```
In [14]: import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [31]: df=pd.read_csv('C:\\Users\\Lenovo\\Downloads\\cfpb_data_with_clean_zip.csv')
         df.shape
```

```
Out[31]: (2036509, 21)
```

```
In [32]: # Convert 'Date.received' to datetime format
         df['Date.received'] = pd.to_datetime(df['Date.received'], format='%m/%d/%y')

         # Format 'Date.received' to 'mm-dd-yyyy'
         df['Date.received'] = df['Date.received'].dt.strftime('%m-%d-%Y')
         # Replace periods in column names with spaces
         df.columns = df.columns.str.replace('.', ' ', regex=False)

         print(df.columns)
```

```
Index(['Date received', 'Product', 'Sub product', 'Issue', 'Sub issue',
       'Consumer complaint narrative', 'Company public response', 'Company',
       'State', 'ZIP code', 'Tags', 'Consumer consent provided ',
       'Submitted via', 'Date sent to company', 'Company response to consumer',
       'Timely response ', 'Consumer disputed ', 'Complaint ID', 'zip_best',
       'zip_three_best', 'zip_state_best'],
      dtype='object')
```

```
In [33]: df.columns
```

```
Out[33]: Index(['Date received', 'Product', 'Sub product', 'Issue', 'Sub issue',
       'Consumer complaint narrative', 'Company public response', 'Company',
       'State', 'ZIP code', 'Tags', 'Consumer consent provided ',
       'Submitted via', 'Date sent to company', 'Company response to consumer',
       'Timely response ', 'Consumer disputed ', 'Complaint ID', 'zip_best',
       'zip_three_best', 'zip_state_best'],
      dtype='object')
```

```
In [34]: df_population = pd.read_csv('C:\\Users\\Lenovo\\Downloads\\census_data_by_zip.csv', encoding='IS
```

```
In [35]: df_population.shape
```

```
Out[35]: (33774, 20)
```

```
In [36]: df_population.columns
```

```
Out[36]: Index(['zip', 'pop', 'pop_moe', 'pop_white', 'pop_black_or_aa', 'pop_ai_or_an',
       'pop_asian', 'pop_nh_or_opi', 'pop_other', 'pop_multiple', 'pop_hol',
       'pop_not_hol', 'income_cnt_households', 'income_cnt_households_moe',
       'income_cnt_households_with_earnings',
       'income_cnt_households_with_earnings_moe',
       'income_cnt_households_with_pub_assist',
       'income_cnt_households_with_pub_assist_moe',
       'income_mean_household_dollars', 'income_mean_household_dollars_moe'],
      dtype='object')
```

```
In [37]: df['zip_best'] = df['zip_best'].astype(str)
         df_population['zip'] = df_population['zip'].astype(str)
         #Join complaints and census data
         df_joined = pd.merge(df, df_population, left_on='zip_best', right_on='zip', how='inner')

         df_joined.shape
```

```
Out[37]: (1800487, 41)
```

```
In [38]: df_joined.columns
```

```
Out[38]: Index(['Date received', 'Product', 'Sub product', 'Issue', 'Sub issue',
               'Consumer complaint narrative', 'Company public response', 'Company',
               'State', 'ZIP code', 'Tags', 'Consumer consent provided ',
               'Submitted via', 'Date sent to company', 'Company response to consumer',
               'Timely response ', 'Consumer disputed ', 'Complaint ID', 'zip_best',
               'zip_three_best', 'zip_state_best', 'zip', 'pop', 'pop_moe',
               'pop_white', 'pop_black_or_aa', 'pop_ai_or_an', 'pop_asian',
               'pop_nh_or_opi', 'pop_other', 'pop_multiple', 'pop_hol', 'pop_not_hol',
               'income_cnt_households', 'income_cnt_households_moe',
               'income_cnt_households_with_earnings',
               'income_cnt_households_with_earnings_moe',
               'income_cnt_households_with_pub_assist',
               'income_cnt_households_with_pub_assist_moe',
               'income_mean_household_dollars', 'income_mean_household_dollars_moe'],
              dtype='object')
```

```
In [39]: df_joined.head(1)
```

Out[39]:

	Date received	Product	Sub product	Issue	Sub issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	...	pop_hol
0	12-07-2021	Mortgage	VA mortgage	Trouble during payment process	NaN	NaN	Company has responded to the consumer and the ...	WELLS FARGO & COMPANY	TX	78666	...	36770

1 rows × 41 columns

```
In [24]: #  unique values in each column
         unique_values = df_joined.nunique()

         print(unique_values)
```

```
Date received                                 1491
Product                                          9
Sub product                                     48
Issue                                           81
Sub issue                                      166
Consumer complaint narrative                598272
Company public response                         10
Company                                       4934
State                                           50
ZIP code                                     20020
Tags                                             3
Consumer consent provided                        4
Submitted via                                    7
Date sent to company                          1560
Company response to consumer                     5
Timely response                                  2
Consumer disputed                                0
Complaint ID                               1800487
zip_best                                     18980
zip_three_best                                 808
zip_state_best                                  44
zip                                          18980
pop                                          13878
pop_moe                                       2912
pop_white                                    12324
pop_black_or_aa                               5324
pop_ai_or_an                                  1420
pop_asian                                     3530
pop_nh_or_opi                                  591
pop_other                                     4227
pop_multiple                                  5165
pop_hol                                       6169
pop_not_hol                                  13347
income_cnt_households                        10054
income_cnt_households_moe                     1177
income_cnt_households_with_earnings           9019
income_cnt_households_with_earnings_moe       1130
income_cnt_households_with_pub_assist         3498
income_cnt_households_with_pub_assist_moe      434
income_mean_household_dollars                16748
income_mean_household_dollars_moe            11107
dtype: int64
```

In [25]:
```
# Find the number of non-null values in each column
not_null_counts = df_joined.count()

print(not_null_counts)
```

```
Date received                                  1800487
Product                                        1800487
Sub product                                    1800394
Issue                                          1800487
Sub issue                                      1663744
Consumer complaint narrative                    724558
Company public response                         947499
Company                                        1800487
State                                          1800130
ZIP code                                       1800487
Tags                                            168966
Consumer consent provided                      1656267
Submitted via                                  1800487
Date sent to company                           1800487
Company response to consumer                   1800486
Timely response                                1800487
Consumer disputed                                    0
Complaint ID                                   1800487
zip_best                                       1800487
zip_three_best                                 1800487
zip_state_best                                 1800487
zip                                            1800487
pop                                            1800487
pop_moe                                        1800409
pop_white                                      1800487
pop_black_or_aa                                1800487
pop_ai_or_an                                   1800487
pop_asian                                      1800487
pop_nh_or_opi                                  1800487
pop_other                                      1800487
pop_multiple                                   1800487
pop_hol                                        1800487
pop_not_hol                                    1800487
income_cnt_households                          1797075
income_cnt_households_moe                      1797075
income_cnt_households_with_earnings            1797075
income_cnt_households_with_earnings_moe        1797075
income_cnt_households_with_pub_assist          1797075
income_cnt_households_with_pub_assist_moe      1797075
income_mean_household_dollars                  1791428
income_mean_household_dollars_moe              1791428
dtype: int64
```

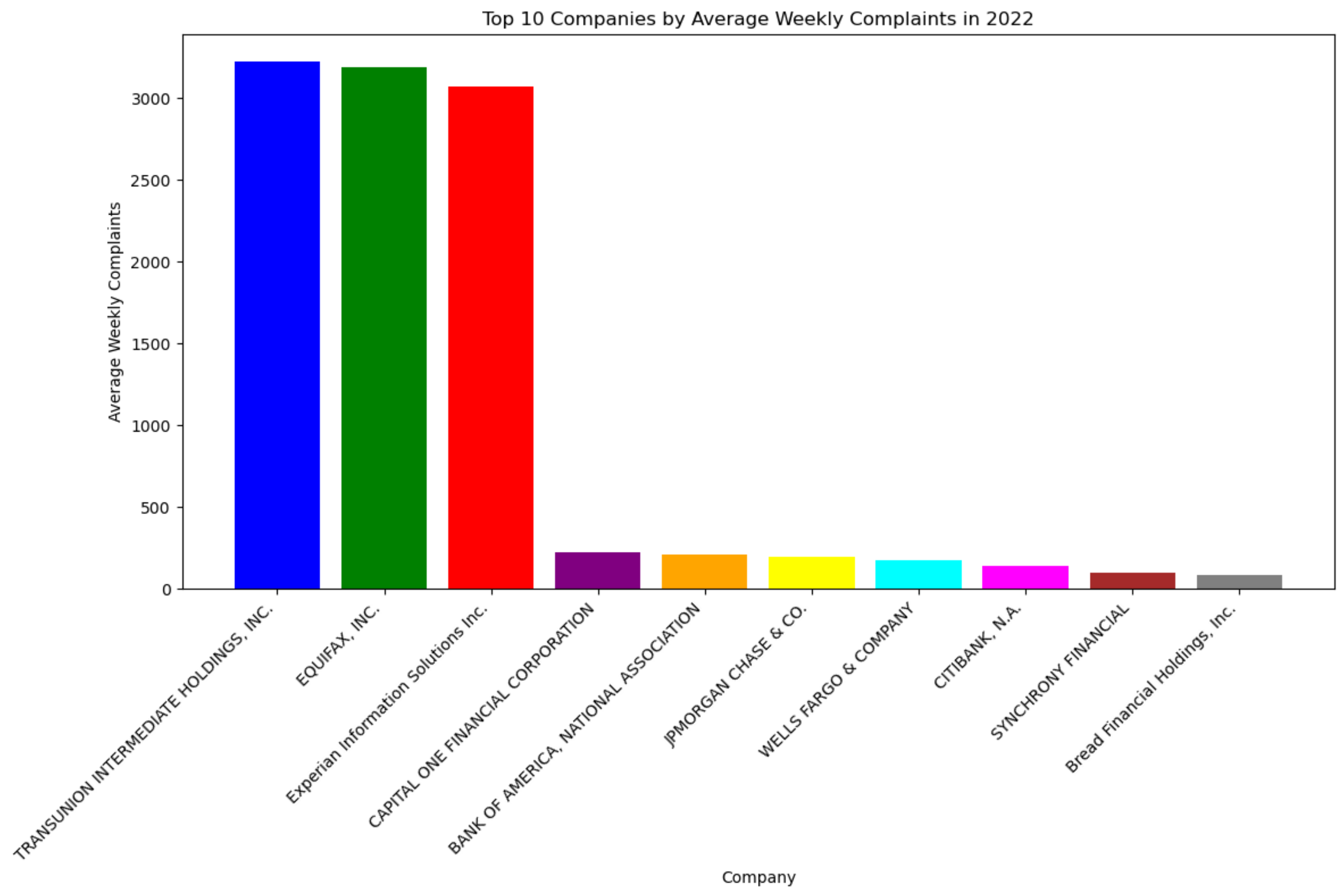
TOP 10 COMPAINES WITH HIGHEST AVERAGE WEEKLY COMPLAINT COUNT 2022

In [52]:

```
# DataFrame for the year 2022
df_2022 = df_joined[df_joined.index.year == 2022]

# Group by company and count complaints weekly
weekly_complaints_2022 = df_2022.groupby('Company').resample('W')['Complaint ID'].nunique().rese

# Average weekly complaint count for each company
average_weekly_complaints_2022 = weekly_complaints_2022.groupby('Company')['Weekly Complaints'].

# Sorting, ordering and get the top 10
top_companies = average_weekly_complaints_2022.sort_values(by='Average Weekly Complaints', ascen

print(top_companies)
```

```
                                  Company  Average Weekly Complaints
2775  TRANSUNION INTERMEDIATE HOLDINGS, INC.                3227.000000
1044                           EQUIFAX, INC.                3190.339623
1108      Experian Information Solutions Inc.               3068.471698
512         CAPITAL ONE FINANCIAL CORPORATION                222.962264
367     BANK OF AMERICA, NATIONAL ASSOCIATION                210.981132
1567                      JPMORGAN CHASE & CO.                198.943396
3033                     WELLS FARGO & COMPANY                178.415094
554                             CITIBANK, N.A.                142.716981
2552                      SYNCHRONY FINANCIAL                  96.509434
469             Bread Financial Holdings, Inc.                 85.698113
```

In [53]:
```python
import matplotlib.pyplot as plt

colors = ['blue', 'green', 'red', 'purple', 'orange', 'yellow', 'cyan', 'magenta', 'brown', 'gray

plt.figure(figsize=(12, 8))
plt.bar(top_companies['Company'], top_companies['Average Weekly Complaints'], color=colors)
plt.title('Top 10 Companies by Average Weekly Complaints in 2022')
plt.xlabel('Company')
plt.ylabel('Average Weekly Complaints')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Weekly Complaints Counts 2019-2023

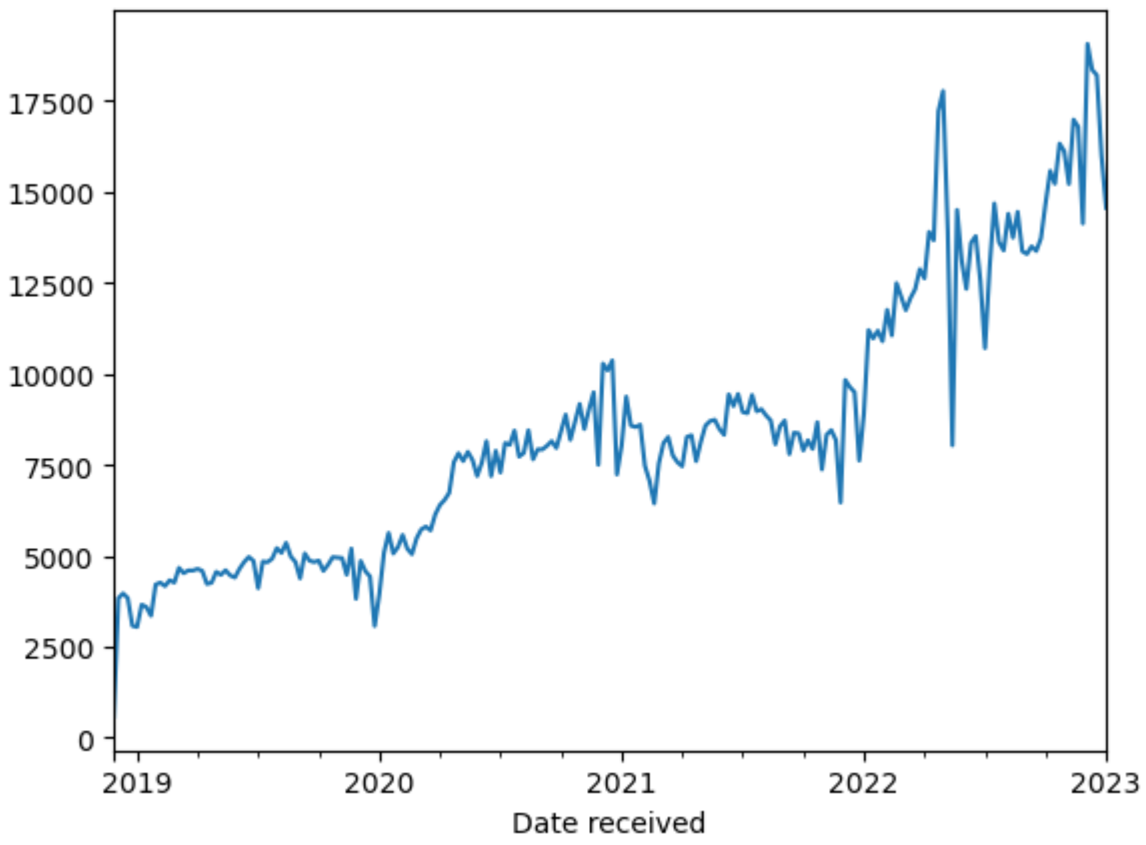
In [67]:
```python
# Resample data to WEEKLY counts of complaints
weekly_complaints = df_joined['Complaint ID'].resample('W').count()
```

In [68]:
```python
weekly_complaints.shape
```

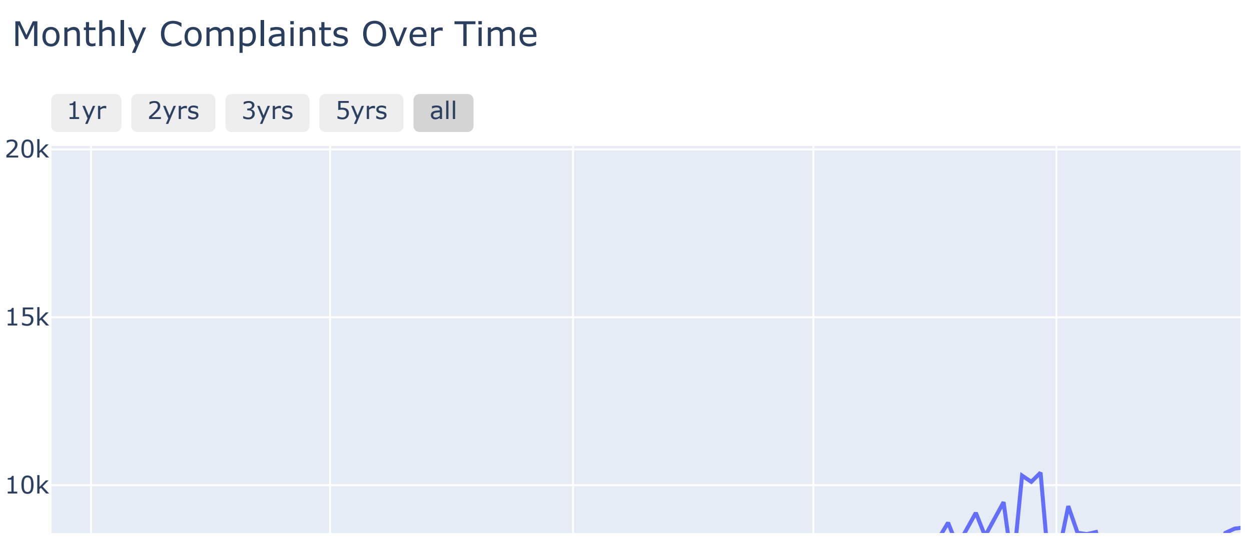
Out[68]:
```
(214,)
```

In [69]:
```python
weekly_complaints.plot(kind='line')
```

Out[69]:
```
<Axes: xlabel='Date received'>
```



In [70]:
```python
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Scatter(x=weekly_complaints.index, y=weekly_complaints, name="Complaints"),
    secondary_y=False,
)

# Add slider
fig.update_layout(
    title="Monthly Complaints Over Time",
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=12, label="1yr", step="month", stepmode="backward"),
                dict(count=24, label="2yrs", step="month", stepmode="backward"),
                dict(count=36, label="3yrs", step="month", stepmode="backward"),
                dict(count=60, label="5yrs", step="month", stepmode="backward"),
                dict(step="all")
            ])
        ),
        type="date"
    )
)

# Show plot
fig.show()
```



Average Weekly Complaints count for each year

```
In [57]: import pandas as pd



         # Calculate average weekly complaints for each year
         def calculate_average_weekly_complaints(df):
             grouped = df.groupby('Company')

             def resample_fill(group):
                 return group.resample('W')['Complaint ID'].nunique().fillna(0)

             weekly_complaints = grouped.apply(resample_fill).reset_index(name='Weekly Complaints')

             average_weekly_complaints = weekly_complaints.groupby('Company')['Weekly Complaints'].mean()

             return average_weekly_complaints

         # Create an empty DataFrame
         results = pd.DataFrame(columns=['Year', 'Company', 'Average Weekly Count'])

         # Loop through each year
         for year in range(df_joined.index.year.min(), df_joined.index.year.max() + 1):
             df_year = df_joined[df_joined.index.year == year]

             # Calculate  weekly complaints for the current year
             average_weekly_complaints_year = calculate_average_weekly_complaints(df_year)

             average_weekly_complaints_year['Year'] = year

             # Append the results to the overall DataFrame
             results = pd.concat([results, average_weekly_complaints_year[['Year', 'Company', 'Average We

         # Save the results to a CSV file
         results.to_csv('C:\\Users\\Lenovo\\Downloads\\average_weekly_complaints_by_company.csv', index=Fa

         print("Results saved to average_weekly_complaints_by_company.csv")
```

```
Results saved to average_weekly_complaints_by_company.csv
```

In [58]:

```python
import pandas as pd


# Calculate average weekly complaints for each year
def calculate_average_weekly_complaints(df):
    # Group by company and count complaints weekly
    grouped = df.groupby('Company')

    def resample_fill(group):
        return group.resample('W')['Complaint ID'].nunique().fillna(0)

    # Apply the resampling and filling function to each group
    weekly_complaints = grouped.apply(resample_fill).reset_index(name='Weekly Complaints')

    # Average weekly complaint count for each company
    average_weekly_complaints = weekly_complaints.groupby('Company')['Weekly Complaints'].mean()

    return average_weekly_complaints

#  empty DataFrame to store the results
results = pd.DataFrame(columns=['Year', 'Company', 'Average Weekly Complaints'])

# Loop through each year
for year in range(df_joined.index.year.min(), df_joined.index.year.max() + 1):
    df_year = df_joined[df_joined.index.year == year]

    average_weekly_complaints_year = calculate_average_weekly_complaints(df_year)

    sorted_companies = average_weekly_complaints_year.sort_values(by='Average Weekly Complaints'

    # Select only the top 20 companies
    top_20_companies = sorted_companies.head(20)

    top_20_companies.loc[:, 'Year'] = year

    results = pd.concat([results, top_20_companies[['Year', 'Company', 'Average Weekly Complaint

# Save the results to a CSV file
results.to_csv('C:\\Users\\Lenovo\\Downloads\\average_weekly_complaints_top_20_by_company.csv', 

print("Results saved to average_weekly_complaints_top_20_by_company.csv")
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5940\653320612.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  top_20_companies.loc[:, 'Year'] = year
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5940\653320612.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  top_20_companies.loc[:, 'Year'] = year
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5940\653320612.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  top_20_companies.loc[:, 'Year'] = year
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5940\653320612.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  top_20_companies.loc[:, 'Year'] = year
Results saved to average_weekly_complaints_top_20_by_company.csv
```

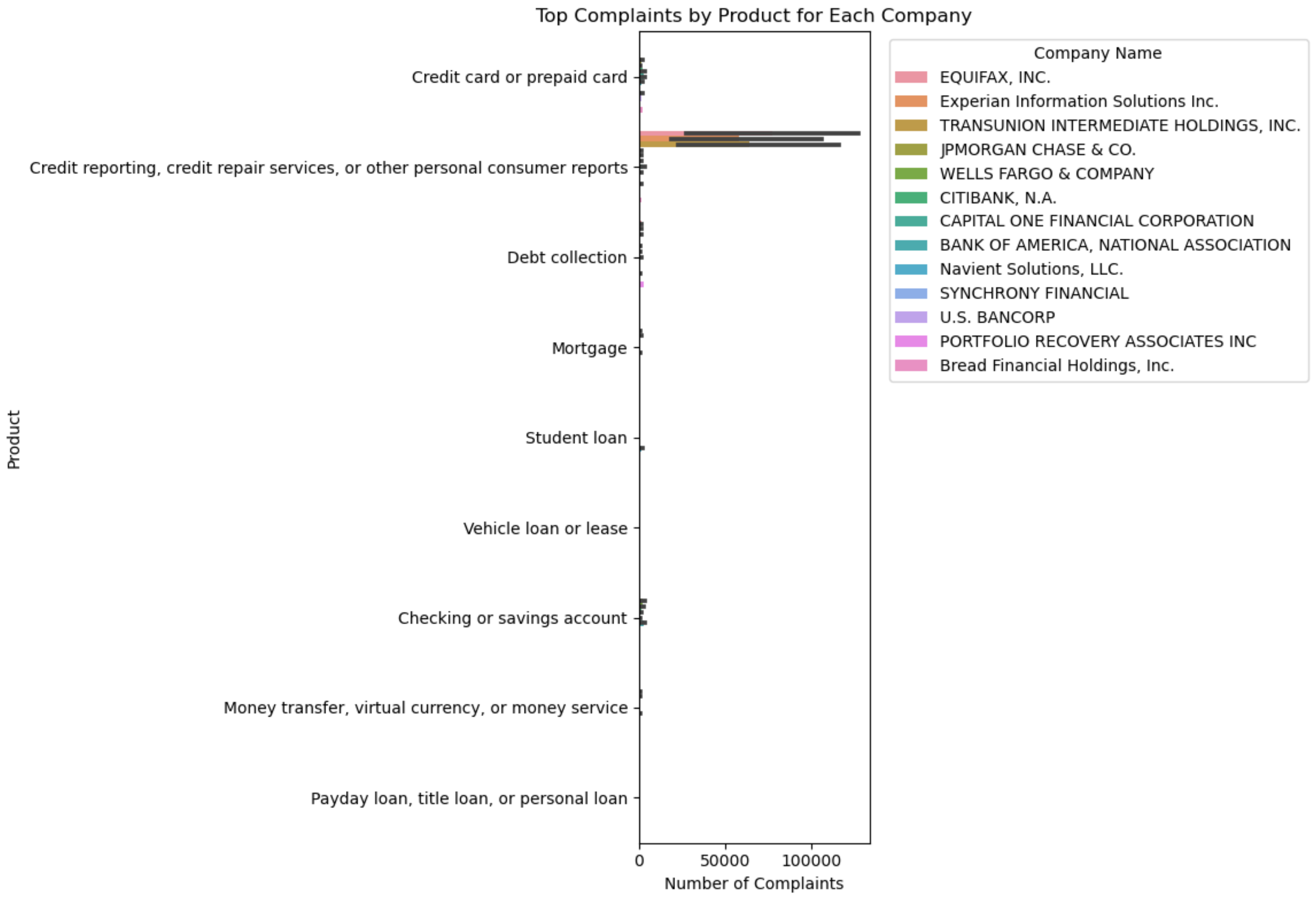
```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5940\653320612.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  top_20_companies.loc[:, 'Year'] = year
```

TOP 10 COMPANIES WITH HIGHEST COMPLAINT COUNT EACH YEAR AND FOR THOSE COMPANIES , COUNT BY "Product"

In [62]:
```
import pandas as pd


# Calculate total complaints and complaints by product for top 10 companies each year
def top_companies_complaints_by_product(df):
    results = []

    for year in range(df.index.year.min(), df.index.year.max() + 1):
        df_year = df[df.index.year == year]
        total_complaints_year = df_year.groupby('Company').size().reset_index(name='Total Compla
        top_companies = total_complaints_year.nlargest(10, 'Total Complaints')

        for company in top_companies['Company']:
            df_company = df_year[df_year['Company'] == company]
            product_counts = df_company.groupby('Product').size().reset_index(name='Product Compl
            product_counts['Year'] = year
            product_counts['Company Name'] = company
            product_counts['Total Complaint Count'] = top_companies[top_companies['Company'] == 
            product_counts = product_counts[['Year', 'Company Name', 'Total Complaint Count', 'P
            results.append(product_counts)

    final_results = pd.concat(results, ignore_index=True)
    return final_results

# Now that 'Date received' is correctly set as the index, calling the function
results_df = top_companies_complaints_by_product(df_joined)

# Save the results to a CSV file
results_df.to_csv('C:\\Users\\Lenovo\\Downloads\\top_companies_complaints_by_product.csv', index

print("Results saved to top_companies_complaints_by_product.csv")
```

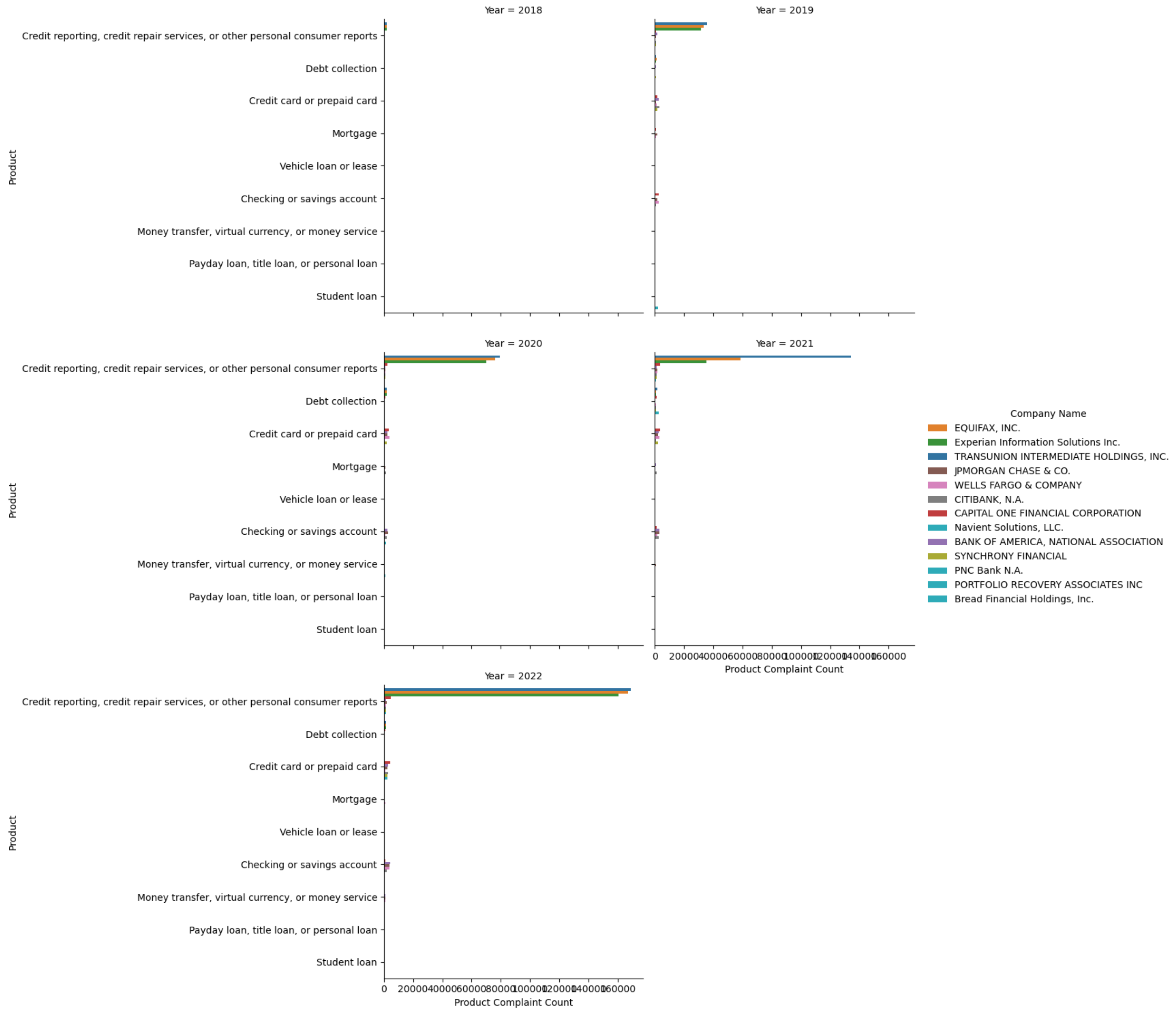
```
Results saved to top_companies_complaints_by_product.csv
```

In [140…
```
import matplotlib.pyplot as plt
import seaborn as sns


# Plot a bar chart for the top complaints by product for each company
plt.figure(figsize=(12, 8))
sns.barplot(x='Product Complaint Count', y='Product', hue='Company Name', data=results_df)
plt.title('Top Complaints by Product for Each Company')
plt.xlabel('Number of Complaints')
plt.ylabel('Product')
plt.legend(title='Company Name', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



In [63]:
```
g = sns.FacetGrid(results_df, col="Year", col_wrap=2, height=5, aspect=1.5)
g.map(sns.barplot, 'Product Complaint Count', 'Product', 'Company Name', palette='tab10', order=
g.add_legend(title='Company Name')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:717: UserWarning: Using the barpl
ot function without specifying `hue_order` is likely to produce an incorrect plot.
  warnings.warn(warning)
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layo
ut has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



TOP 10 COMPANIES WITH HIGHEST COMPLAINT COUNT EACH YEAR AND FOR THOSE COMPANIES , COUNT BY "Company response to consumer" -

```
In [64]: import pandas as pd

         # Total complaints and complaints by company response for top 10 companies each year
         def top_companies_response_by_product(df):
             results = []

             for year in range(df.index.year.min(), df.index.year.max() + 1):
                 df_year = df[df.index.year == year]
                 total_complaints_year = df_year.groupby('Company').size().reset_index(name='Total Compla
                 top_companies = total_complaints_year.nlargest(10, 'Total Complaints')

                 for company in top_companies['Company']:
                     df_company = df_year[df_year['Company'] == company]
                     response_counts = df_company.groupby('Company response to consumer').size().reset_in
                     response_counts['Year'] = year
                     response_counts['Company Name'] = company
                     response_counts['Total Complaint Count'] = top_companies[top_companies['Company'] ==
                     response_counts = response_counts[['Year', 'Company Name', 'Total Complaint Count',
                     results.append(response_counts)

             final_results = pd.concat(results, ignore_index=True)
             return final_results

         # Calculate the top companies and response counts
         results_df = top_companies_response_by_product(df_joined)

         # Save the results to a CSV file
         results_df.to_csv('C:\\Users\\Lenovo\\Downloads\\top_companies_responses_by_consumer.csv', index

         print("Results saved to top_companies_responses_by_consumer.csv")
```

```
Results saved to top_companies_responses_by_consumer.csv
```

Identify the top 15 states with the highest complaint count for each year.

For these states, categorize the zip codes based on the percentage contribution to the state's total complaints (100-75%, 75-50%, 50-25%, and 0-25%).

Calculate the average income_mean_household_dollars_moe for zip codes in each category.

```
In [65]: import pandas as pd
         import numpy as np

         def analyze_population_segments_by_complaints(df):
             results = []

             for year in range(df.index.year.min(), df.index.year.max() + 1):
                 df_year = df[df.index.year == year]

                 # Step 1: Identify top 15 states by total complaints for the year
                 state_complaints = df_year.groupby('State').size().reset_index(name='Total Complaints')
                 top_states = state_complaints.nlargest(15, 'Total Complaints')['State']

                 for state in top_states:
                     df_state = df_year[df_year['State'] == state]

                     # Step 2: Calculate complaint counts per zip and sort
                     zip_complaints = df_state.groupby('zip').size().reset_index(name='Complaints').sort_
                     total_complaints = zip_complaints['Complaints'].sum()
                     zip_complaints['Cumulative Percentage'] = zip_complaints['Complaints'].cumsum() / to

                     # Step 3: Categorize ZIP codes into percentile groups based on their contribution
                     zip_complaints['Percentile Group'] = pd.cut(zip_complaints['Cumulative Percentage'],
                                                                 bins=[0, 0.25, 0.5, 0.75, 1],
                                                                 labels=['75-100%', '50-75%', '25-50%', '
                                                                 right=False)

                     # Fetch unique income_mean_household_dollars for each zip
                     zip_income = df_state[['zip', 'income_mean_household_dollars']].drop_duplicates()

                     zip_complaints_income = pd.merge(zip_complaints, zip_income, on='zip', how='left')

                     # Step 4: Calculate average population segments for each percentile group
                     avg_income_by_group = zip_complaints_income.groupby('Percentile Group')['income_mean_

                     # Prepare and append results for this state
                     state_results = {
                         'Year': year,
                         'State': state,
                         'Total Complaint Count': total_complaints,
                     }
                     for group in ['75-100%', '50-75%', '25-50%', '0-25%']:
                         state_results[f'Avg Income {group}'] = avg_income_by_group.loc[avg_income_by_grou

                     results.append(state_results)

             return pd.DataFrame(results)

         results_df = analyze_population_segments_by_complaints(df_joined)


         # Save the results to a CSV file
         results_df.to_csv('C:\\Users\\Lenovo\\Downloads\\state_complaints_income_analysis.csv', index=Fa

         print("Results saved to state_complaints_income_analysis.csv")

         Results saved to state_complaints_income_analysis.csv
```

Population Segment Averages: The function now includes a loop over the specified population segment columns to calculate their averages for each percentile group of ZIP codes based on their contribution to the state's total complaints.

Dynamic Result Construction: For each state and year, the function dynamically constructs the result dictionary, including the average count for each population segment across the defined percentile groups (75-100%, 50-75%, 25-50%, and 0-25%).

```
In [66]: import pandas as pd

         def analyze_population_segments_by_complaints(df):
             results = []

             for year in range(df.index.year.min(), df.index.year.max() + 1):
                 df_year = df[df.index.year == year]

                 # Count complaints per state
                 state_complaints = df_year.groupby('State').size().reset_index(name='Total Complaints')
                 top_states = state_complaints.nlargest(15, 'Total Complaints')['State']

                 for state in top_states:
                     df_state = df_year[df_year['State'] == state]
                     zip_complaints = df_state.groupby('zip').size().reset_index(name='Complaints').sort_
                     total_complaints = zip_complaints['Complaints'].sum()
                     zip_complaints['Cumulative Percentage'] = zip_complaints['Complaints'].cumsum() / to

                     # Categorize ZIP codes into percentile groups based on their contribution
                     zip_complaints['Percentile Group'] = pd.cut(zip_complaints['Cumulative Percentage'],
                                                                 bins=[0, 0.25, 0.5, 0.75, 1],
                                                                 labels=['75-100%', '50-75%', '25-50%', '(
                                                                 right=False)

                     # Merge to associate each zip with its population segments
                     population_columns = ['pop_white', 'pop_black_or_aa', 'pop_ai_or_an', 'pop_asian',
                                           'pop_nh_or_opi', 'pop_other', 'pop_multiple', 'pop_hol', 'pop_
                     zip_population = df_state[['zip'] + population_columns].drop_duplicates()
                     zip_complaints_population = pd.merge(zip_complaints, zip_population, on='zip', how='

                     # Calculate average population segments for each percentile group
                     state_results = {
                         'Year': year,
                         'State': state,
                         'Total Complaint Count': total_complaints
                     }

                     for segment in population_columns:
                         avg_population_by_group = zip_complaints_population.groupby('Percentile Group')[

                         for group in ['75-100%', '50-75%', '25-50%', '0-25%']:
                             state_results[f'Avg {segment} {group}'] = avg_population_by_group.loc[avg_pop

                     results.append(state_results)

             return pd.DataFrame(results)

         # Calculate the population segment averages and complaint contributions
         results_df = analyze_population_segments_by_complaints(df_joined)

         # Save the results to a CSV file
         results_df.to_csv('C:\\Users\\Lenovo\\Downloads\\state_complaints_population_segments_analysis.c

         print("Results saved to state_complaints_population_segments_analysis.csv")

         Results saved to state_complaints_population_segments_analysis.csv

In [ ]:

In [ ]:

In [ ]:
```

In [71]:
```
#pip install statsmodels
```
In [ ]:
```
```