30538 Problem Set 2: Parking Tickets

Huiting Zhang

2024-10-17

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **_HZ_**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **_HZ_** (1 point)
3. Late coins used this pset: **_0_* *Late coins left after submission: **_3_**
4. Knit your `ps2.qmd` to make `ps2.pdf`.
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
5. Push `ps2.qmd` and `ps2.pdf` to your github repo. It is fine to use Github Desktop.
6. Submit `ps2.pdf` via Gradescope (4 points)
7. Tag your submission in Gradescope

```
import pandas as pd
import altair as alt
alt.renderers.enable("png")
import time

import warnings
warnings.filterwarnings('ignore')
```

Data cleaning continued (15 points)

1.

```
file_path =
 ↪  'F:\python2\student30538\problem_sets\ps2\data\parking_tickets_one_percent.csv'
parking_tickets = pd.read_csv(file_path)
```

```
def na_count(df):
    na = df.isna().sum()
    df = pd.DataFrame({
        'Variable': na.index,
        'NA_Count': na.values
    })
    return df

na_result = na_count(parking_tickets)
print(na_result)
```

```
                 Variable  NA_Count
0             Unnamed: 0         0
1          ticket_number         0
2             issue_date         0
3     violation_location         0
4    license_plate_number         0
5     license_plate_state        97
6      license_plate_type      2054
7                 zipcode     54115
8          violation_code         0
9   violation_description         0
10                   unit        29
11       unit_description         0
12           vehicle_make         0
13     fine_level1_amount         0
14     fine_level2_amount         0
15     current_amount_due         0
16         total_payments         0
17           ticket_queue         0
18      ticket_queue_date         0
19           notice_level     84068
20    hearing_disposition    259899
21          notice_number         0
22                officer         0
23                address         0
```

2.

The three columns with significantly more missing values are 'zipcode', 'notice_level', and 'hearing_disposition'. The reasons are as follows.

'zipcode' column: many parking tickets have incomplete or improperly formatted location records. ProPublica noted that about 12% of tickets could not be accurately geocoded due to address misspellings or input errors, leading to missing geographic information like ZIP codes. Additionally, some tickets might have been incorrectly assigned to the wrong area due to these data issues.

'notice_level' column: this field is relevant only for tickets that have reached a specific enforcement stage, such as when a formal notice is issued due to non-payment. If a ticket hasn't progressed to this stage, the notice_level remains blank.

'hearing_disposition'column: most tickets are not contested, so they never go through a hearing process, according to ProPublica. As a result, the hearing_disposition field is left empty for tickets that weren't disputed or did not require a formal hearing outcome.

3.

```
sticker_data =
 ↪  parking_tickets[parking_tickets['violation_description'].str.contains('NO
 ↪  CITY STICKER')]
sticker_data = sticker_data[['violation_code', 'issue_date']]
sticker_data['issue_date'] = pd.to_datetime(sticker_data['issue_date'],
 ↪  errors='coerce')

sticker_data['year'] = sticker_data['issue_date'].dt.year
yearly_freq = sticker_data.groupby(['year',
 ↪  'violation_code']).size().reset_index(name='count')

yearly_freq
```

	year	violation_code	count
0	2007	0964125	2264
1	2007	0976170	7
2	2008	0964125	2212
3	2008	0976170	1
4	2009	0964125	2149
5	2009	0976170	3
6	2010	0964125	1985
7	2010	0976170	2
8	2011	0964125	1933
9	2011	0976170	2
10	2012	0964125	215
11	2012	0964125B	1977

	year	violation_code	count
12	2012	0964125C	24
13	2013	0964125B	2567
14	2013	0964125C	40
15	2014	0964125B	2025
16	2014	0964125C	20
17	2015	0964125B	2467
18	2015	0964125C	19
19	2016	0964125B	2264
20	2016	0964125C	17
21	2017	0964125B	2256
22	2017	0964125C	11
23	2018	0964125B	690

The old violation codes are 0964125 and 0976170, while the new violation codes, which appeared in 2012, are 0964125B and 0964125C. The code 0964125B refers to "NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS," while 0964125C refers to "NO CITY STICKER VEHICLE OVER 16,000 LBS."

4.

```
codes = ['0964125', '0976170', '0964125B']
fine_data = parking_tickets[parking_tickets['violation_code'].isin(codes)]

initial_fines =
 ↪  fine_data.groupby('violation_code')['fine_level1_amount'].unique().reset_index()
initial_fines.columns = ['violation_code', 'initial_fine_amount']

print(initial_fines)
```

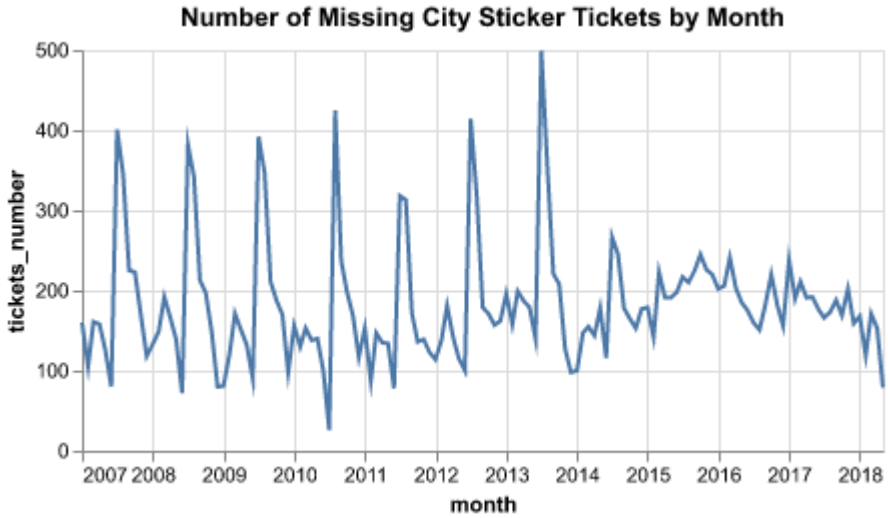
```
  violation_code initial_fine_amount
0        0964125               [120]
1       0964125B               [200]
2        0976170               [120]
```

The new violation code, 0964125B, which was introduced in 2012, has a higher initial fine of $200 compared to the previous $120 fine under code 0964125 and 0976170. This finding supports the claims in some ProPublica articles that there was an increase in the fine amount for not having a city sticker.

Revenue increase from "missing city sticker" tickets (20 Points)

1.

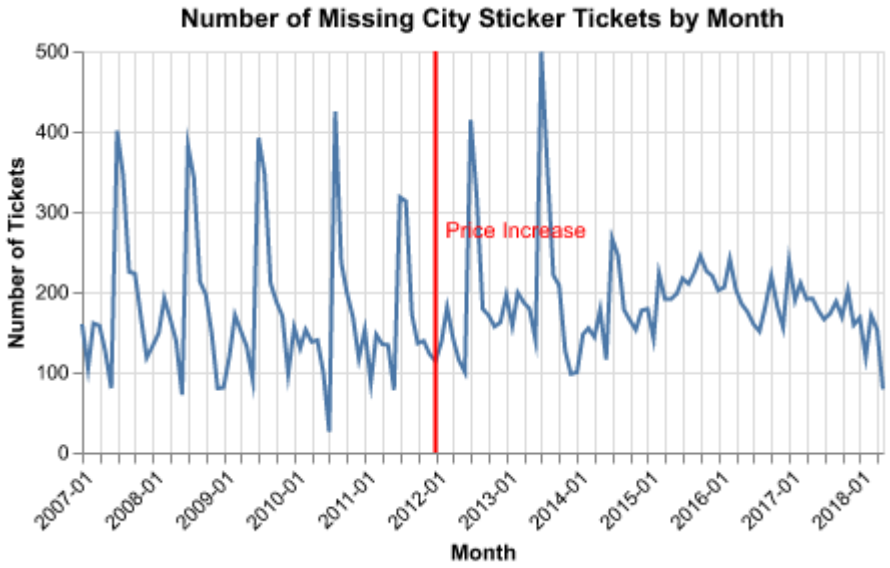
```
parking_tickets['city_sticker_violation'] =
 ↪  parking_tickets['violation_code'].apply(lambda x: 'missing_city_sticker'
 ↪  if x in ['0964125', '0976170', '0964125B'] else 'other')

city_sticker = parking_tickets[parking_tickets['city_sticker_violation'] ==
 ↪  'missing_city_sticker']

city_sticker['issue_date'] = pd.to_datetime(city_sticker['issue_date'],
 ↪  errors='coerce')
city_sticker['month'] = city_sticker['issue_date'].dt.strftime('%Y-%m')

city_sticker_month =
 ↪  city_sticker.groupby('month').size().reset_index(name='tickets_number')

alt.Chart(city_sticker_month).mark_line().encode(
    x=alt.X('month:T'),
    y=alt.Y('tickets_number:Q')
).properties(
    title='Number of Missing City Sticker Tickets by Month',
    width=400,
    height=200
)
```



2.

I used this help page: https://altair-viz.github.io/user_guide/customization.html

```
chart = alt.Chart(city_sticker_month).mark_line().encode(
    x=alt.X('month:T', axis=alt.Axis(format='%Y-%m', labelAngle=-45,
 ↪  tickCount=40, title='Month')),
    y=alt.Y('tickets_number:Q',
            axis=alt.Axis(title='Number of Tickets'))
).properties(
    title='Number of Missing City Sticker Tickets by Month',
    width=400,
    height=200
)

line = alt.Chart(pd.DataFrame({
    'month': ['2012-01-01']
})).mark_rule(color='red', strokeWidth=2).encode(
    x='month:T'
).properties(
    title='Number of Missing City Sticker Tickets by Month with Price
 ↪  Increase Mark'
)

text = alt.Chart(pd.DataFrame({
    'month': ['2012-01-01'],
    'label': ['Price Increase']
})).mark_text(align='left', dx=5, dy=-10, color='red').encode(
    x='month:T',
    text='label:N'
)

final_chart = chart + line + text

final_chart
```



3.

```
tickets_2011 = city_sticker[city_sticker['issue_date'].dt.year == 2011]
num_tickets_2011 = tickets_2011.shape[0]

old_price = 120
new_price = 200
price_increase = new_price - old_price

revenue_increase = num_tickets_2011 * price_increase * 100

print(f"Estimated total revenue increase: ${revenue_increase:.2f}")
```

```
Estimated total revenue increase: $15480000.00
```

4.

```
tickets_2011 = city_sticker[city_sticker['issue_date'].dt.year == 2011]
tickets_2012 = city_sticker[city_sticker['issue_date'].dt.year == 2012]

# Calculate repayment rate for 2011 and 2012 using the 'ticket_queue' column
rate_2011 = tickets_2011[tickets_2011['ticket_queue'] == 'Paid'].shape[0] /
 ↪  tickets_2011.shape[0]
rate_2012 = tickets_2012[tickets_2012['ticket_queue'] == 'Paid'].shape[0] /
 ↪  tickets_2012.shape[0]
```

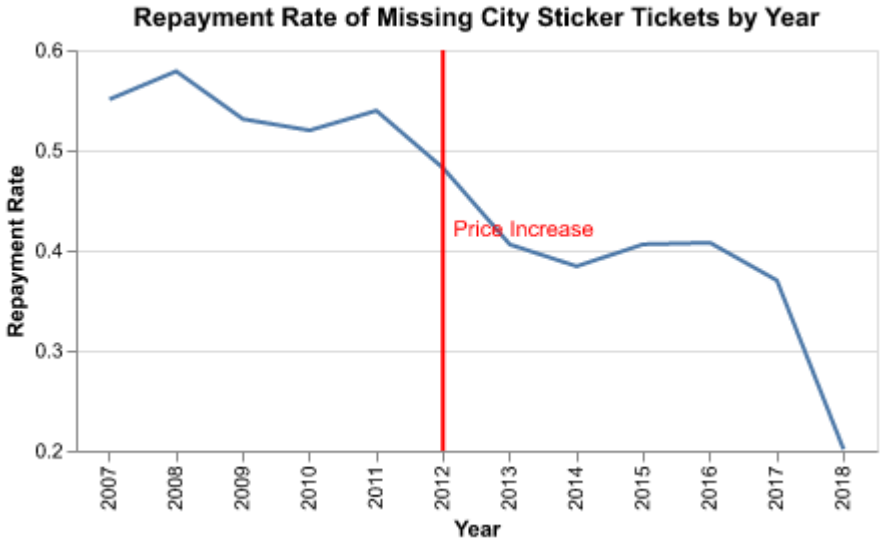
```
rate_change = (rate_2012 - rate_2011) * 100

old_price = 120
new_price = 200

revenue_2011 = tickets_2011.shape[0] * old_price * rate_2012
revenue_2012 = tickets_2011.shape[0] * new_price * rate_2012

revenue_change = (revenue_2012 - revenue_2011) *100

print(f"Repayment rate change: {rate_change:.2f}%")
print(f"Projected revenue change: ${revenue_change:.2f}")
```

```
Repayment rate change: -5.73%
Projected revenue change: $7464580.29
```

5.

```
city_sticker['year'] = city_sticker['issue_date'].dt.year

rate = city_sticker.groupby('year').apply(lambda x: (x['ticket_queue'] ==
↪  'Paid').sum() / len(x)).reset_index(name='repayment_rate')

chart = alt.Chart(rate).mark_line().encode(
    x=alt.X('year:O', title='Year'),
    y=alt.Y('repayment_rate:Q', scale=alt.Scale(domain=[0.2, 0.6]),
↪  title='Repayment Rate')
).properties(
    title='Repayment Rate of Missing City Sticker Tickets by Year',
    width=400,
    height=200
)

line = alt.Chart(pd.DataFrame({
    'year': [2012]
})).mark_rule(color='red', strokeWidth=2).encode(
    x='year:O'
)

text = alt.Chart(pd.DataFrame({
    'year': [2012],
```

```
    'label': ['Price Increase']
})).mark_text(align='left', dx=5, dy=-10, color='red').encode(
    x='year:O',
    text='label:N'
)

final_chart = chart + line + text

final_chart
```

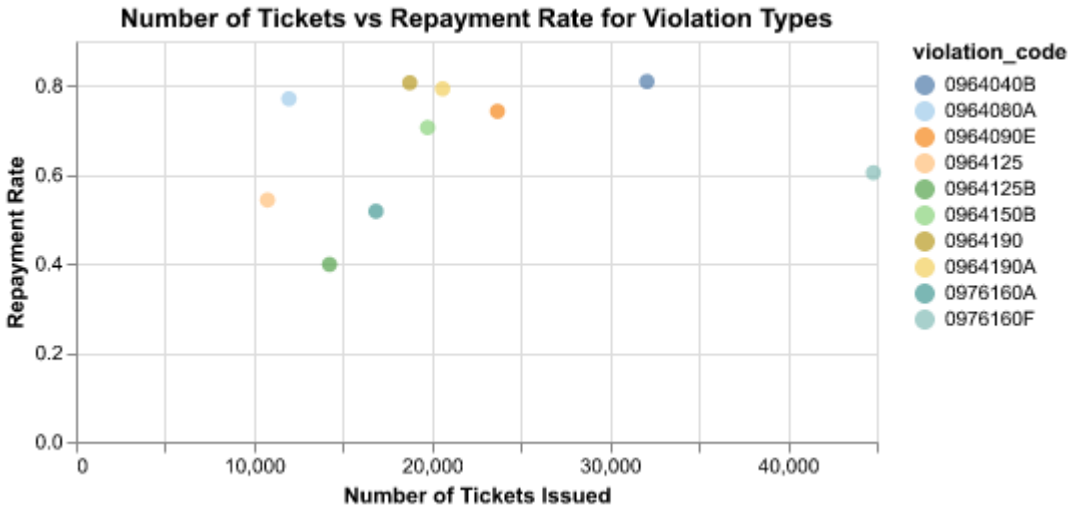


The price increase was introduced in 2012. Following this price increase, there was a noticeable decline in the repayment rate, suggesting that the higher fine amount may have discouraged people from paying their tickets. Then it stabilized between 2014 and 2016, and further dropped to 20% by 2018, reflecting a cumulative effect of the higher fines over time.

6.

```
violation = parking_tickets.groupby('violation_code').agg(
    num_tickets=('ticket_number', 'count'),
    repayment_rate=('ticket_queue', lambda x: (x == 'Paid').sum() / len(x))
).reset_index()

violation = violation[violation['num_tickets'] > 10000]

chart = alt.Chart(violation).mark_circle(size=60).encode(
    alt.X('num_tickets:Q', title='Number of Tickets Issued'),
```

```
    alt.Y('repayment_rate:Q', title='Repayment Rate'),
    alt.Color('violation_code:N', scale=alt.Scale(scheme='tableau20')),
    tooltip=['violation_code', 'num_tickets', 'repayment_rate']
).properties(
    title='Number of Tickets vs Repayment Rate for Violation Types',
    width=400,
    height=200
)

chart
```



I will recommend increase the price of violation types which are 'EXPIRED PLATES OR TEMPORARY REGISTRATION', 'STREET CLEANING OR SPECIAL EVENT' and 'RESIDENTIAL PERMIT PARKING'. Their violation codes are 0976160F, 0964040B, and 0964090E respectively. As shown in the figure, these violation types are on the top right corner, which means they have highest repayment rate as well as highest number of tickets issued, and these will enable them to generate the largest amount of revenue.

Headlines and sub-messages (20 points)

1.

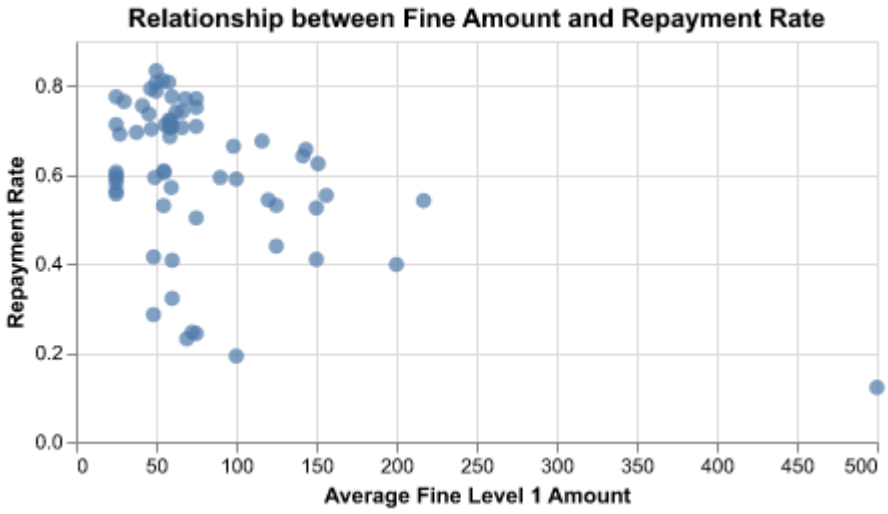
```
violation_df = parking_tickets.groupby('violation_description').agg(
    num_tickets=('ticket_number', 'count'),
    repayment_rate=('ticket_queue', lambda x: (x == 'Paid').sum() / len(x)),
    avg_fine_level1=('fine_level1_amount', 'mean')
```

```
).reset_index()

violation_dfs = violation_df.sort_values(by='num_tickets', ascending=False)

top5_violation = violation_dfs.head(5)

print(top5_violation)
```

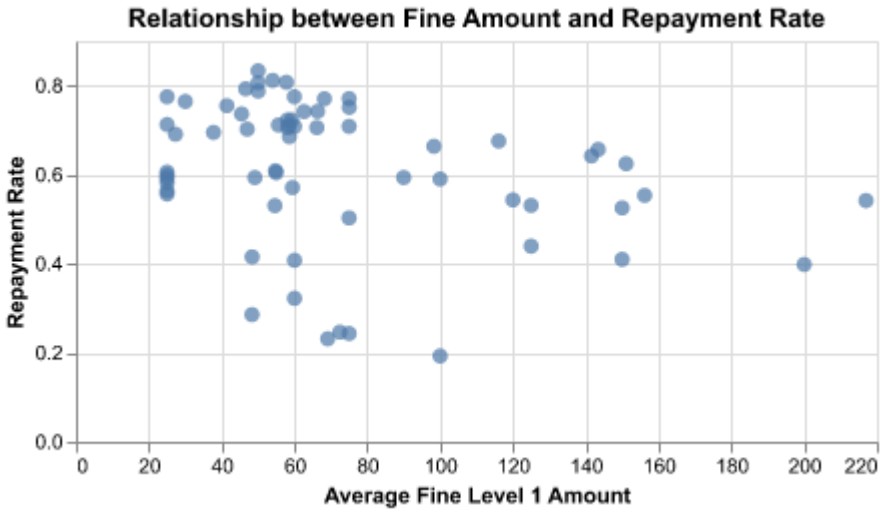
```
                        violation_description  num_tickets  repayment_rate  \
23   EXPIRED PLATES OR TEMPORARY REGISTRATION        44811        0.604361
101                            STREET CLEANING        28712        0.811612
90                 RESIDENTIAL PERMIT PARKING        23683        0.742262
19   EXP. METER NON-CENTRAL BUSINESS DISTRICT        20600        0.792913
81        PARKING/STANDING PROHIBITED ANYTIME        19753        0.705817

     avg_fine_level1
23         54.968869
101        54.004249
90         66.338302
19         46.598058
81         66.142864
```

2.

```
violation_df100 = violation_df[violation_df['num_tickets'] > 100]

scatter_plot1 = alt.Chart(violation_df100).mark_circle(size=60).encode(
    x=alt.X('avg_fine_level1:Q', title='Average Fine Level 1 Amount'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate'),
    tooltip=['violation_description', 'avg_fine_level1', 'repayment_rate']
).properties(
    title='Relationship between Fine Amount and Repayment Rate',
    width=400,
    height=200
)

scatter_plot1
```



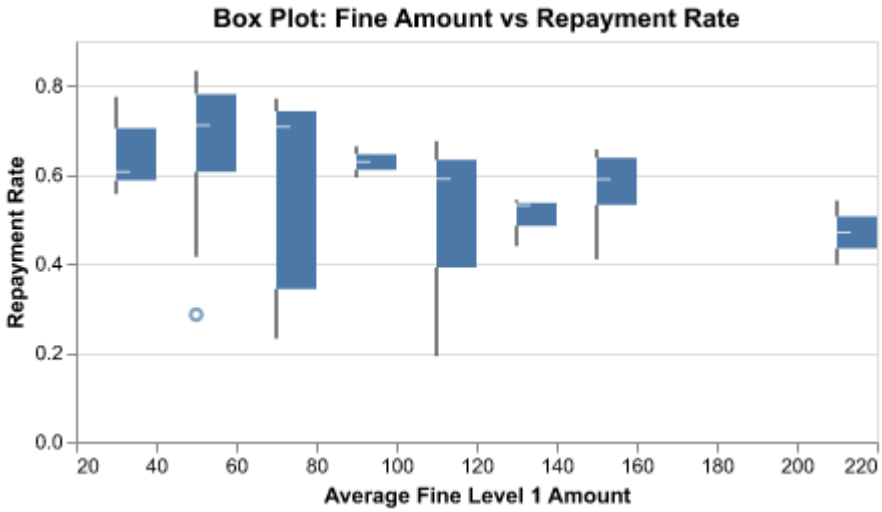

```
violation_df100 = violation_df100[violation_df100['avg_fine_level1'] < 250]

scatter_plot = alt.Chart(violation_df100).mark_circle(size=60).encode(
    x=alt.X('avg_fine_level1:Q', title='Average Fine Level 1 Amount'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate'),
    tooltip=['violation_description', 'avg_fine_level1', 'repayment_rate']
).properties(
    title='Relationship between Fine Amount and Repayment Rate',
    width=400,
    height=200
)

scatter_plot
```



Headline: Higher fine amounts generally correlate with lower repayment rates.

Sub-message: When fines increase beyond a certain point (around $100), repayment rates tend to decrease, particularly as fines exceed $150, though the variability is higher for lower fines.

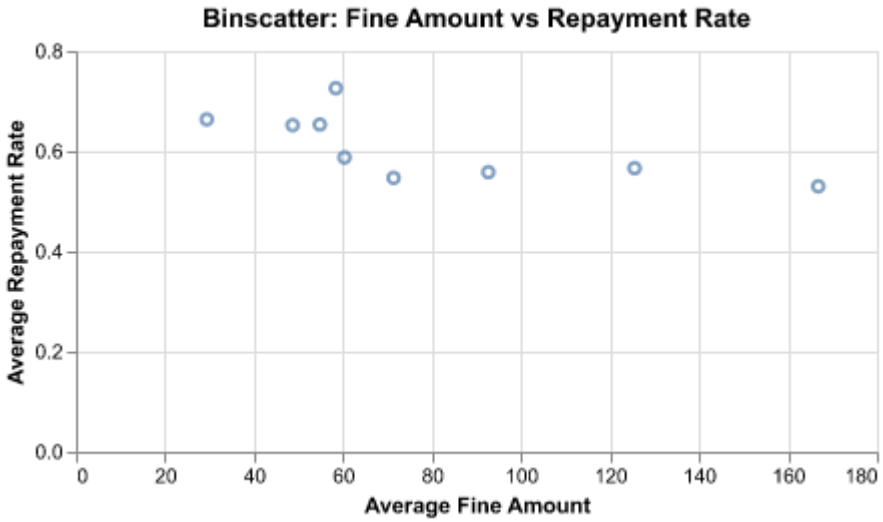
```
box_plot = alt.Chart(violation_df100).mark_boxplot().encode(
    alt.X('avg_fine_level1:Q', bin=alt.Bin(maxbins=10), title='Average Fine
 ↪  Level 1 Amount'),
    alt.Y('repayment_rate:Q', title='Repayment Rate')
).properties(
    title='Box Plot: Fine Amount vs Repayment Rate',
    width=400,
    height=200
)

box_plot
```



Headline: Distribution of repayment rates varies significantly across fine levels.

Sub-message: For fines below $100, there is significant variation in repayment rates. However, fines above $100 show more consistent repayment behavior with narrower distributions, indicating lower variance in repayment rates for higher fines.

```
violation_df100['fine_bin'] = pd.qcut(violation_df100['avg_fine_level1'],
 ↪  q=10, duplicates='drop').astype(str)

df_binned = violation_df100.groupby('fine_bin').agg(
    avg_fine=('avg_fine_level1', 'mean'),
    avg_repayment_rate=('repayment_rate', 'mean')
).reset_index()

binscatter = alt.Chart(df_binned).mark_point().encode(
    alt.X('avg_fine:Q', title='Average Fine Amount'),
    alt.Y('avg_repayment_rate:Q', title='Average Repayment Rate')
).properties(
    title='Binscatter: Fine Amount vs Repayment Rate',
    width=400,
    height=200
)

binscatter
```



Headline: Repayment rate decreases only a little bit but not much as average fine amount increases.

Sub-message: Lower fine amounts tend to have a wider range of repayment rate, and the average repayment rate is around 60%.

3.

I would bring the binscatter plot to the City Clerk because it simplifies the data by aggregating fines into bins, making it easier to interpret the overall trend without the clutter of individual data points. From the binscatter plot, it can be seen that the repayment rate is concentrated around 60%, and while the increase in fine amounts does lead to a decline in the number of tickets issued, the drop is not very significant. The plot effectively conveys that higher fines do impact repayment rates and ticket issuance, but the overall reduction in tickets is relatively small. Since the City Clerk doesn't understand regressions or complex statistics, this plot provides a straightforward and easy-to-understand visual summary.

Understanding the structure of the data and summarizing it (Lecture 5, 20 Points)

1.

```
parking_tickets['fine_ratio'] = parking_tickets['fine_level2_amount'] /
 ↪  parking_tickets['fine_level1_amount']

all_double = parking_tickets['fine_ratio'].eq(2).all()
```

```
if all_double:
    print("All violation types double in price if unpaid.")
else:
    print("Not all violation types double in price if unpaid.")
```

```
Not all violation types double in price if unpaid.
```

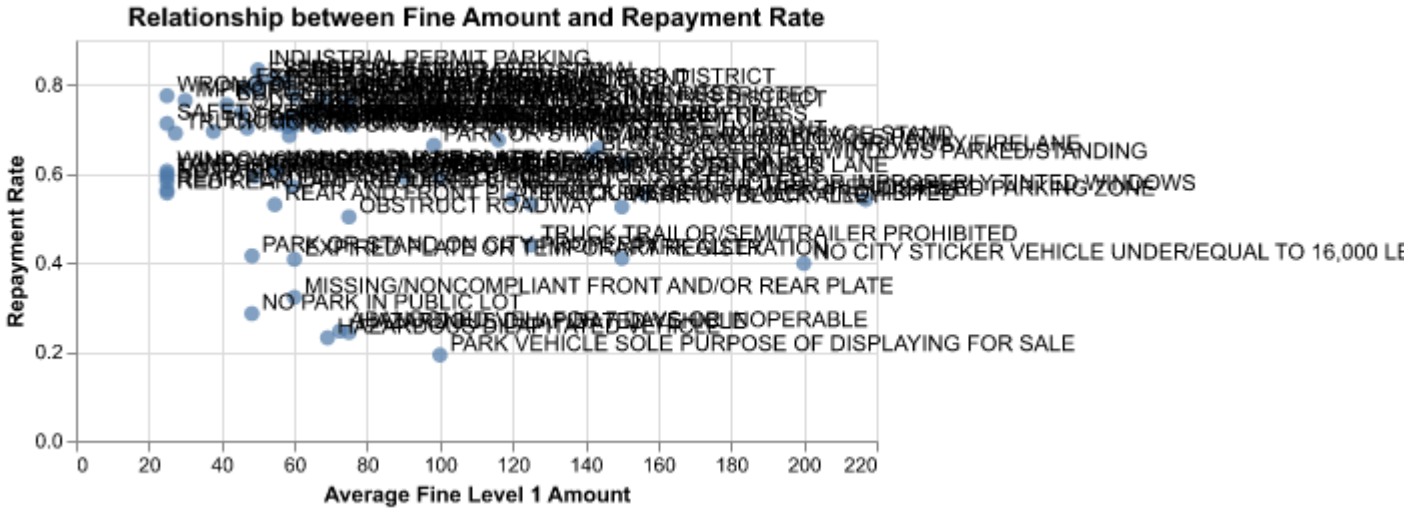
```
violation_counts =
 ↪  parking_tickets.groupby('violation_description').size().reset_index(name='num_tickets')
violation_100 = violation_counts[violation_counts['num_tickets'] >= 100]

violation_fines = parking_tickets.merge(violation_100,
 ↪  on='violation_description')

non_double_fines = violation_fines[(violation_fines['fine_ratio'] != 2)]

fine_increase = non_double_fines.groupby('violation_description').agg(
    avg_fine_level1=('fine_level1_amount', 'mean'),
    avg_fine_level2=('fine_level2_amount', 'mean'),
    fine_increase=('fine_ratio', 'mean'),
    num_tickets=('num_tickets', 'mean')
).reset_index()

print(fine_increase)
```

```
                     violation_description  avg_fine_level1  avg_fine_level2  \
0      BLOCK ACCESS/ALLEY/DRIVEWAY/FIRELANE            150.0            250.0
1                     DISABLED PARKING ZONE            200.0            250.0
2  NO CITY STICKER VEHICLE OVER 16,000 LBS.            500.0            775.0
3   OBSTRUCTED OR IMPROPERLY TINTED WINDOWS            250.0            250.0
4                       PARK OR BLOCK ALLEY            150.0            250.0
5                PARK/STAND ON BICYCLE PATH            150.0            250.0
6     SMOKED/TINTED WINDOWS PARKED/STANDING            250.0            250.0

   fine_increase  num_tickets
0       1.666667       1579.0
1       1.250000       2034.0
2       1.550000        131.0
3       1.000000        271.0
4       1.666667       2050.0
```

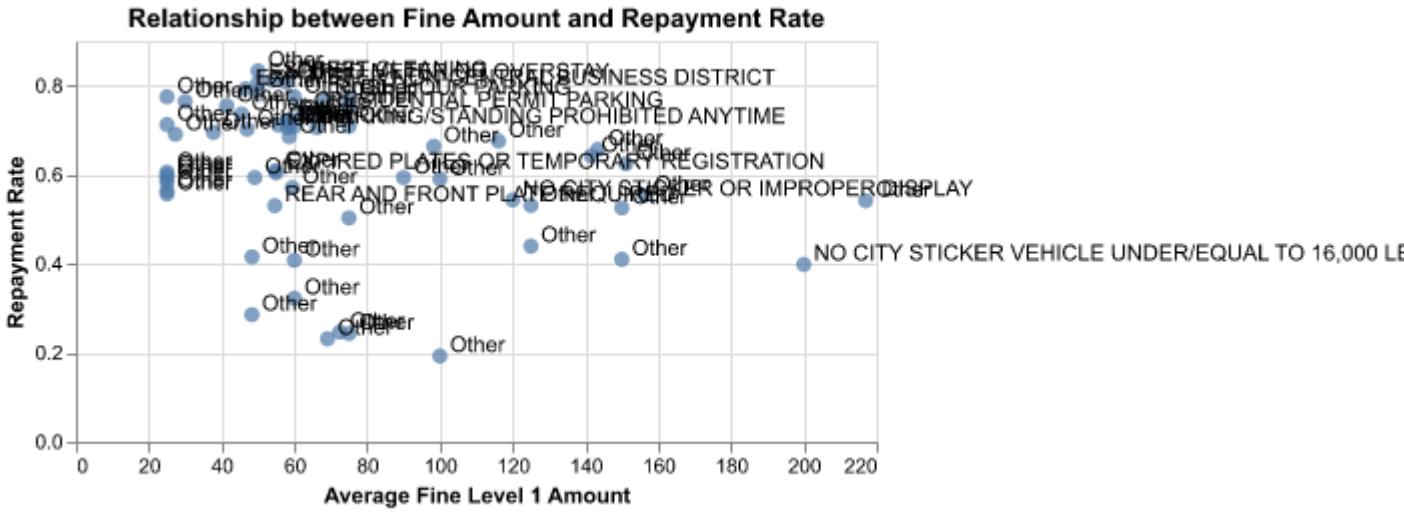
```
5       1.666667         236.0
6       1.000000        1697.0
```

2.
3.

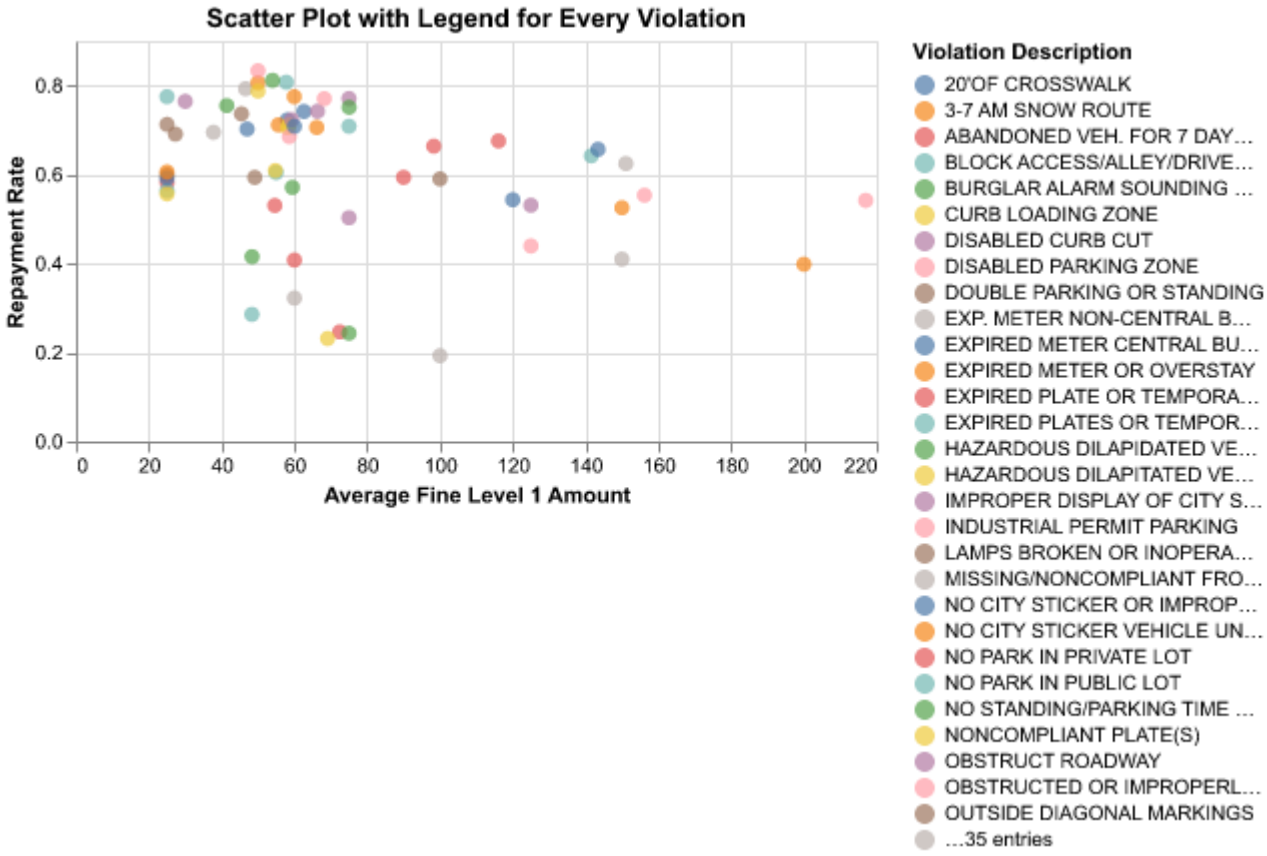
```
# a
text = scatter_plot.mark_text(align='left', dx=5, dy=-5).encode(
    text='violation_description:N'
)

scatter_plot + text
```




```
# revised a
top_10_violations = violation_df100.nlargest(10,
 ↪  'num_tickets')['violation_description'].tolist()

violation_df100['violation_label'] =
 ↪  violation_df100['violation_description'].apply(lambda x: x if x in
 ↪  top_10_violations else 'Other')

text_top_10 = scatter_plot.mark_text(align='left', dx=5, dy=-5).encode(
    text='violation_label:N'
)

scatter_plot + text_top_10
```

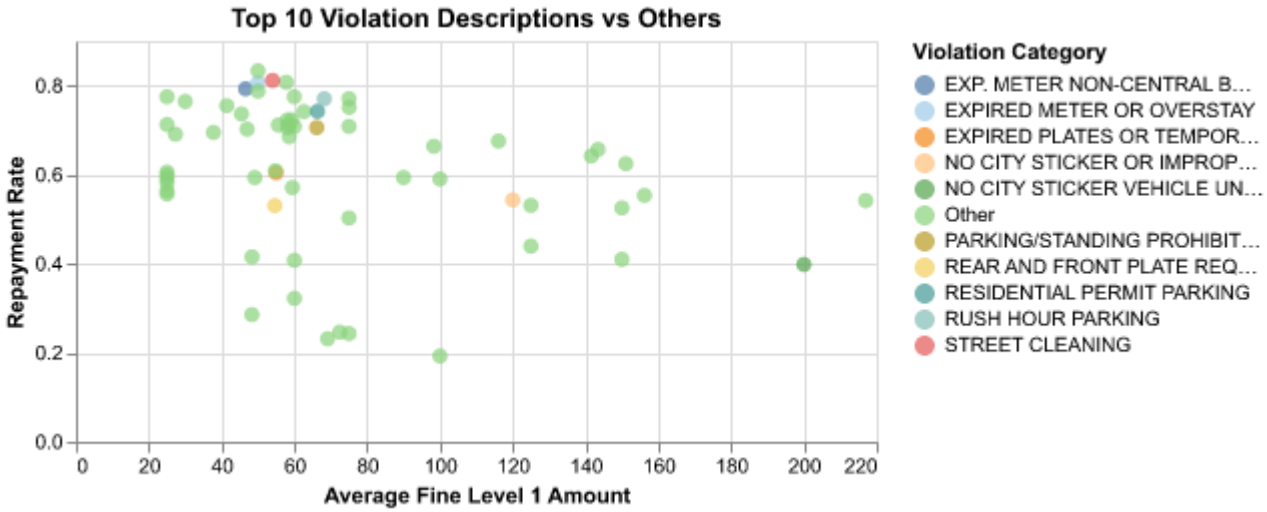



```
#b
scatter_plot_with_legend =
 ↪  alt.Chart(violation_df100).mark_circle(size=60).encode(
    x=alt.X('avg_fine_level1:Q', title='Average Fine Level 1 Amount'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate'),
    color=alt.Color('violation_description:N', title='Violation
 ↪  Description'),
    tooltip=['violation_description', 'avg_fine_level1', 'repayment_rate']
).properties(
    title='Scatter Plot with Legend for Every Violation',
    width=400,
    height=200
)

scatter_plot_with_legend
```



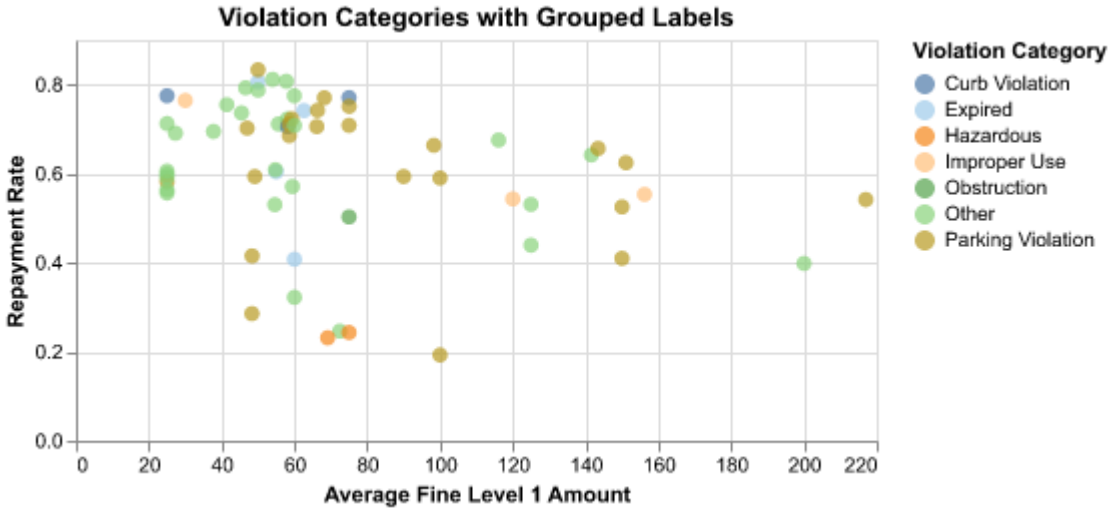

```
# revised b
top_10_violations = violation_df100.nlargest(10,
 ↪  'num_tickets')['violation_description'].tolist()

# Create a new column marking top 10 violations and 'Other'
violation_df100['violation_category'] =
 ↪  violation_df100['violation_description'].apply(
    lambda x: x if x in top_10_violations else 'Other'
)

# Plot with labeled categories
scatter_plot_top10 = alt.Chart(violation_df100).mark_circle(size=60).encode(
    x=alt.X('avg_fine_level1:Q', title='Average Fine Level 1 Amount'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate'),
    color=alt.Color('violation_category:N', title='Violation Category',
 ↪  scale=alt.Scale(scheme='tableau20')),
    tooltip=['violation_description', 'avg_fine_level1', 'repayment_rate']
```

```
).properties(
    title='Top 10 Violation Descriptions vs Others',
    width=400,
    height=200
)

scatter_plot_top10
```




```
# revised b
def categorize_violations(description):
    if 'EXPIRED' in description:
        return 'Expired'
    elif 'PARK' in description or 'STAND' in description:
        return 'Parking Violation'
    elif 'CURB' in description:
        return 'Curb Violation'
    elif 'IMPROPER' in description:
        return 'Improper Use'
    elif 'HAZARDOUS' in description:
        return 'Hazardous'
    elif 'OBSTRUCT' in description:
        return 'Obstruction'
    else:
        return 'Other'

violation_df100['violation_category'] =
 ↪  violation_df100['violation_description'].apply(categorize_violations)
```

```
scatter_plot_grouped =
 ↪  alt.Chart(violation_df100).mark_circle(size=60).encode(
    x=alt.X('avg_fine_level1:Q', title='Average Fine Level 1 Amount'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate'),
    color=alt.Color('violation_category:N', title='Violation Category',
 ↪  scale=alt.Scale(scheme='tableau20')),
    tooltip=['violation_description', 'avg_fine_level1', 'repayment_rate']
).properties(
    title='Violation Categories with Grouped Labels',
    width=400,
    height=200
)

scatter_plot_grouped
```



Extra Credit (max 5 points)

1.

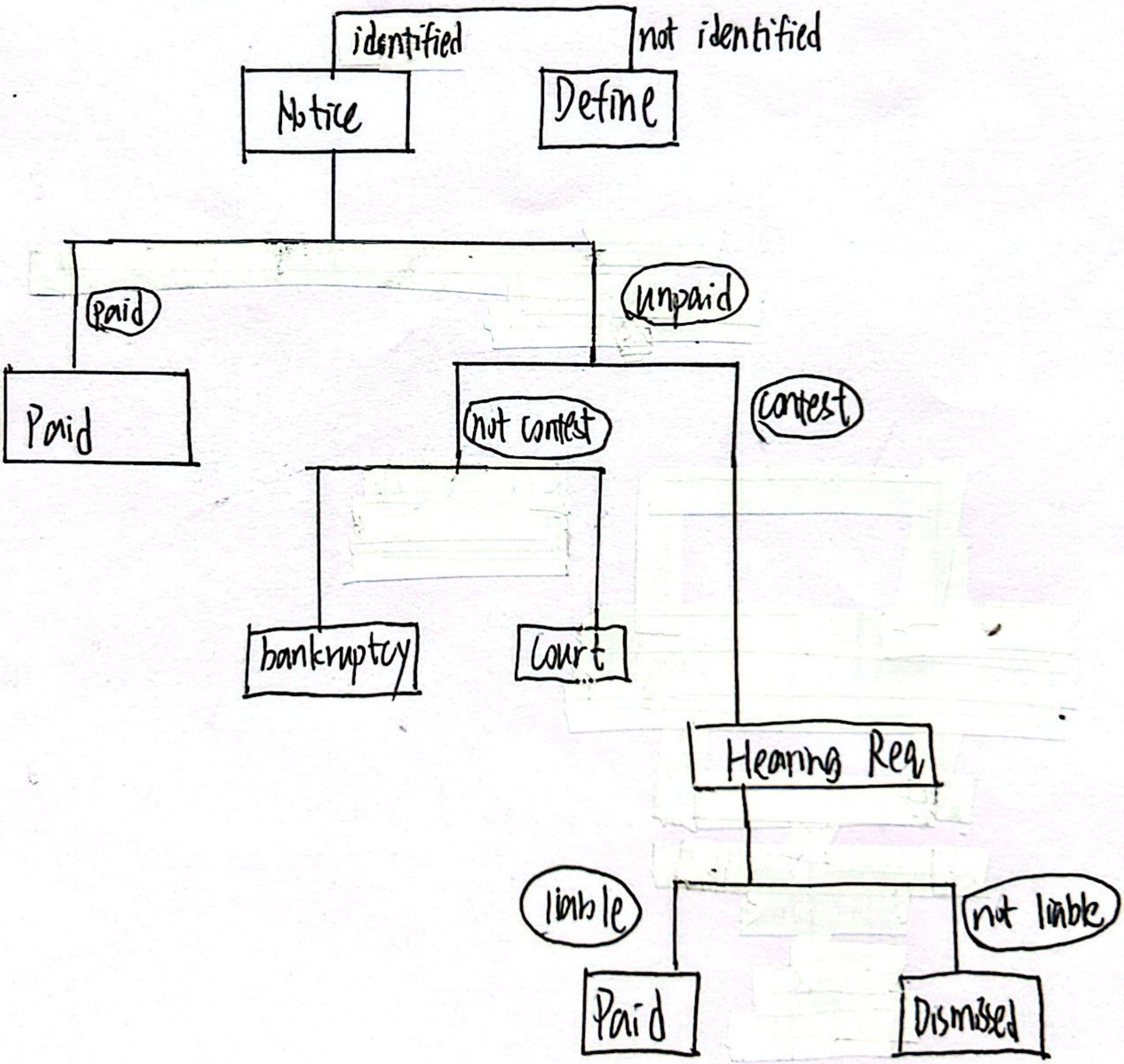
```
violation_counts = parking_tickets.groupby(['violation_code',
 ↪  'violation_description']).size().reset_index(name='count')

multiple_descriptions =
 ↪  violation_counts.groupby('violation_code').size().reset_index(name='description_count')
```

```
multiple_descriptions =
 ↪  multiple_descriptions[multiple_descriptions['description_count'] > 1]

multi_desc_violation_codes =
 ↪  violation_counts.merge(multiple_descriptions[['violation_code']],
 ↪  on='violation_code')

most_common_description =
 ↪  multi_desc_violation_codes.groupby('violation_code').apply(
    lambda x: x.loc[x['count'].idxmax(),
     ↪  'violation_description']).reset_index(name='most_common_description')

parking_tickets = parking_tickets.merge(most_common_description,
 ↪  on='violation_code', how='left')

top_three_codes =
 ↪  multi_desc_violation_codes.groupby('violation_code')['count'].sum().reset_index()
top_three_codes = top_three_codes.sort_values(by='count',
 ↪  ascending=False).head(3)

print(top_three_codes)
```

```
  violation_code  count
0       0964040B  32082
5       0976160A  16853
6       0976160B   3072
```

2.

Ticket Queue :



Notice Level:

(notice sent) VIOL — NaN (no notice sent)

VIOL

Not Paid

Paid

Hearing

liable

not liable

DETR

not Paid

(collection) FINL

paid

SEIZ (on boat list)

DLS (driver license suspension)