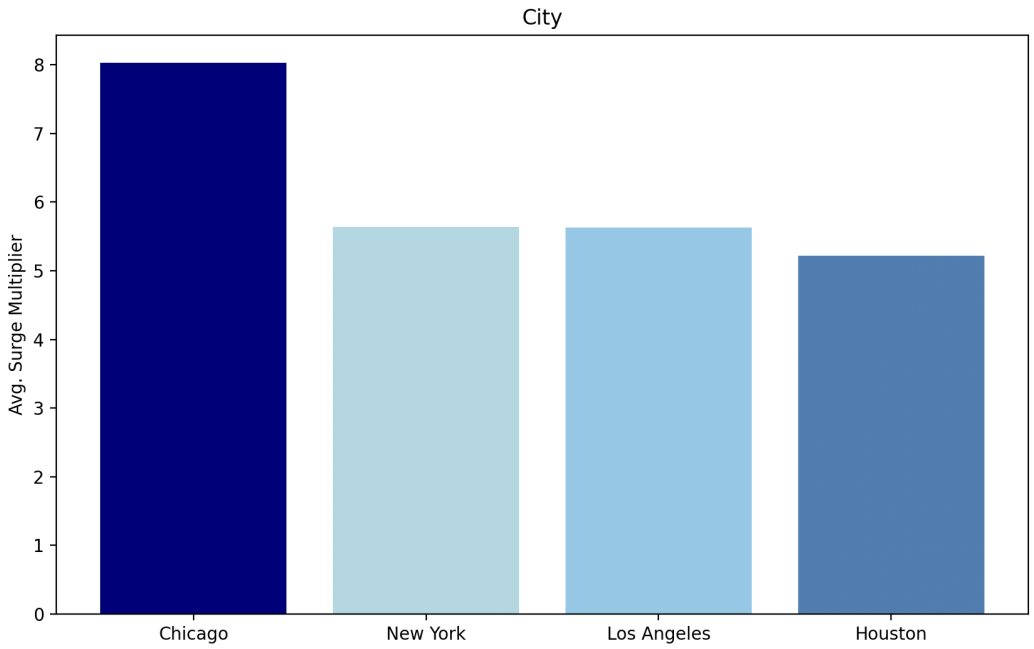
Libby Ford

Professor Frimpong

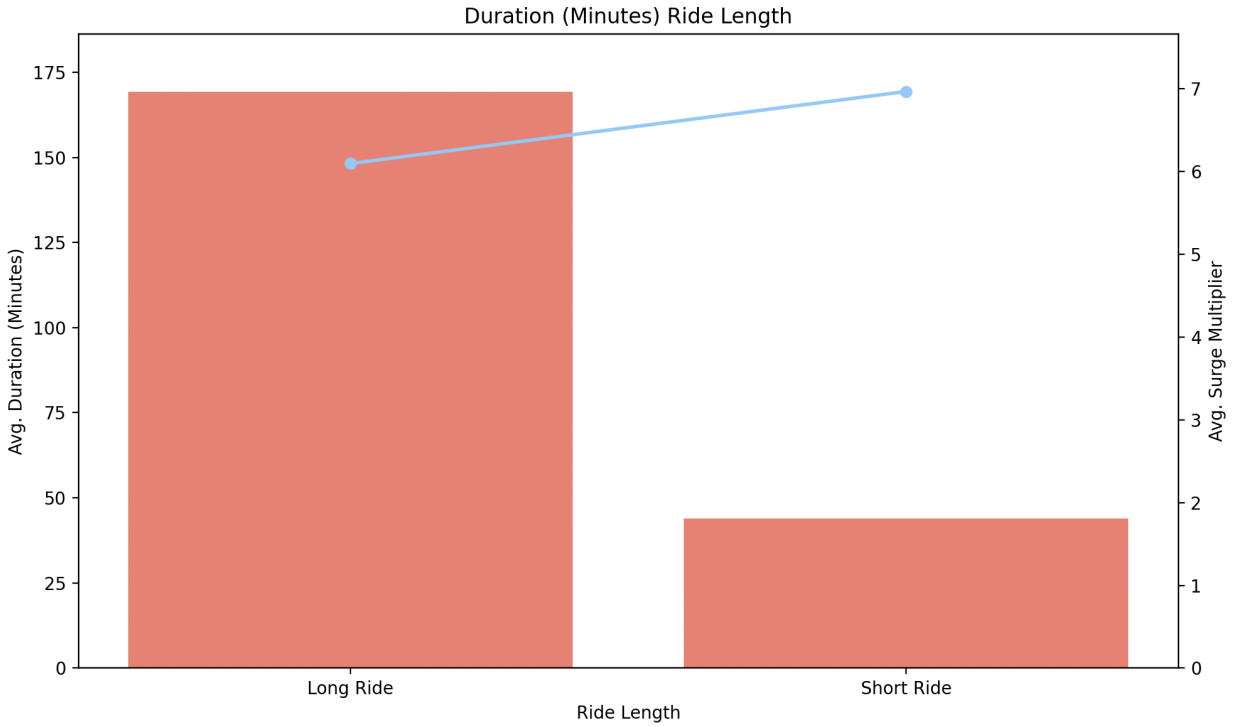
BUS 306F: Data Visualization

13 April 2025

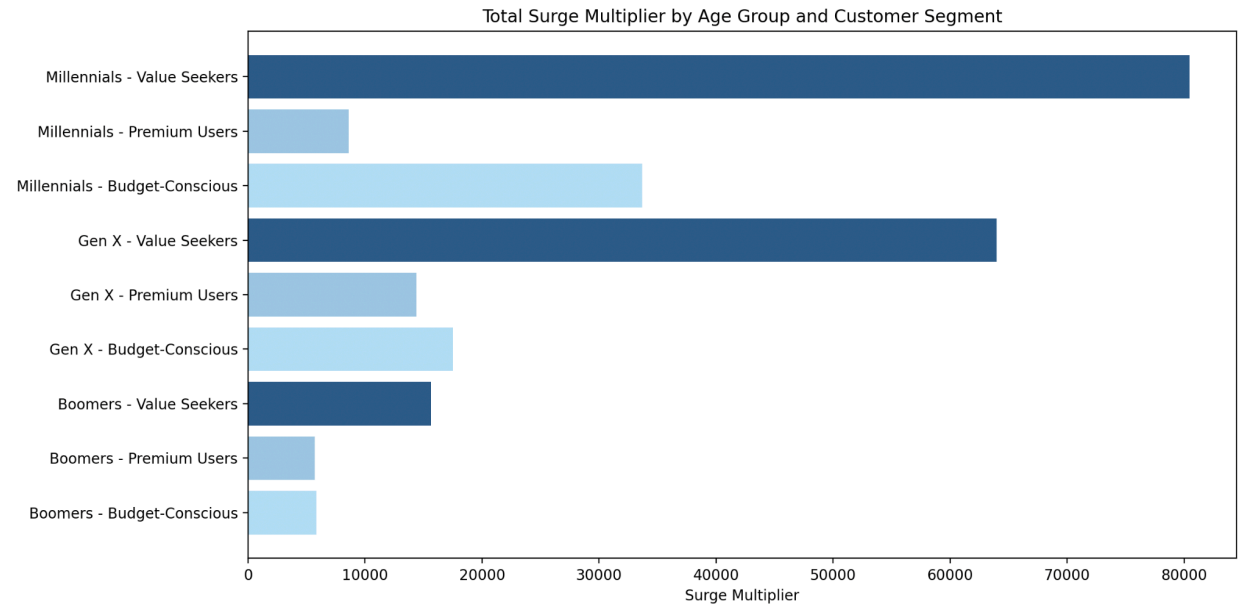
Python Replication Exam



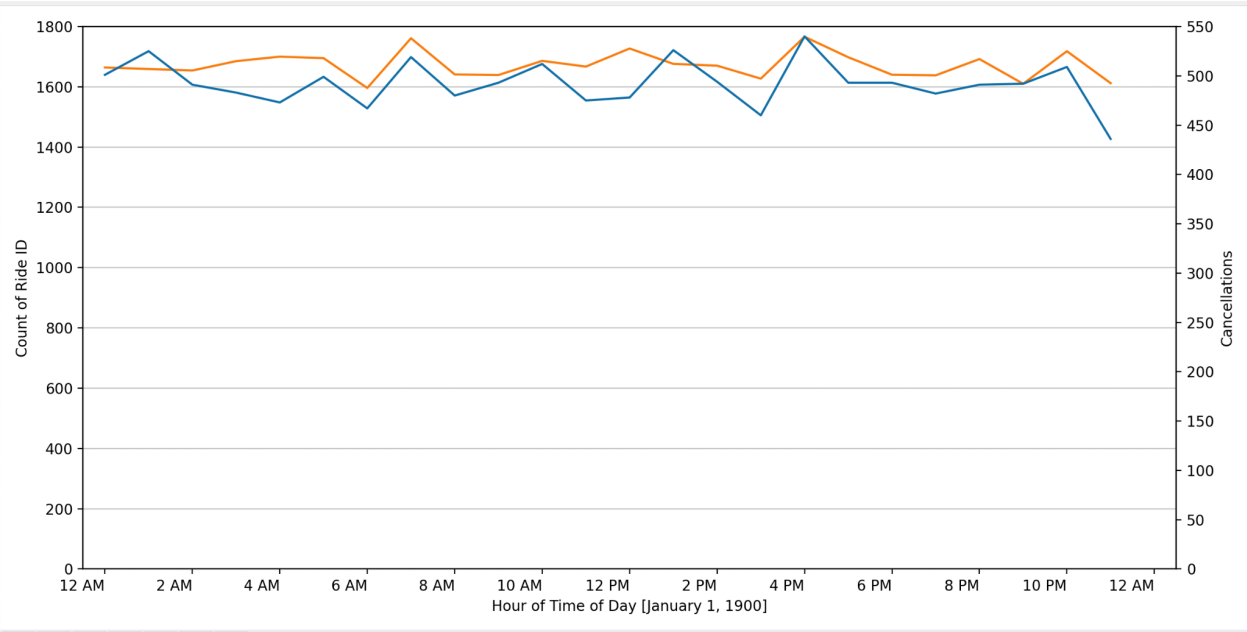

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv("/Users/libbyford/Desktop/Uber Data.csv")

# Calculate average surge multiplier per city
avg_surge_by_city = df.groupby('City')['SurgeMultiplier'].mean().sort_values(ascending=False)

# Define custom colors similar to the image
colors = ['navy', 'lightblue', 'skyblue', 'steelblue'][:len(avg_surge_by_city)]

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(avg_surge_by_city.index, avg_surge_by_city.values, color=colors)
plt.ylabel('Avg. Surge Multiplier')
plt.title('City')
plt.show()
```




```python
import pandas as pd
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv("/Users/libbyford/Desktop/Uber Data.csv")

# Create ride length categories
df['Ride Length'] = df['Duration (Minutes)'].apply(lambda x: 'Long Ride' if x >= 60 else 'Short Ride')

# Group and calculate averages
ride_summary = df.groupby('Ride Length').agg({
    'Duration (Minutes)': 'mean',
    'SurgeMultiplier': 'mean'
}).reset_index()

# Plot
fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for Avg. Duration
ax1.bar(ride_summary['Ride Length'], ride_summary['Duration (Minutes)'], color='#FA8072')
ax1.set_ylabel('Avg. Duration (Minutes)')
ax1.set_xlabel('Ride Length')
ax1.set_title('Duration (Minutes) Ride Length')
ax1.set_ylim(0, ride_summary['Duration (Minutes)'].max() * 1.1)

# Line plot for Avg. Surge Multiplier
ax2 = ax1.twinx()
ax2.plot(ride_summary['Ride Length'], ride_summary['SurgeMultiplier'], color='#87CEFA', marker='o', linewidth=2)
ax2.set_ylabel('Avg. Surge Multiplier')
ax2.set_ylim(0, ride_summary['SurgeMultiplier'].max() * 1.1)

plt.tight_layout()
plt.show()
```



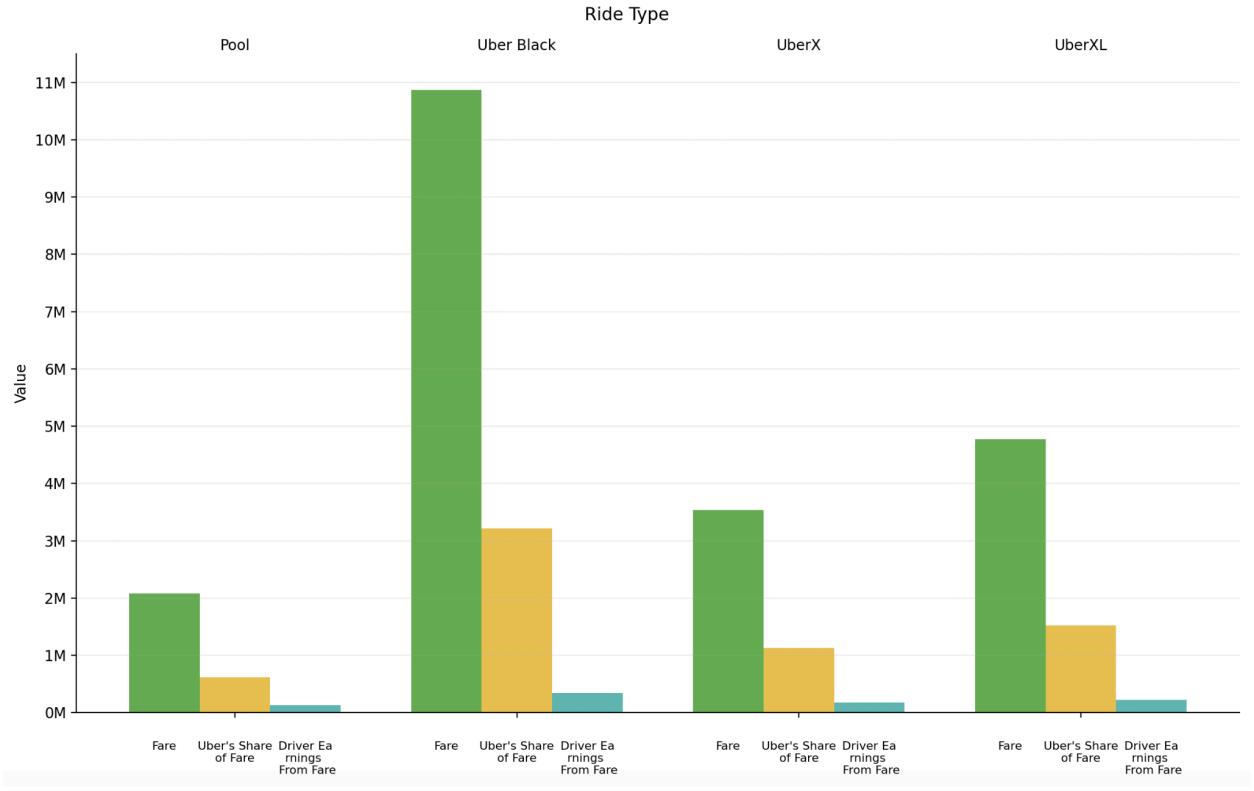

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv("/Users/libbyford/Desktop/Uber Data.csv")

# Aggregate total surge multiplier by Age Group and Customer Segment
grouped = df.groupby(['AgeGroup', 'CustomerSegment'])['SurgeMultiplier'].sum().reset_index()

# Sort Age Groups for display order
age_order = ['Millennials', 'Gen X', 'Boomers']
grouped['AgeGroup'] = pd.Categorical(grouped['AgeGroup'], categories= age_order, ordered=True)

# Assign color by Customer Segment
color_map = {
    'Value Seekers': '#2c5a89',
    'Budget-Conscious': '#b3dcf4',
    'Premium Users': '#9bc8e2'
}
grouped['Customer'] = grouped['CustomerSegment'].map(color_map)

# Create a label for each row to use on the y-axis
grouped['Label'] = grouped['AgeGroup'].astype(str) + " - " + grouped['CustomerSegment']

# Sort the labels for better readability (optional)
#grouped = grouped.sort_values(by=['AgeGroup', 'CustomerSegment'])

# Plot
plt.figure(figsize=(12, 6))
plt.barh(grouped['Label'], grouped['SurgeMultiplier'], color=grouped['Customer'])
plt.xlabel('Surge Multiplier')
plt.title('Total Surge Multiplier by Age Group and Customer Segment')
plt.tight_layout()
plt.show()
```




```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you've already read in your CSV file
df = pd.read_csv('/Users/libbyford/Desktop/Uber Data.csv')

# Extract the hour from the 'Time of Day' column
df['Hour'] = df['Time of Day'].str.split(':').str[0].astype(int)

# Group by hour and count rides
rides_by_hour = df.groupby('Hour').size().reset_index(name='Ride Count')

# Group by hour and sum cancellations
cancellations_by_hour = df.groupby('Hour')['Cancellations'].sum().reset_index()

# Create a figure with two y-axes
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plot ride counts on the left y-axis
color1 = '#fc7d0b'
ax1.set_xlabel('Hour of Time of Day [January 1, 1900]')
ax1.set_ylabel('Count of Ride ID')
ax1.plot( *args: rides_by_hour['Hour'], rides_by_hour['Ride Count'], color=color1)
ax1.tick_params(axis='y')

# Set y-axis limits for rides
ride_min = 0
ride_max = max(rides_by_hour['Ride Count']) * 1.1  # Add 10% padding
ax1.set_ylim(ride_min, ride_max)

# Create the second y-axis for cancellations
ax2 = ax1.twinx()
color2 = '#1170aa'
```

```
    ax2.set_ylabel('Cancellations')
    ax2.plot(cancellations_by_hour['Hour'], cancellations_by_hour['Cancellations'], color=color2)
    ax2.tick_params(axis='y')

    # Calculate the scale factor between rides and cancellations
    scale_factor = ride_max / (max(cancellations_by_hour['Cancellations']) * 1.1)

    # Set y-axis limits for cancellations to match the proportion of the rides axis
    cancel_min = 0
    cancel_max = max(cancellations_by_hour['Cancellations']) * 1.1
    ax2.set_ylim(cancel_min, cancel_max)

    # Format x-axis with AM/PM
    hour_ticks = list(range(0, 24, 2)) + [24]  # This adds 24 to the end of the list
    hour_labels = [f"{h if h < 12 else h-12} {'AM' if h < 12 or h == 24 else 'PM'}"
                   if h != 0 and h != 12 else f"{'12 AM' if h == 0 else '12 PM'}"
                   for h in hour_ticks[:-1]] + ['12 AM']

    # Extend the x-axis limit
    ax1.set_xlim(-0.5, 24.5)
    ax1.set_xticks(hour_ticks)
    ax1.set_xticklabels(hour_labels, ha='right')

    # Set y-axis limits and ticks for Ride ID (left axis)
    ride_min = 0
    ride_max = 1800
    ride_ticks = range(0, ride_max + 1, 200)
    ax1.set_ylim(ride_min, ride_max)
    ax1.set_yticks(ride_ticks)

    # Set y-axis limits and ticks for Cancellations (right axis)
    cancel_min = 0
    cancel_max = 550
    cancel_ticks = range(0, cancel_max + 1, 50)
```

```
# Set y-axis limits and ticks for Cancellations (right axis)
cancel_min = 0
cancel_max = 550
cancel_ticks = range(0, cancel_max + 1, 50)
ax2.set_ylim(cancel_min, cancel_max)
ax2.set_yticks(cancel_ticks)

# Add gridlines
ax1.grid(True, axis = 'y', linestyle='-', alpha=0.7)

fig.tight_layout()
plt.show()
```

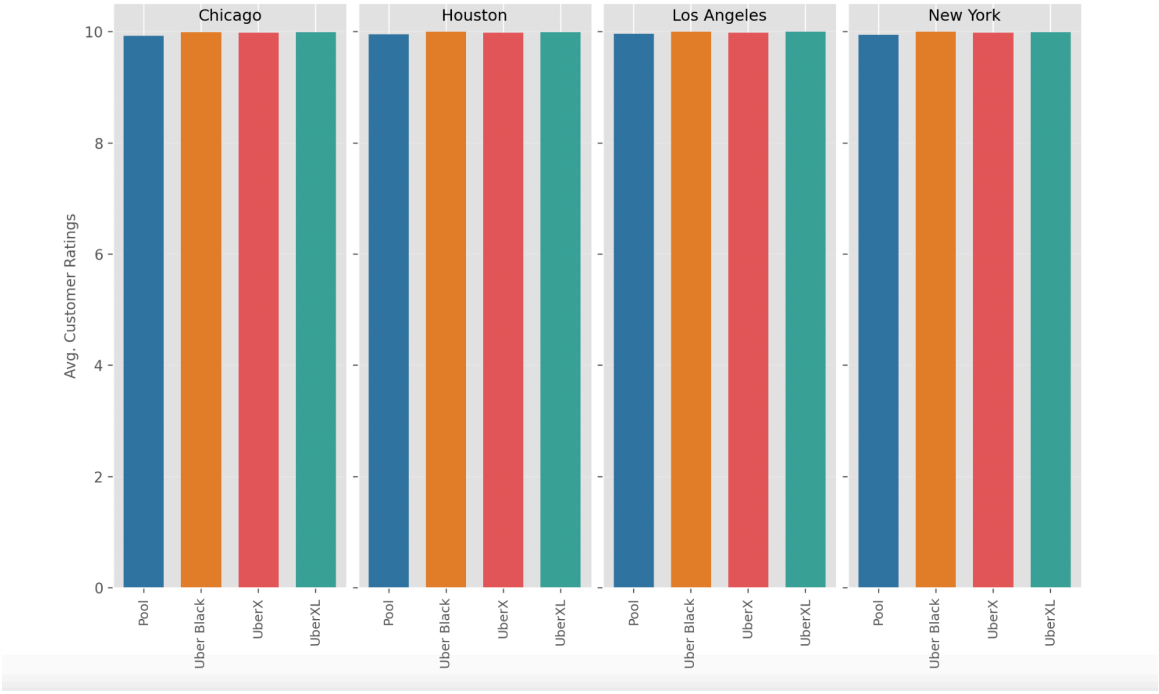



```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the data
df = pd.read_csv('/Users/libbyford/Desktop/Uber Data.csv')

# Group by RideType and calculate sums
ride_summary = df.groupby('RideType').agg({
    'Fare': 'sum',
    'Uber\'s Share of Fare': 'sum',
    'Driver Earnings from Fare': 'sum'
}).reset_index()

# Define the order of ride types to match the chart
ride_order = ['Pool', 'Uber Black', 'UberX', 'UberXL']
ride_summary = ride_summary.set_index('RideType').loc[ride_order].reset_index()

# Set up the figure and axes
fig, ax = plt.subplots(figsize=(12, 8))

# Define x-positions for the bars within each ride type group
x = np.arange(len(ride_order))
width = 0.25  # width of the bars

# Define colors to match the chart
colors = {
    'Fare': '#64AD50',  # Green
    'Uber\'s Share of Fare': '#E8C154',  # Yellow/Gold
    'Driver Earnings from Fare': '#60B5B2'  # Teal
}

# Plot bars for each metric
rects1 = ax.bar(x - width, ride_summary['Fare'], width, label='Fare', color=colors['Fare'])
```

```
    rects1 = ax.bar(x - width, ride_summary['Fare'], width, label='Fare', color=colors['Fare'])
    rects2 = ax.bar(x, ride_summary['Uber\'s Share of Fare'], width, label='Uber\'s Share of Fare', color=colors['Uber\'s Share of Fare'])
    rects3 = ax.bar(x + width, ride_summary['Driver Earnings from Fare'], width, label='Driver Earnings from Fare', color=colors['Driver Earnings from Fare'])

    # Customize the chart to match the original
    ax.set_ylabel('Value', fontsize=10)
    # Set a title for the entire chart
    fig.suptitle('Ride Type', fontsize=12, y=0.98)

    # Hide the original x-tick labels as we'll place them at the top
    ax.set_xticks(x)
    ax.set_xticklabels(['' for _ in ride_order])

    # Add ride type labels at the top of the chart
    for i, ride in enumerate(ride_order):
        ax.text(i, ax.get_ylim()[1] + 100000, ride, ha='center', va='bottom', fontsize=10)

    # Add gridlines to match the original chart (horizontal only)
    ax.grid(axis='y', linestyle='-', alpha=0.2)

    # Set the y-axis limit to match the original chart (up to 11M)
    # Add a bit of extra space at the top for the ride type labels
    ax.set_ylim(0, 11500000)

    # Format y-axis ticks to match the original chart (0M, 1M, 2M, etc.)
    y_ticks = np.arange(0, 12000000, 1000000)
    ax.set_yticks(y_ticks)
    ax.set_yticklabels([f'{int(tick/1000000)}M' for tick in y_ticks])

    # Remove the frame on the right and top
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

```

```
# Rotate the x-axis labels to match the original
plt.setp(ax.get_xticklabels(), rotation=0)

# Add x-axis labels for each group
x_labels = ['Fare', 'Uber\'s Share\nof Fare', 'Driver Ea\nrnings \n From Fare']
for i, group in enumerate(ride_order):
    for j, label in enumerate(x_labels):
        ax.text(i + (j-1)*width, -500000, label, ha='center', va='top', fontsize=8)

# Adjust layout to make room for the bottom labels
plt.subplots_adjust(bottom=0.2, top=0.85)

# Show the plot
plt.tight_layout(rect=[0, 0.05, 1, 1])

plt.show()
```

City1 / Ride Type

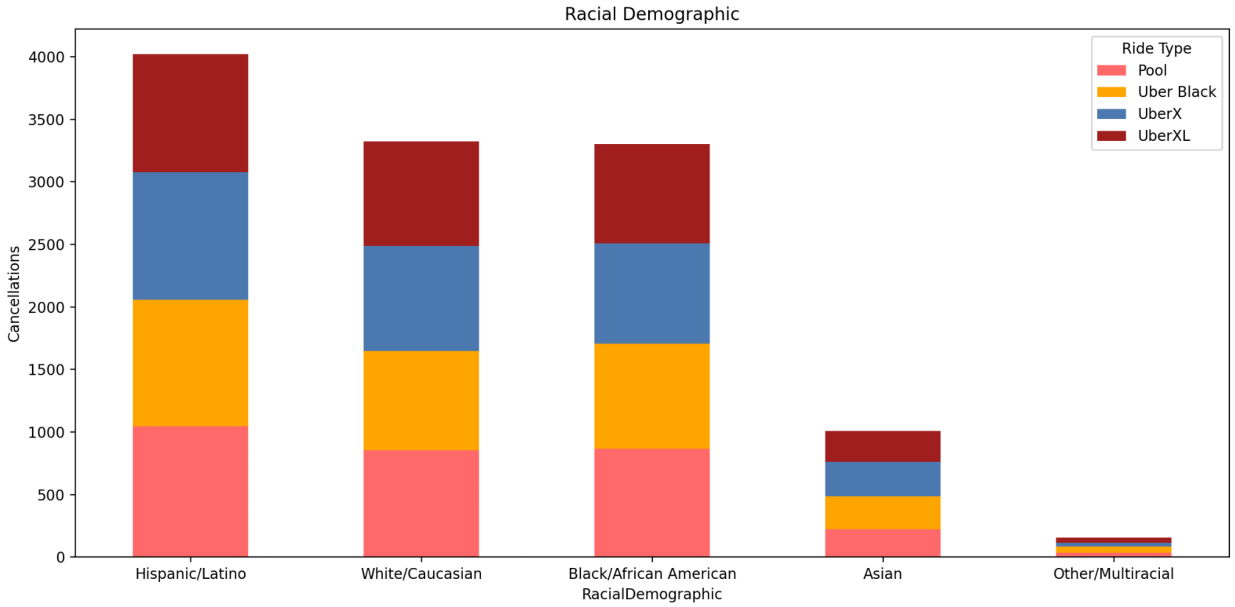



```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.gridspec as gridspec


# Function to read and prepare the data
def prepare_data(file_path):  1 usage
    df = pd.read_csv(file_path)

    # Filter for the cities and ride types shown in the chart
    target_cities = ["Chicago", "Houston", "Los Angeles", "New York"]
    target_ride_types = ["Pool", "Uber Black", "UberX", "UberXL"]

    df_filtered = df[df['City'].isin(target_cities) & df['RideType'].isin(target_ride_types)]

    # Calculate the average customer ratings by city and ride type
    avg_ratings = df_filtered.groupby(['City', 'RideType'])['CustomerRatings'].mean().reset_index()

    return avg_ratings


# Function to create the chart
def create_customer_ratings_chart(data):  1 usage
    # Set up figure and style
    plt.style.use('ggplot')
    fig = plt.figure(figsize=(12, 8))

```

```
    # Create a grid layout matching the original chart
    gs = gridspec.GridSpec( nrows: 1,  ncols: 4, width_ratios=[1, 1, 1, 1])

    # Define colors for different ride types
    colors = {
        'Pool': '#3274A1',
        'Uber Black': '#E1812C',
        'UberX': '#E15759',
        'UberXL': '#3BA097'
    }

    # Create subplots for each city
    axes = []
    cities = ["Chicago", "Houston", "Los Angeles", "New York"]

    for i, city in enumerate(cities):
        ax = plt.subplot(gs[i])
        axes.append(ax)

        # Filter data for this city
        city_data = data[data['City'] == city]

        # Set up x positions for bars
        ride_types = ["Pool", "Uber Black", "UberX", "UberXL"]
        x_pos = np.arange(len(ride_types))
```

```
        # Plot bars for this city
        for j, ride_type in enumerate(ride_types):
            ride_data = city_data[city_data['RideType'] == ride_type]
            if not ride_data.empty:
                ax.bar(x_pos[j], ride_data['CustomerRatings'].values[0],
                       color=colors[ride_type], width=0.7)

        # Set x-ticks
        ax.set_xticks(x_pos)
        ax.set_xticklabels(ride_types, rotation=90, fontsize=9)

        # Set y-axis limits to match original
        ax.set_ylim(0, 10.5)

        # Add city name at the top
        ax.text( x: 1.5,  y: 10.2, city, ha='center', fontsize=11)

        # Configure grid lines
        ax.grid(True, axis='y', linestyle='-', alpha=0.2)
        ax.set_axisbelow(True)

        # Remove top and right spines
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)

        # Only add y-axis label for the first subplot
        if i == 0:
```

```
79
80          # Only add y-axis label for the first subplot
81          if i == 0:
82              ax.set_ylabel('Avg. Customer Ratings', fontsize=10)
83
84          # Remove y-ticks for all but the first subplot
85          if i > 0:
86              ax.set_yticklabels([])
87
88      # Adjust layout
89      plt.subplots_adjust(wspace=0.05, bottom=0.15, top=0.85)
90
91      return fig
92
93
94  # Main function
95  def main():  1 usage
96      # Path to your CSV file
97      file_path = '/Users/libbyford/Desktop/Uber Data.csv'
98      data = prepare_data(file_path)
99      fig = create_customer_ratings_chart(data)
100     fig.suptitle( t: "City1 / Ride Type", y=0.95, fontsize=12)
101     plt.show()
102 if __name__ == "__main__":
103     main()
```




```python
import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv('/Users/libbyford/Desktop/Uber Data.csv')

# Group by racial demographic and ride type, summing cancellations
cancellation_counts = df.groupby(['RacialDemographic', 'RideType'])['Cancellations'].sum().reset_index()

# Pivot for stacked bar
pivot_df = cancellation_counts.pivot(index='RacialDemographic', columns='RideType', values='Cancellations').fillna(0)

# Sort by total cancellations
pivot_df['Cancellations'] = pivot_df.sum(axis=1)
pivot_df_sorted = pivot_df.sort_values(by='Cancellations', ascending=False).drop(columns='Cancellations')

colors = {
    'Pool': '#FF6B6B',
    'Uber Black': '#FFA600',
    'UberX': '#4D7BB0',
    'UberXL': '#A02120'
}

# Plot
fig, ax = plt.subplots(figsize=(12, 6))
pivot_df_sorted.plot(
    kind='bar',
    stacked=True,
    color=colors,
    ax=ax
)
```

```
32
33  ax.set_ylabel('Cancellations')
34  ax.set_title('Racial Demographic')
35  ax.legend(title='Ride Type')
36  plt.xticks(rotation=0)
37  plt.tight_layout()
38  plt.show()
```