets are issued during certain seasons, months, or days. However, a drawback is that it doesn't offer detailed information about the types of

violations or specific vehicles involved, and it may be challenging to interpret for those unfamiliar with heatmaps or seasonal patterns.

#for lasagna
#One advantage is that it effectively shows how ticket issuance varies over time for different violation types, allowing you to visualize trends and identify spikes in enforcement for specific violations. However, a drawback is that it can be harder to interpret compared to simpler bar plots or line charts and may become overwhelming if there are too many categories or time points to analyze.

#3.7
#The **Lasagna Plot** effectively shows uneven enforcement over time, highlighting changes in ticket issuance for different violations. Its heatmap design makes it easy to spot trends, such as peaks and dips, clearly displaying variations across time periods.

#3.3 #my comoputer crashed on this question for some unfathomable reasons

#3.4 import matplotlib.pyplot as plt import seaborn as sns

Step 1: Extract the month and day from the 'issue_date'

df_1['month'] = df_1['issue_date'].dt.month df_1['day'] = df_1['issue_date'].dt.day

generate pivot table for heatmap

heatmapyy = df_1.pivot_table(index='day', columns='month', values='ticket_number', aggfunc='count')

create heatmap

plt.figure(figsize=(12, 6)) sns.heatmap(heatmapyy, cmap='coolwarm', annot=True, fmt='g')

Add asesthtic

plt.title('Number of Tickets Issued by Day and Month') plt.ylabel('Day') plt.xlabel('Month')

plt.show()

#3.5 # Select data for the top 5 most common violation types big_5 = df_1['violation_description'].value_counts().head(5).index # Get the top 5 violations df_36 = df_1[df_1['violation_description'].isin(big_5)] # Subset the data to only include these violations

Extract month and year-month columns for time-based analysis

df_36['month'] = df_36['issue_date'].dt.month # Extract the month df_36['yr_n_month'] = df_36['issue_date'].dt.to_period('M') # Create a column with the year and month

Create a pivot table with 'yr_n_month' as rows and 'violation_description' as columns