30538 Problem Set 2: Parking Tickets

Kohei Inagaki

Invalid Date

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **_KI_**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **_Yes_** (1 point)
3. Late coins used this pset: **_0_* *Late coins left after submission: **_3_**
4. Knit your `ps1.qmd` to make `ps1.pdf`.
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
5. Push `ps1.qmd` and `ps1.pdf` to your github repo. It is fine to use Github Desktop.
6. Submit `ps1.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.

```
#1-1
file_path = 'C:/Users/kohei/Python/parking_tickets_one_percent.csv'

def read_parking_tickets(file_path):
    df = pd.read_csv(file_path, index_col=False)
```

```
    na_count = df.isna().sum()
    na_count_df = pd.DataFrame({'Variable': na_count.index, 'Numbers of NA': na_count.values
    na_count_df = na_count_df[~na_count_df['Variable'].str.contains('Unnamed')]

    return na_count_df

df_na_count = read_parking_tickets(file_path)

print(df_na_count)
```

```
                 Variable  Numbers of NA
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.

```
##1-2
#The three variables-notice_level, hearing_disposition, and zipcode-have
more missing values because:
notice_level: This is usually only filled in when the ticket goes to a higher level,
such as when a fine is overdue, so it might not needed for most tickets.
```

hearing_disposition: This is only used when someone disputes the ticket
and a hearing is held, which doesn't happen often.
zipcode: If the location of the violation isn't fully recorded or is unclear,
the zipcode may be left blank, especially for tickets issued outside the city or
in uncertain areas.

3.

```
##1-3
file_path = 'C:/Users/kohei/Python/parking_tickets_one_percent.csv'
df = pd.read_csv(file_path, index_col=False)

filtered_df = df[
    (
        (df['violation_description'] == 'NO CITY STICKER OR IMPROPER DISPLAY') |
        (df['violation_description'] == 'NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 LBS.')
    ) &
    ((df['fine_level1_amount'] == 120) | (df['fine_level1_amount'] == 200))
]

old_violation_code = filtered_df[filtered_df['fine_level1_amount'] == 120]['violation_code']
new_violation_code = filtered_df[filtered_df['fine_level1_amount'] == 200]['violation_code']

print("Old violation code (120):")
for code in old_violation_code:
    print(code)

print("\nNew violation code (200):")
for code in new_violation_code:
    print(code)

old_subset = filtered_df[filtered_df['fine_level1_amount'] == 120]
new_subset = filtered_df[filtered_df['fine_level1_amount'] == 200]

print(f"\nOld violation code subset shape: {old_subset.shape}")
print(f"New violation code subset shape: {new_subset.shape}")
```

```
Old violation code (120):
964125
976170

New violation code (200):
```

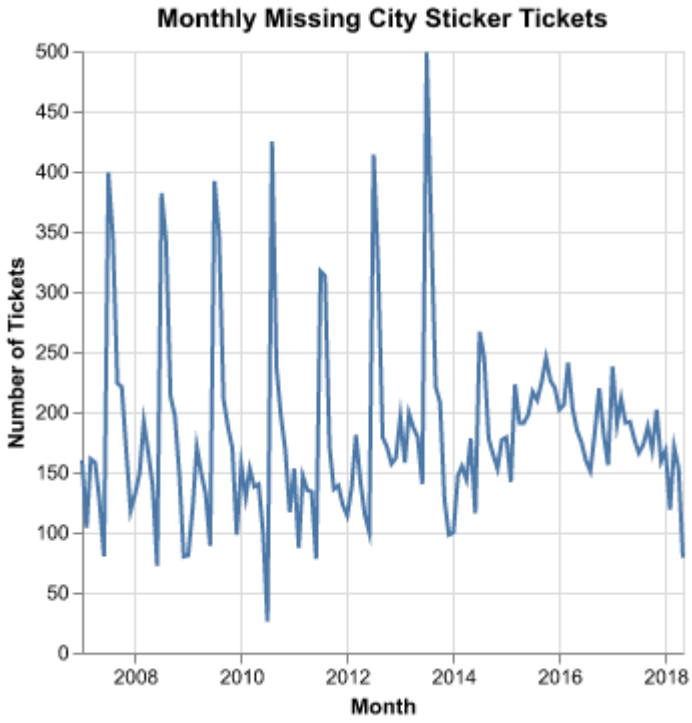
```
0964125B

Old violation code subset shape: (10773, 24)
New violation code subset shape: (14246, 24)
```

While the penalty was originally $120, it was increased to $200.

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

1.

```
file_path = 'C:/Users/kohei/Python/parking_tickets_one_percent.csv'
df = pd.read_csv(file_path)

df['combined_code'] = df.apply(lambda x: '964125_0964125B' if x['violation_code'] in ['96412

df['violation_date'] = pd.to_datetime(df['issue_date'], errors='coerce')

df['month'] = df['violation_date'].dt.to_period('M').dt.to_timestamp()

monthly_counts = df[df['combined_code'] != ''].groupby('month').size().reset_index(name='tic

chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('month:T', title='Month'),
    y=alt.Y('ticket_count:Q', title='Number of Tickets')
).properties(
    title='Monthly Missing City Sticker Tickets'
)

chart.show()
```

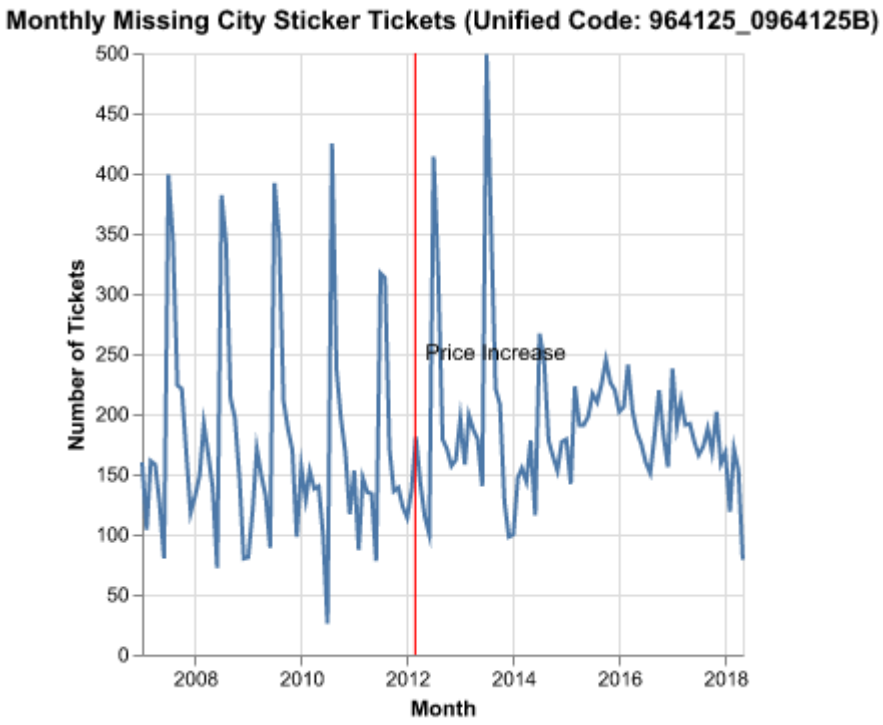


2.

```
df['violation_code'] = df['violation_code'].apply(lambda x: '964125_0964125B' if x in ['9641

df['violation_date'] = pd.to_datetime(df['issue_date'], errors='coerce')

df['month'] = df['violation_date'].dt.to_period('M').dt.to_timestamp()

monthly_counts = df[df['violation_code'] == '964125_0964125B'].groupby('month').size().reset_

price_increase_date = df[(df['fine_level1_amount'] == 200) & (df['violation_code'] == '96412

if pd.isna(price_increase_date):
    print("no dates")
else:
    print(f"price increase: {price_increase_date}")

    chart = alt.Chart(monthly_counts).mark_line().encode(
        x=alt.X('month:T', title='Month'),
        y=alt.Y('ticket_count:Q', title='Number of Tickets')
    ).properties(
```

```
        title='Monthly Missing City Sticker Tickets (Unified Code: 964125_0964125B)'
    )

    rule = alt.Chart(pd.DataFrame({'date': [price_increase_date]})).mark_rule(color='red').en
        x='date:T'
    )

    text = alt.Chart(pd.DataFrame({'date': [price_increase_date]})).mark_text(
        align='left',
        baseline='middle',
        dx=5
    ).encode(
        x='date:T',
        text=alt.value('Price Increase')
    )

    chart = chart + rule + text

    chart.show()

  ## I have referred to the following page:
  #  https://altair-viz.github.io/user_guide/marks/rule.html
```

```
price increase: 2012-02-25 02:00:00
```



3.

```
df_prior = df[(df['violation_date'] >= price_increase_date - pd.DateOffset(years=1)) &
              (df['violation_date'] < price_increase_date) &
              (df['violation_code'] == '964125_0964125B')]

revenue_before = df_prior['fine_level1_amount'].sum()

new_ticket_price = 200
projected_revenue_after = new_ticket_price * len(df_prior)

revenue_increase = (projected_revenue_after - revenue_before) * 100

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

```
Estimated revenue increase: $15336000.00
```

4.

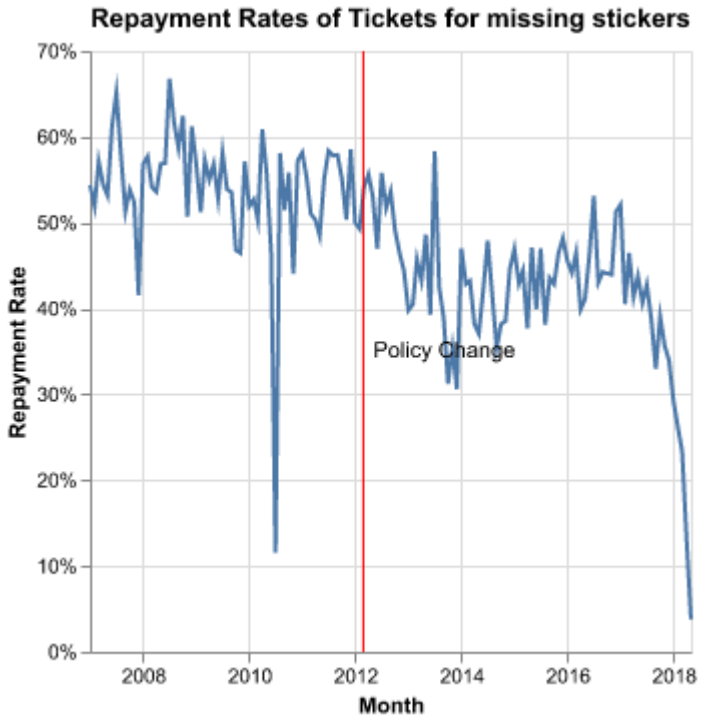
```
df_after = df[(df['violation_date'] >= price_increase_date) &
              (df['violation_date'] < price_increase_date + pd.DateOffset(years=1)) &
              (df['violation_code'] == '964125_0964125B')]

total_tickets_after = len(df_after)
paid_tickets_after = len(df_after[df_after['total_payments'] > 0])
repayment_rate_after = paid_tickets_after / total_tickets_after

df_prior = df[(df['violation_date'] >= price_increase_date - pd.DateOffset(years=1)) &
              (df['violation_date'] < price_increase_date) &
              (df['violation_code'] == '964125_0964125B')]

revenue_before = df_prior['fine_level1_amount'].sum()

new_ticket_price = 200
expected_paid_tickets = repayment_rate_after * len(df_prior)

projected_revenue_after = new_ticket_price * expected_paid_tickets

revenue_increase = (projected_revenue_after - revenue_before) * 100


print(f"Repayment rates: {repayment_rate_after:.2%}")
print(f"Estimated change in revenue: ${revenue_increase:.2f}")
```

```
Repayment rates: 49.98%
Estimated change in revenue: $-3842309.49
```

5.

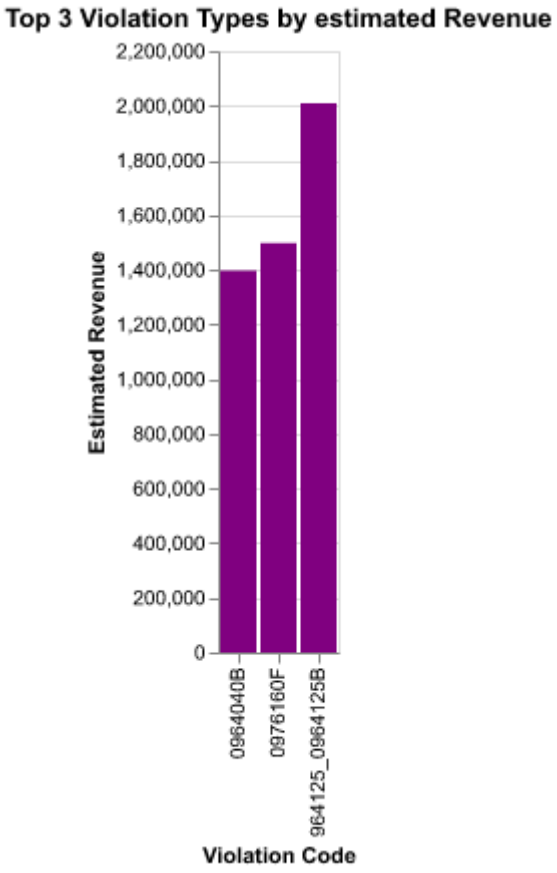
```
monthly_totals = df[df['violation_code'] == '964125_0964125B'].groupby('month').size().reset_
monthly_paid = df[(df['violation_code'] == '964125_0964125B') & (df['total_payments'] > 0)].g

repayment_rates = pd.merge(monthly_totals, monthly_paid, on='month', how='left')
repayment_rates['repayment_rate'] = repayment_rates['paid_tickets'] / repayment_rates['total_

chart = alt.Chart(repayment_rates).mark_line().encode(
    x=alt.X('month:T', title='Month'),
    y=alt.Y('repayment_rate:Q', title='Repayment Rate', axis=alt.Axis(format='%')),
).properties(
    title='Repayment Rates of Tickets for missing stickers'
)
```

```
rule = alt.Chart(pd.DataFrame({'date': [price_increase_date]})).mark_rule(color='red').encode
    x='date:T'
)

text = alt.Chart(pd.DataFrame({'date': [price_increase_date]})).mark_text(
    align='left',
    baseline='middle',
    dx=5
).encode(
    x='date:T',
    text=alt.value('Policy Change')
)

chart = chart + rule + text

chart.show()
```



6.

```
violation_summary = df.groupby('violation_code').agg(
    total_tickets=('ticket_number', 'size'),
    paid_tickets=('total_payments', lambda x: (x > 0).sum()),
    fine_amount=('fine_level1_amount', 'mean')
).reset_index()

violation_summary['repayment_rate'] = violation_summary['paid_tickets'] / violation_summary[

violation_summary['estimated_revenue'] = violation_summary['total_tickets'] * violation_summa

top_violations = violation_summary.nlargest(3, 'estimated_revenue')


bar_chart_revenue = alt.Chart(top_violations).mark_bar().encode(
    x=alt.X('violation_code:N', title='Violation Code'),
    y=alt.Y('estimated_revenue:Q', title='Estimated Revenue'),
    color=alt.value('purple'),
    tooltip=['violation_code', 'total_tickets', 'repayment_rate', 'fine_amount', 'estimated_
).properties(
    title='Top 3 Violation Types by estimated Revenue'
)

bar_chart_revenue.show()

print("Three recommended violation type to maximize the revenue:")
print(top_violations[['violation_code', 'total_tickets', 'repayment_rate', 'fine_amount', 'es
```




```
Three recommended violation type to maximize the revenue:
      violation_code  total_tickets  repayment_rate  fine_amount  \
105  964125_0964125B          25004        0.485242   165.579907
77           0976160F          44811        0.608065    54.968869
4            0964040B          32082        0.813073    53.583629

     estimated_revenue
105       2.008981e+06
77        1.497792e+06
4         1.397729e+06
```

Headlines and sub-messages (20 points)

1.

```
df['violation_date'] = pd.to_datetime(df['issue_date'], errors='coerce')

violation_summary = (
    df.groupby('violation_description')
    .agg(
        total_tickets=('ticket_number', 'size'),
        repayment_rate=('total_payments', lambda x: (x > 0).sum() / len(x)),
        average_fine=('fine_level1_amount', 'mean')
    )
    .reset_index()
)

top_5_violations = violation_summary.sort_values(by='total_tickets', ascending=False).head(5)

print("Top 5 Most Common Violation Descriptions:")
print(top_5_violations[['violation_description', 'repayment_rate', 'average_fine', 'total_ti
```

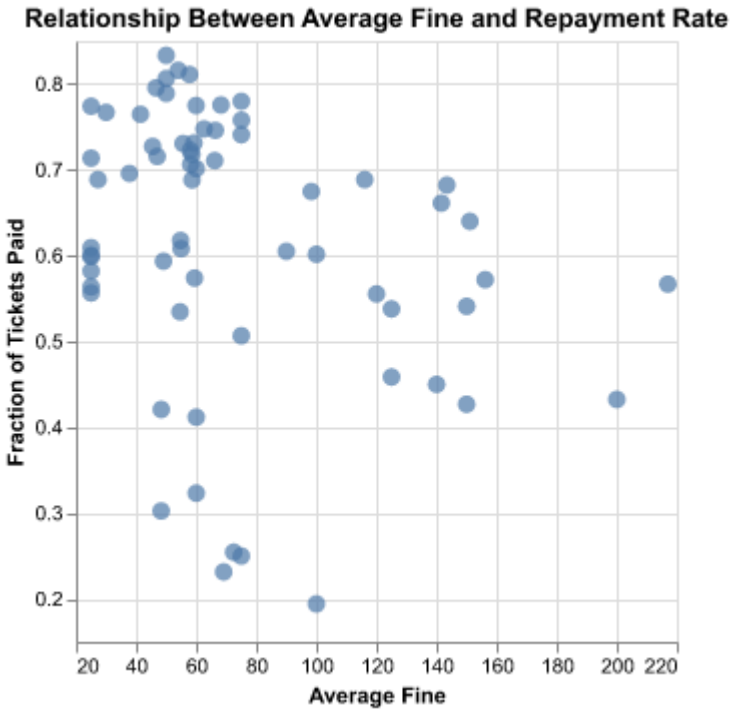
```
Top 5 Most Common Violation Descriptions:
                        violation_description  repayment_rate  average_fine  \
23   EXPIRED PLATES OR TEMPORARY REGISTRATION        0.608065     54.968869
101                            STREET CLEANING        0.815896     54.004249
90                 RESIDENTIAL PERMIT PARKING        0.745978     66.338302
19   EXP. METER NON-CENTRAL BUSINESS DISTRICT        0.795485     46.598058
81        PARKING/STANDING PROHIBITED ANYTIME        0.710677     66.142864

     total_tickets
23           44811
101          28712
90           23683
19           20600
81           19753
```

2.

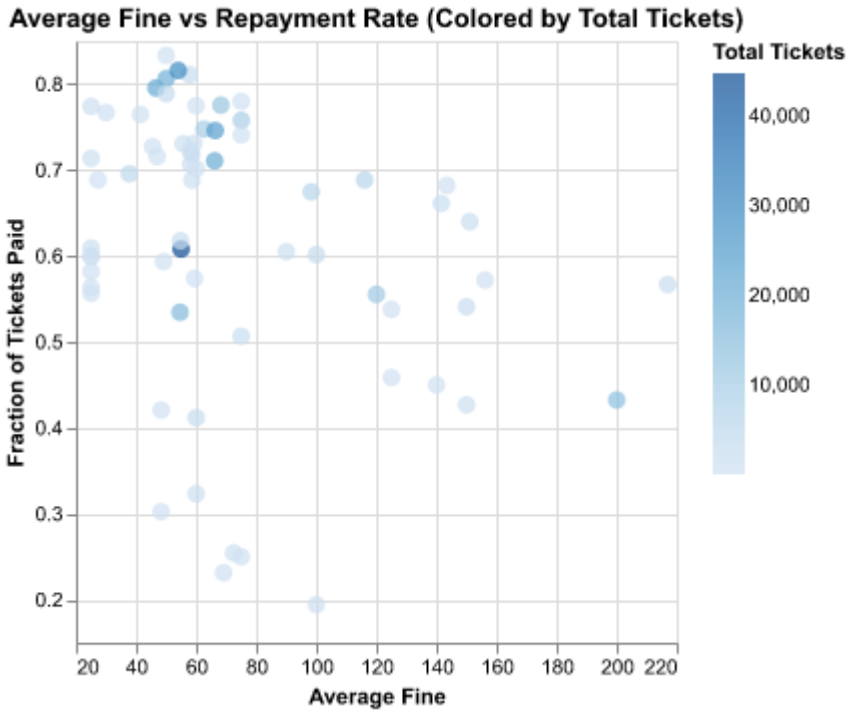
```
filtered_data = violation_summary[violation_summary['total_tickets'] >= 100]
outlier_fine = filtered_data['average_fine'].max()
filtered_data = filtered_data[filtered_data['average_fine'] != outlier_fine]

base = alt.Chart(filtered_data).encode(
    x=alt.X('average_fine:Q', title='Average Fine', scale=alt.Scale(zero=False)),
    y=alt.Y('repayment_rate:Q', title='Fraction of Tickets Paid', scale=alt.Scale(zero=False)
    tooltip=[
```

```
        alt.Tooltip('violation_description:N', title='Violation Type'),
        alt.Tooltip('average_fine:Q', title='Average Fine'),
        alt.Tooltip('repayment_rate:Q', title='Repayment Rate', format='.2%'),
        alt.Tooltip('total_tickets:Q', title='Total Tickets')
    ]
)

scatter_plot_1 = base.mark_circle(size=80).properties(
    title="Relationship Between Average Fine and Repayment Rate"
)
print("Graph 1:")
print("Headline: 'Correlation Between Fine Size and Ticket Payment Rate'")
print("Sub-message: There is a noticeable correlation where tickets with higher fines are les
scatter_plot_1.display()

scatter_plot_2 = base.mark_circle(size=80).encode(
    color=alt.Color('total_tickets:Q', scale=alt.Scale(scheme='blues'), title='Total Tickets
).properties(
    title="Average Fine vs Repayment Rate (Colored by Total Tickets)"
)
print("Graph 2:")
print("Headline: 'Payment Rate vs Fine Amount (Color-Coded by Ticket Count)'")
print("Sub-message: The color gradient indicates the number of tickets issued. Violations wi
scatter_plot_2.display()

scatter_plot_3 = base.mark_circle().encode(
    size=alt.Size('total_tickets:Q', title='Total Tickets', scale=alt.Scale(range=[20, 400])
).properties(
    title="Average Fine vs Repayment Rate (Point Size by Total Tickets)"
)
print("Graph 3:")
print("Headline: 'Payment Rate and Average Fine (Ticket Count Reflected by Point Size)'")
print("Sub-message: The size of each point corresponds to the ticket volume. More common vio
scatter_plot_3.display()
```

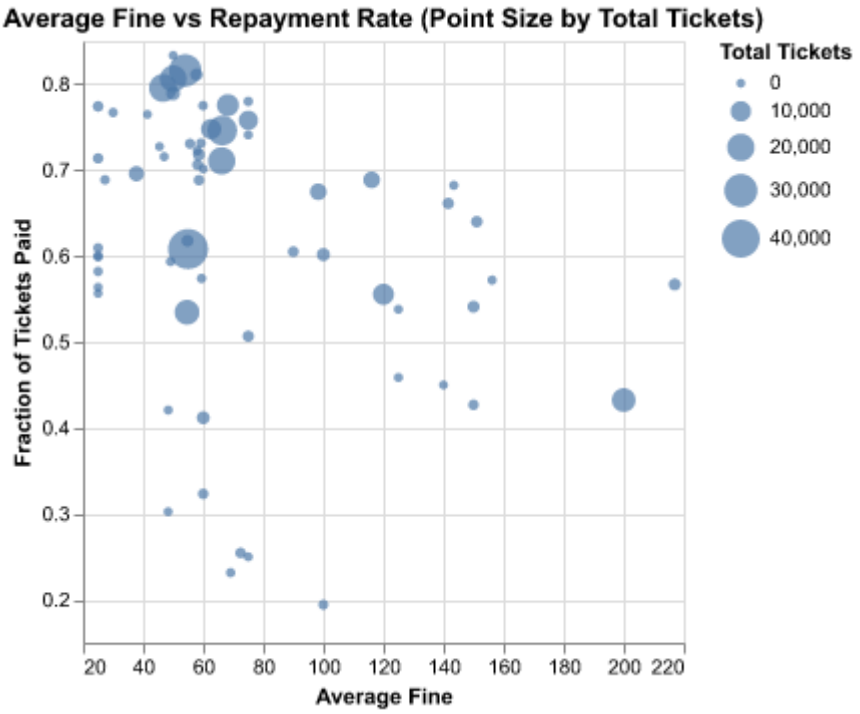
```
Graph 1:
Headline: 'Correlation Between Fine Size and Ticket Payment Rate'
Sub-message: There is a noticeable correlation where tickets with higher fines are less likel
```



Graph 2:
Headline: 'Payment Rate vs Fine Amount (Color-Coded by Ticket Count)'
Sub-message: The color gradient indicates the number of tickets issued. Violations with more



Graph 3:
Headline: 'Payment Rate and Average Fine (Ticket Count Reflected by Point Size)'
Sub-message: The size of each point corresponds to the ticket volume. More common violations



3.

Plot 3 ("Average Fine vs Repayment Rate (Point Size by Total Tickets)") is the best choice for the City Clerk. It effectively visualizes the relationship between fine amounts and payment rates using point size to represent ticket frequency, making it easy to interpret without needing statistical knowledge. Plot 3 balances clarity and detail by showing both the frequency of violations and the relationship between fines and payments in a simple, intuitive format.

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

1.

```
violation_summary = df.groupby('violation_description').agg(
    total_tickets=('ticket_number', 'size'),
    avg_fine_paid=('fine_level1_amount', 'mean'),
    avg_fine_unpaid=('fine_level2_amount', 'mean')
).reset_index()
```

```
violation_summary_filtered = violation_summary[violation_summary['total_tickets'] >= 100]

violation_summary_filtered['fine_doubles'] = violation_summary_filtered['avg_fine_unpaid'] >=

violations_not_doubling = violation_summary_filtered[violation_summary_filtered['fine_doubles

violations_not_doubling['increase_if_unpaid'] = ((violations_not_doubling['avg_fine_unpaid']

print(violations_not_doubling[['violation_description', 'total_tickets', 'avg_fine_paid', 'av
```

```
                    violation_description  total_tickets  avg_fine_paid  \
5       BLOCK ACCESS/ALLEY/DRIVEWAY/FIRELANE           1579     141.592780
15                     DISABLED PARKING ZONE           2034     216.986234
42  NO CITY STICKER VEHICLE OVER 16,000 LBS.            131     500.000000
54   OBSTRUCTED OR IMPROPERLY TINTED WINDOWS            271     156.180812
62                       PARK OR BLOCK ALLEY           2050     150.000000
79                PARK/STAND ON BICYCLE PATH            236     143.432203
95     SMOKED/TINTED WINDOWS PARKED/STANDING           1697     151.090159

    avg_fine_unpaid  increase_if_unpaid
5        266.751108           88.393157
15       358.308751           65.129716
42       955.343511           91.068702
54       225.645756           44.477259
62       259.926829           73.284553
79       278.601695           94.239291
95       209.516794           38.670047
```

2. ##4-2 Initial Stage: | v “VIOL” (Violation Issued) | |—> [Paid] —-> Process Ends | v “DETR” (Determination Notice) | |—> [Paid] —-> Process Ends | v “SEIZ” (Seizure Warning) | |—> [Paid] —-> Process Ends | v [Unpaid] —-> Escalates to Legal Action

---

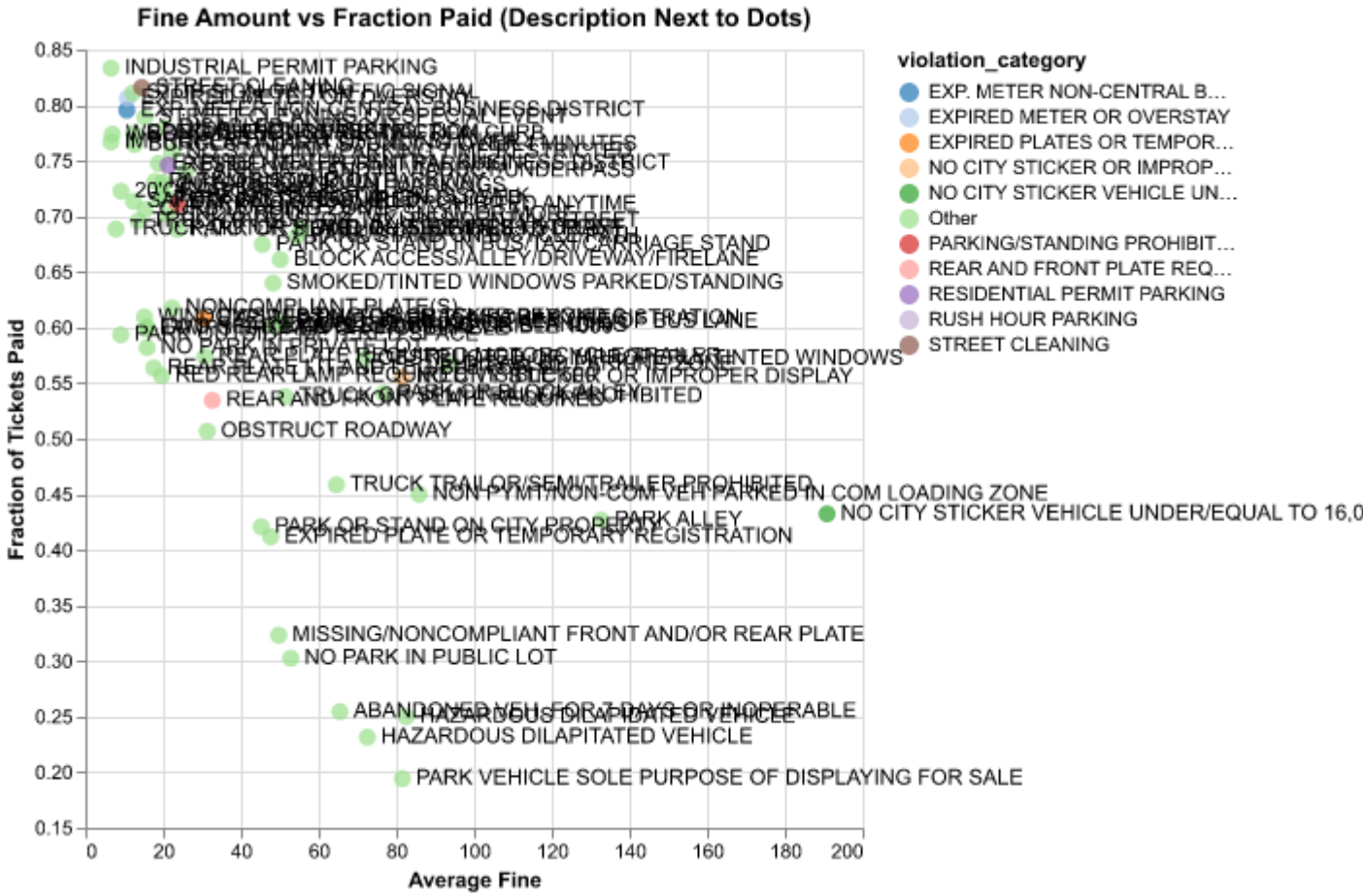
Contestation: | v [Contest Ticket] | |—> [Liable] —-> Continue Process (as unpaid) | |—> [Not Liable] —-> Ticket Dismissed

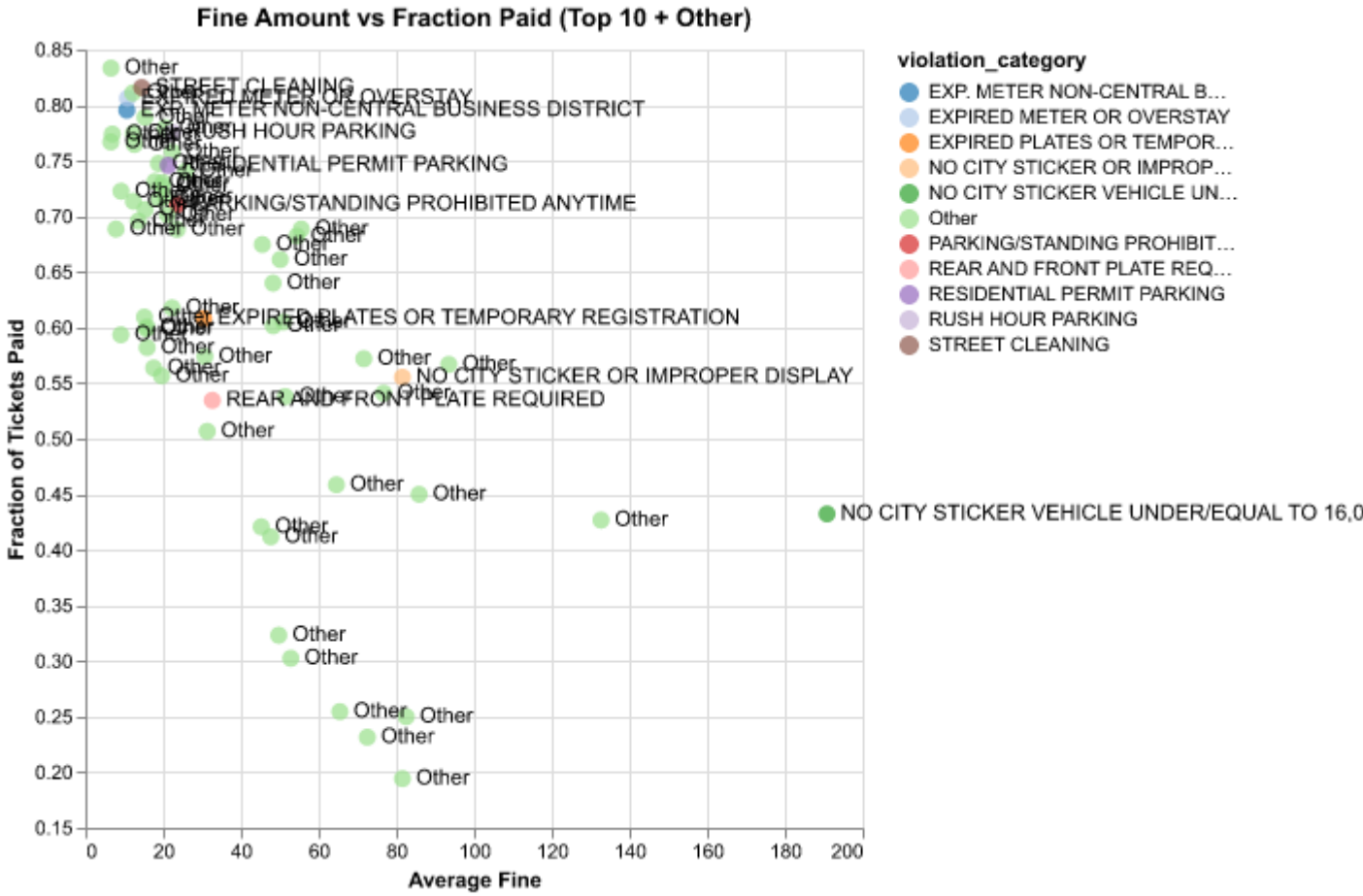
3.

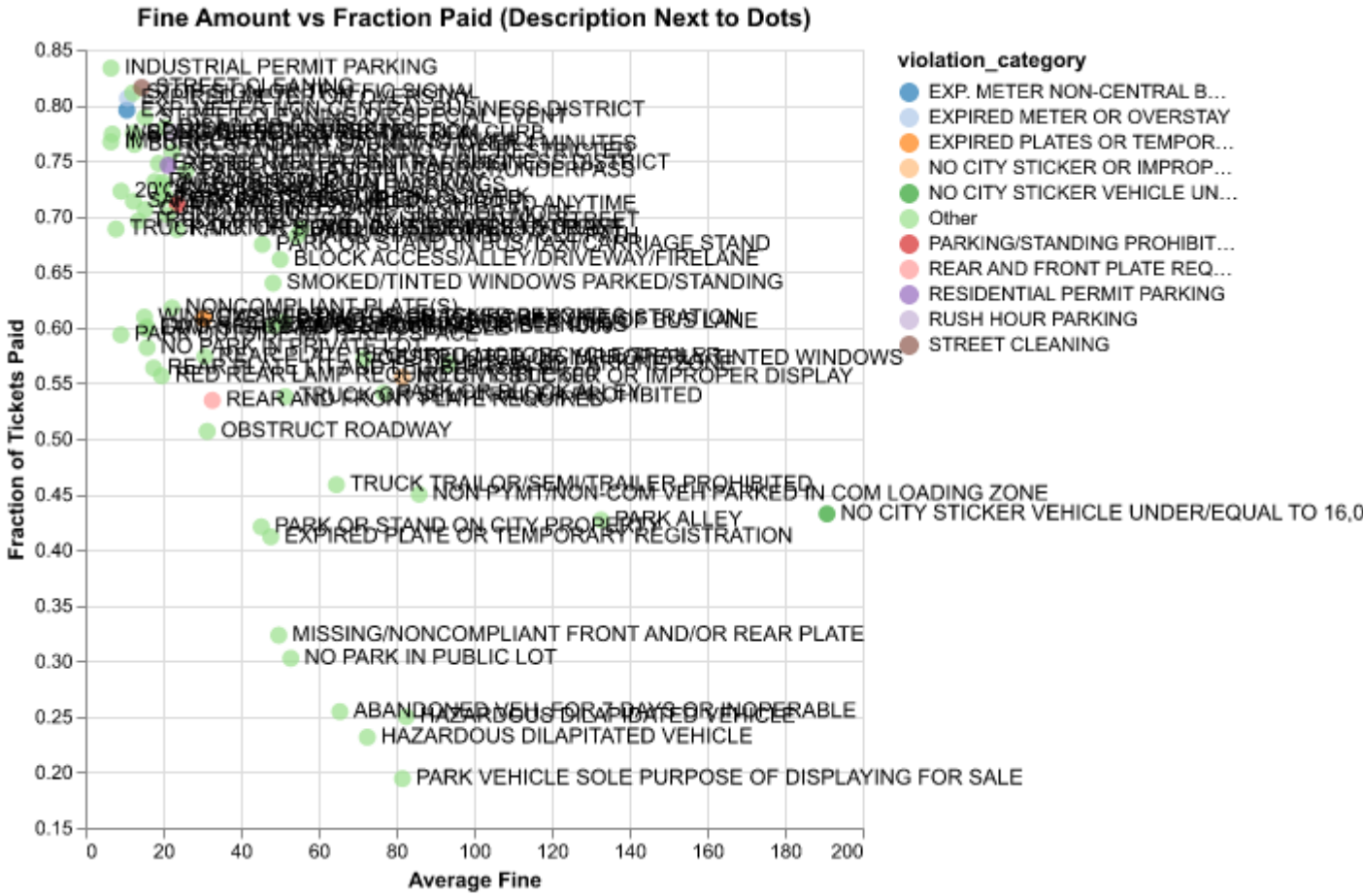
```
import pandas as pd
import altair as alt


violation_summary = df.groupby('violation_description').agg({
    'ticket_number': 'count',
    'current_amount_due': 'mean',
    'total_payments': lambda x: (x > 0).mean()
}).reset_index()

violation_summary.columns = ['violation_description', 'total_tickets', 'average_fine', 'fract


filtered_data = violation_summary[violation_summary['total_tickets'] >= 100]


outlier_fine = filtered_data['average_fine'].max()
filtered_data = filtered_data[filtered_data['average_fine'] != outlier_fine]


top_10_violations = filtered_data.nlargest(10, 'total_tickets')['violation_description'].toli


def categorize_violation(violation):
    if violation in top_10_violations:
        return violation
    return 'Other'

filtered_data['violation_category'] = filtered_data['violation_description'].apply(categorize


base = alt.Chart(filtered_data).encode(
    x=alt.X('average_fine:Q', title='Average Fine', scale=alt.Scale(zero=False)),
    y=alt.Y('fraction_paid:Q', title='Fraction of Tickets Paid', scale=alt.Scale(zero=False)
    tooltip=[
        alt.Tooltip('violation_description:N', title='Violation Type'),
        alt.Tooltip('average_fine:Q', title='Average Fine'),
        alt.Tooltip('fraction_paid:Q', title='Fraction Paid', format='.2%'),
        alt.Tooltip('total_tickets:Q', title='Total Tickets')
    ]
)
```

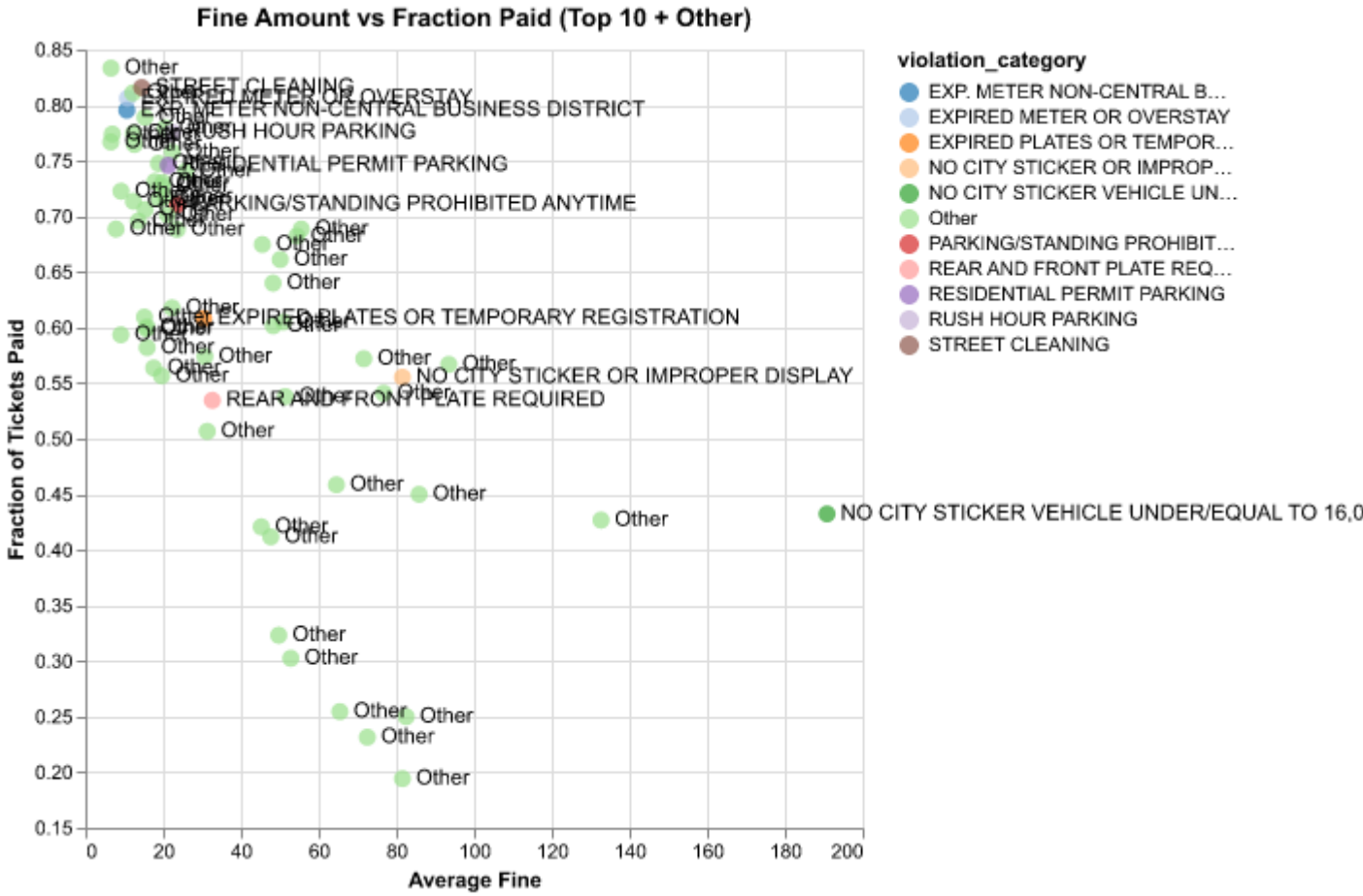
```
scatter_plot_with_description = base.mark_circle(size=80).encode(
    color=alt.Color('violation_category:N', scale=alt.Scale(scheme='category20'))
) + base.mark_text(align='left', dx=7).encode(
    text='violation_description:N'
)

scatter_plot_with_description = scatter_plot_with_description.properties(
    title="Fine Amount vs Fraction Paid (Description Next to Dots)",
    width=400,
    height=400
)


scatter_plot_with_description.display()



scatter_plot_with_top10_and_other = base.mark_circle(size=80).encode(
    color=alt.Color('violation_category:N', scale=alt.Scale(scheme='category20'))
) + base.mark_text(align='left', dx=7).encode(
    text=alt.condition(
        alt.datum.violation_category != 'Other',
        'violation_category:N',
        alt.value('Other')
    )
)

scatter_plot_with_top10_and_other = scatter_plot_with_top10_and_other.properties(
    title="Fine Amount vs Fraction Paid (Top 10 + Other)",
    width=400,
    height=400
)


scatter_plot_with_top10_and_other.display()



def categorize_violation_meaningful(violation):
    if 'PARKING' in violation:
        return 'Parking Violation'
    elif 'PERMIT' in violation:
```

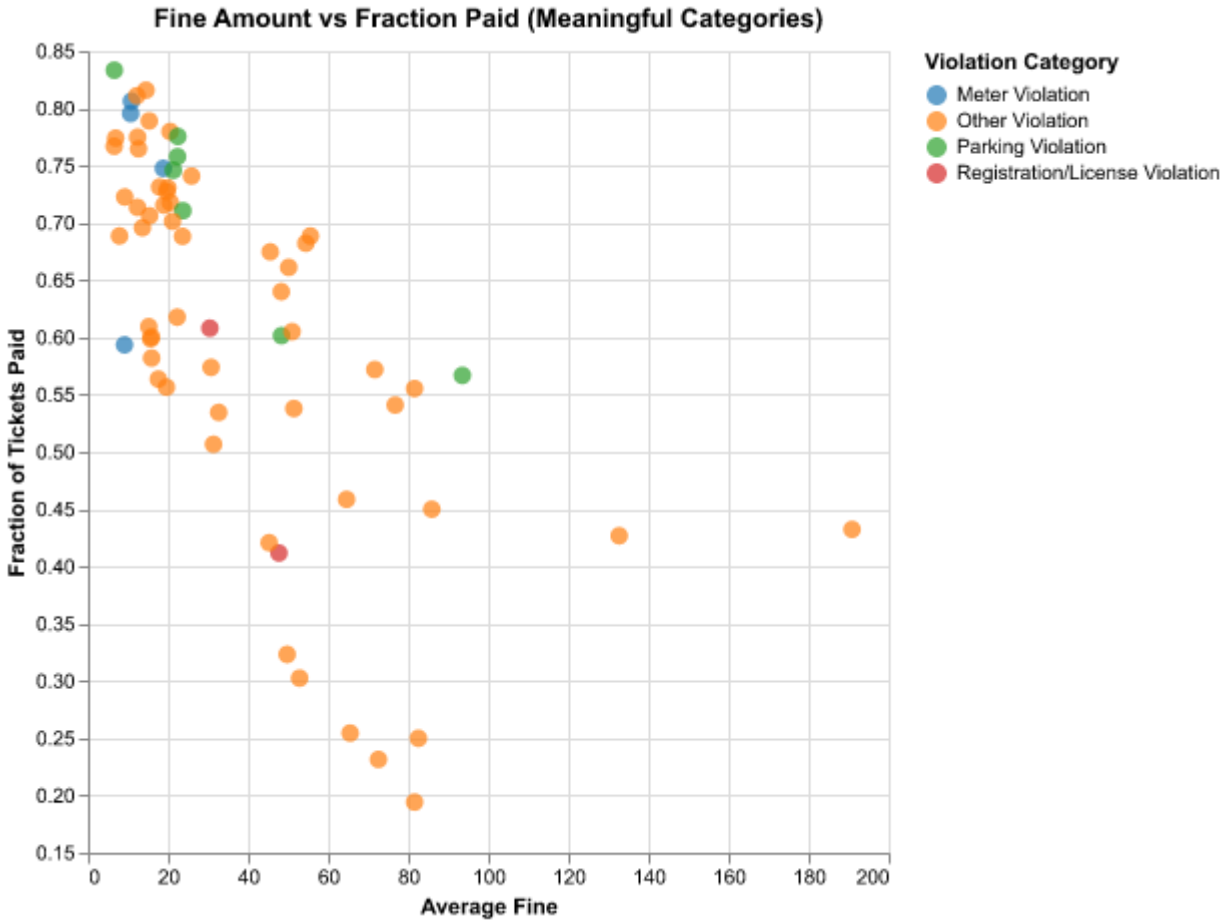
```
        return 'Permit Violation'
    elif 'METER' in violation:
        return 'Meter Violation'
    elif 'REGISTRATION' in violation or 'LICENSE' in violation:
        return 'Registration/License Violation'
    else:
        return 'Other Violation'

filtered_data['meaningful_category'] = filtered_data['violation_description'].apply(categoriz


scatter_plot_meaningful = base.mark_circle(size=80).encode(
    color=alt.Color('meaningful_category:N', scale=alt.Scale(scheme='category10'), legend=alt
)

scatter_plot_meaningful = scatter_plot_meaningful.properties(
    title="Fine Amount vs Fraction Paid (Meaningful Categories)",
    width=400,
    height=400
)


scatter_plot_with_description.display()
scatter_plot_with_top10_and_other.display()
scatter_plot_meaningful.display()
```


Fine Amount vs Fraction Paid (Description Next to Dots)
Fraction of Tickets Paid
Average Fine
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100
120
140
160
180
200
INDUSTRIAL PERMIT PARKING
BLOCK ACCESS/ALLEY/DRIVEWAY/FIRELANE
SMOKED/TINTED WINDOWS PARKED/STANDING
OBSTRUCT ROADWAY
TRUCK TRAILOR/SEMI-TRAILER PROHIBITED
NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,0
PARK ALLEY
EXPIRED PLATE OR TEMPORARY REGISTRATION
MISSING/NONCOMPLIANT FRONT AND/OR REAR PLATE
NO PARK IN PUBLIC LOT
HAZARDOUS DILAPITATED VEHICLE
PARK VEHICLE SOLE PURPOSE OF DISPLAYING FOR SALE
violation_category
EXP. METER NON-CENTRAL B...
EXPIRED METER OR OVERSTAY
EXPIRED PLATES OR TEMPOR...
NO CITY STICKER OR IMPROP...
NO CITY STICKER VEHICLE UN...
Other
PARKING/STANDING PROHIBIT...
REAR AND FRONT PLATE REQ...
RESIDENTIAL PERMIT PARKING
RUSH HOUR PARKING
STREET CLEANING


Fine Amount vs Fraction Paid (Top 10 + Other)
Fraction of Tickets Paid
Average Fine
violation_category
EXP. METER NON-CENTRAL B...
EXPIRED METER OR OVERSTAY
EXPIRED PLATES OR TEMPOR...
NO CITY STICKER OR IMPROP...
NO CITY STICKER VEHICLE UN...
Other
PARKING/STANDING PROHIBIT...
REAR AND FRONT PLATE REQ...
RESIDENTIAL PERMIT PARKING
RUSH HOUR PARKING
STREET CLEANING
EXPIRED METER NON-CENTRAL BUSINESS DISTRICT
RUSH HOUR PARKING
RESIDENTIAL PERMIT PARKING
PARKING/STANDING PROHIBITED ANYTIME
EXPIRED PLATES OR TEMPORARY REGISTRATION
NO CITY STICKER OR IMPROPER DISPLAY
REAR AND FRONT PLATE REQUIRED
NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,0


Fine Amount vs Fraction Paid (Description Next to Dots)
Fraction of Tickets Paid
Average Fine
violation_category
EXP. METER NON-CENTRAL B...
EXPIRED METER OR OVERSTAY
EXPIRED PLATES OR TEMPOR...
NO CITY STICKER OR IMPROP...
NO CITY STICKER VEHICLE UN...
Other
PARKING/STANDING PROHIBIT...
REAR AND FRONT PLATE REQ...
RESIDENTIAL PERMIT PARKING
RUSH HOUR PARKING
STREET CLEANING
INDUSTRIAL PERMIT PARKING
BLOCK ACCESS/ALLEY/DRIVEWAY/FIRELANE
SMOKED/TINTED WINDOWS PARKED/STANDING
OBSTRUCT ROADWAY
TRUCK TRAILOR/SEMI-TRAILER PROHIBITED
NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,0
PARK ALLEY
EXPIRED PLATE OR TEMPORARY REGISTRATION
MISSING/NONCOMPLIANT FRONT AND/OR REAR PLATE
NO PARK IN PUBLIC LOT
HAZARDOUS DILAPITATED VEHICLE
PARK VEHICLE SOLE PURPOSE OF DISPLAYING FOR SALE


Fine Amount vs Fraction Paid (Top 10 + Other)
Fraction of Tickets Paid
Average Fine
violation_category
EXP. METER NON-CENTRAL B...
EXPIRED METER OR OVERSTAY
EXPIRED PLATES OR TEMPOR...
NO CITY STICKER OR IMPROP...
NO CITY STICKER VEHICLE UN...
Other
PARKING/STANDING PROHIBIT...
REAR AND FRONT PLATE REQ...
RESIDENTIAL PERMIT PARKING
RUSH HOUR PARKING
STREET CLEANING
EXPIRED PLATES OR TEMPORARY REGISTRATION
NO CITY STICKER OR IMPROPER DISPLAY
REAR AND FRONT PLATE REQUIRED
NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,0



Extra Credit (max 5 points)

1.

```
violation_counts = df.groupby(['violation_code', 'violation_description']).size().reset_index

multiple_descriptions = violation_counts.groupby('violation_code').filter(lambda x: len(x) >

most_common_descriptions = multiple_descriptions.loc[multiple_descriptions.groupby('violation

df['most_common_description'] = df['violation_code'].map(most_common_descriptions.set_index(

top_codes = multiple_descriptions.groupby('violation_code')['count'].sum().nlargest(3).index
print("Top three codes with multiple descriptions:")
print(multiple_descriptions[multiple_descriptions['violation_code'].isin(top_codes)])
```

```
Top three codes with multiple descriptions:
     violation_code                              violation_description  count
4          0964040B                                      STREET CLEANING  28712
5          0964040B                  STREET CLEANING OR SPECIAL EVENT   3370
77         0976160A       MISSING/NONCOMPLIANT FRONT AND/OR REAR PLATE   1024
78         0976160A                         REAR AND FRONT PLATE REQUIRED  15829
113  964125_0964125B                NO CITY STICKER OR IMPROPER DISPLAY  10758
114  964125_0964125B  NO CITY STICKER VEHICLE UNDER/EQUAL TO 16,000 ...  14246
```

2. ##6-2 { “$schema”: “https://vega.github.io/schema/vega/v5.json”, “description”: “A custom case progression tree.”, “width”: 600, “height”: 800, “padding”: 5,

“data”: [{ “name”: “tree”, “values”: [{“id”: “Initial Stage”, “parent”: null}, {“id”: “VIOL (Violation Issued)”, “parent”: “Initial Stage”}, {“id”: “Paid_1”, “parent”: “VIOL (Violation Issued)”}, {“id”: “Process Ends_1”, “parent”: “Paid_1”}, {“id”: “DETR (Determination Notice)”, “parent”: “VIOL (Violation Issued)”}, {“id”: “Paid_2”, “parent”: “DETR (Determination Notice)”}, {“id”: “Process Ends_2”, “parent”: “Paid_2”}, {“id”: “SEIZ (Seizure Warning)”, “parent”: “DETR (Determination Notice)”}, {“id”: “Paid_3”, “parent”: “SEIZ (Seizure Warning)”}, {“id”: “Process Ends_3”, “parent”: “Paid_3”}, {“id”: “Unpaid”, “parent”: “SEIZ (Seizure Warning)”}, {“id”: “Escalates to Legal Action”, “parent”: “Unpaid”}, {“id”: “Contestation”, “parent”: “Initial Stage”}, {“id”: “Contest Ticket”, “parent”: “Contestation”}, {“id”: “Liable”, “parent”: “Contest Ticket”}, {“id”: “Continue Process (as unpaid)”, “parent”: “Liable”}, {“id”: “Not Liable”, “parent”: “Contest Ticket”}, {“id”: “Ticket Dismissed”, “parent”: “Not Liable”}], “transform”: [{ “type”: “stratify”, “key”: “id”, “parentKey”: “parent” }, { “type”: “tree”, “method”: “tidy”, “size”: [{“signal”: “height”}, {“signal”: “width - 100”}], “separation”: true, “as”: [“y”, “x”, “depth”, “children”] }] }, { “name”: “links”, “source”: “tree”, “transform”: [{“type”: “treelinks”}, { “type”: “linkpath”, “orient”: “horizontal”, “shape”: “diagonal” }] }],

“scales”: [{ “name”: “color”, “type”: “linear”, “range”: {“scheme”: “category20”}, “domain”: {“data”: “tree”, “field”: “depth”}, “zero”: true }],

“marks”: [{ “type”: “path”, “from”: {“data”: “links”}, “encode”: { “update”: { “path”: {“field”: “path”}, “stroke”: {“value”: “#ccc”} } } }, { “type”: “symbol”, “from”: {“data”: “tree”}, “encode”: { “enter”: { “size”: {“value”: 100}, “stroke”: {“value”: “#fff”} }, “update”: { “x”: {“field”: “x”}, “y”: {“field”: “y”}, “fill”: {“scale”: “color”, “field”: “depth”} } } }, { “type”: “text”, “from”: {“data”: “tree”}, “encode”: { “enter”: { “text”: {“field”: “id”}, “fontSize”: {“value”: 9}, “baseline”: {“value”: “middle”} }, “update”: { “x”: {“field”: “x”}, “y”: {“field”: “y”}, “dx”: {“signal”: “datum.children ? -7 : 7”}, “align”: {“signal”: “datum.children ? ‘right’ : ‘left’ ”} } } }] }