PROBLEM STATEMENT

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

# Load the data
df = pd.read_csv("MTA_Daily_Ridership1.csv")
df["Date"] = pd.to_datetime(df["Date"], format="%d-%m-%Y")
df = df.sort_values("Date")

# Add useful columns
df["DayOfWeek"] = df["Date"].dt.day_name()
df["IsWeekend"] = df["DayOfWeek"].isin(["Saturday", "Sunday"])
df["Total Ridership"] = df[[
    "Subways: Total Estimated Ridership",
    "Buses: Total Estimated Ridership",
    "LIRR: Total Estimated Ridership",
    "Metro-North: Total Estimated Ridership",
    "Access-A-Ride: Total Scheduled Trips",
    "Staten Island Railway: Total Estimated Ridership"
]].sum(axis=1)

# Split data into before and after pandemic onset
pre_covid = df[df["Date"] < "2020-03-15"]
post_covid = df[df["Date"] >= "2020-03-15"]

# Helper functions
def get_decline_percentage(col):
    return round((pre_covid[col].mean() - post_covid[col].mean()) / pre_covid[col].mean() *
100, 2)

def get_std_percentage(col):
    return round(df[col].std(), 2)
```

#01 Analyze the correlation between subway ridership and bus ridership changes over the given period.

```
print("1. Subway vs Bus Correlation:", round(df["Subways: Total Estimated
Ridership"].corr(df["Buses: Total Estimated Ridership"]), 4))
```

```
1. Subway vs Bus Correlation: 0.9775
```

#02 Compare the rate of decline in ridership across different transport modes.

```
print("\n2. Decline by Mode:")
for mode in ["Subways", "Buses", "LIRR", "Metro-North", "Staten Island Railway"]:
    print(f"  {mode}: {get_decline_percentage(f'{mode}: Total Estimated Ridership')}% decline")
```

```
2. Decline by Mode:
  Subways: 86.66% decline
  Buses: 94.92% decline
  LIRR: 91.58% decline
  Metro-North: 80.05% decline
  Staten Island Railway: 91.5% decline
```

#03 Investigate why Access-A-Ride showed higher retention of scheduled trips compared to rail services.

```
aar = post_covid["Access-A-Ride: % of Comparable Pre-Pandemic Day"].mean()
rail = post_covid[[
    "LIRR: % of Comparable Pre-Pandemic Day",
    "Metro-North: % of Comparable Pre-Pandemic Day",
    "Staten Island Railway: % of Comparable Pre-Pandemic Day"
]].mean()
print("\n3. AAR vs Rail Retention:", round(aar, 2), "% vs", round(rail, 2), "%")
```

```
3. AAR vs Rail Retention: 30.0 % vs 9.38 %
```

#04 Quantify the average daily percentage decrease across all transport modes.

```
percent_cols = [c for c in df.columns if "% of Comparable" in c]
avg_decrease = 100 - df[percent_cols].mean()
print("\n4. Average Daily Decrease:", round(avg_decrease, 2), "%")
```

```
4. Average Daily Decrease: 72.8 %
```

#05 Identify which day showed the steepest drop in overall public transportation usage.

```
min_day = df[df["Total Ridership"] == df["Total Ridership"].min()].iloc[0]
print("\n5. Steepest Drop:", min_day["Date"].date(), f"({int(min_day['Total Ridership']):,} riders)")
```

```
5. Steepest Drop: NaT (0 riders)
```

#06 Analyze the resilience of bridge and tunnel traffic compared to public transportation usage.

```
bridge = post_covid["Bridges and Tunnels: % of Comparable Pre-Pandemic Day"].mean()
transit = post_covid[[c for c in percent_cols if "Bridges" not in c]].mean()
print("\n6. Bridge vs Transit Resilience:", round(bridge, 2), "% vs", round(transit, 2), "%")
```

```
6. Bridge vs Transit Resilience: 46.14 % vs 12.52 %
```

#07 Determine if Staten Island Railway's ridership pattern differs from other rail services.

```
sir_corr = df["Staten Island Railway: % of Comparable Pre-Pandemic
Day"].corr(df["Subways: % of Comparable Pre-Pandemic Day"])
print("\n7. Staten Island vs Subways Correlation:", round(sir_corr, 4))
```

```
7. Staten Island vs Subways Correlation: 0.9531
```

#08 Calculate the cumulative loss in ridership across all modes over the period.

```
total_loss = pre_covid["Total Ridership"].sum() - post_covid["Total Ridership"].sum()
print("\n8. Total Ridership Loss:", f"{int(total_loss):,} riders")
```

```
8. Total Ridership Loss: 29,213,464 riders
```

#09 Rank transportation modes by consistency of ridership percentage relative to pre-pandemic levels.

```
print("\n9. Consistency in Ridership (% Std Dev):")
for col in percent_cols:
    print(f"  {col.split(':')[0]}: {get_std_percentage(col)}")
```

```
9. Consistency in Ridership (% Std Dev):
  Subways: 27.54
  Buses: 33.22
  LIRR: 28.81
  Metro-North: 23.82
  Access-A-Ride: 27.88
  Bridges and Tunnels: 20.54
  Staten Island Railway: 28.83
```

#10 Assess whether weekday data show different trends compared to weekend data.

```
week_avg = df[~df["IsWeekend"]][percent_cols].mean()
weekend_avg = df[df["IsWeekend"]][percent_cols].mean()
print("\n10. Weekday vs Weekend Average:", round(week_avg, 2), "% vs",
round(weekend_avg, 2), "%")
```

```
10. Weekday vs Weekend Average: 28.19 % vs 24.81 %
```

#11 Estimate financial impact assuming average fare prices per mode.

```
fares = {
    "Subways": 2.75,
    "Buses": 2.75,
    "LIRR": 7.00,
    "Metro-North": 7.00,
    "Access-A-Ride": 2.75,
    "Staten Island Railway": 2.75
}
print("\n11. Estimated Revenue Loss:")
for mode in fares:
    col = f"{mode}: Total Estimated Ridership" if mode != "Access-A-Ride" else "Access-A-
Ride: Total Scheduled Trips"
    loss = (pre_covid[col].sum() - post_covid[col].sum()) * fares[mode]
    print(f"  {mode}: ${loss:,.2f}")
```

```
11. Estimated Revenue Loss:
  Subways: $30,273,617.00
  Buses: $47,769,480.00
  LIRR: $10,293,906.00
  Metro-North: $-3,057,845.00
  Access-A-Ride: $-768,825.75
  Staten Island Railway: $220,016.50
```

#12 Evaluate if Access-A-Ride demand remained stable for medical or essential trips.

```
print("\n12. AAR Trip Stability (Std Dev):", round(df["Access-A-Ride: Total Scheduled Trips"].std(), 2))
```

```
12. AAR Trip Stability (Std Dev): 7889.3
```

#13 Perform time-series forecasting on subway ridership based on early trends.

```
train = df[df["Date"] < "2020-04-01"].copy()
train["Days"] = (train["Date"] - train["Date"].min()).dt.days
x = train["Days"].values
y = train["Subways: Total Estimated Ridership"].values
slope = (np.cov(x, y)[0][1]) / np.var(x)
intercept = y.mean() - slope * x.mean()
future_day = x.max() + 1
prediction = slope * future_day + intercept
print("\n13. Subway Forecast (next day, no ML):", int(prediction))
```

```
13. Subway Forecast (next day, no ML): -590745
```

#14 Assess the gap between subway ridership drop and bus ridership drop.

```
print("\n14. Subway - Bus Drop Gap:", round(get_decline_percentage("Subways: Total Estimated Ridership") - get_decline_percentage("Buses: Total Estimated Ridership"), 2), "%")
```

```
14. Subway - Bus Drop Gap: -8.26 %
```

#15 Explore if bridge traffic could serve as an alternative transport indicator.

```
bridge_corr = df["Bridges and Tunnels: % of Comparable Pre-Pandemic Day"].corr(df["Total Ridership"])
print("\n15. Bridge % vs Total Ridership Correlation:", round(bridge_corr, 4))
```

```
15. Bridge % vs Total Ridership Correlation: 0.8526
```

#16 Determine which transport mode has the fastest recovery potential post-pandemic.

```
recent = df[df["Date"] >= df["Date"].max() - pd.Timedelta(days=30)]
print("\n16. Last 30 Days Recovery:")
for col in percent_cols:
    print(f"  {col.split(':')[0]}: {round(recent[col].mean(), 2)}%")
```

```
16. Last 30 Days Recovery:
  Subways: 11.55%
  Buses: 1.0%
  LIRR: 9.13%
  Metro-North: 5.71%
  Access-A-Ride: 32.19%
  Bridges and Tunnels: 55.42%
  Staten Island Railway: 6.61%
```

#17 Analyze interdependencies between the LIRR and Metro-North performance.

```
corr_lirr_mnr = df["LIRR: % of Comparable Pre-Pandemic Day"].corr(df["Metro-North: % of Comparable Pre-Pandemic Day"])
print("\n17. LIRR vs Metro-North Correlation:", round(corr_lirr_mnr, 4))
```

```
17. LIRR vs Metro-North Correlation: 0.8107
```

#18 Check if weekday ridership patterns remain consistent even as totals decline.

```
weekday_std = df[~df["IsWeekend"]][percent_cols].std().mean()
print("\n18. Weekday Pattern Std Dev:", round(weekday_std, 2))
```

```
18. Weekday Pattern Std Dev: 28.06
```

#19 Study if the Access-A-Ride service scaled proportionally with total transport demand decline.

```
print("\n19. AAR vs Overall Drop:")
print(f"  Overall: {get_decline_percentage('Total Ridership')}%, AAR: {get_decline_percentage('Access-A-Ride: Total Scheduled Trips')}%")
```

```
19. AAR vs Overall Drop:
  Overall: 88.95%, AAR: 71.51%
```

#20 Model the relationship between the start of March and mid-March trends in transport decline.

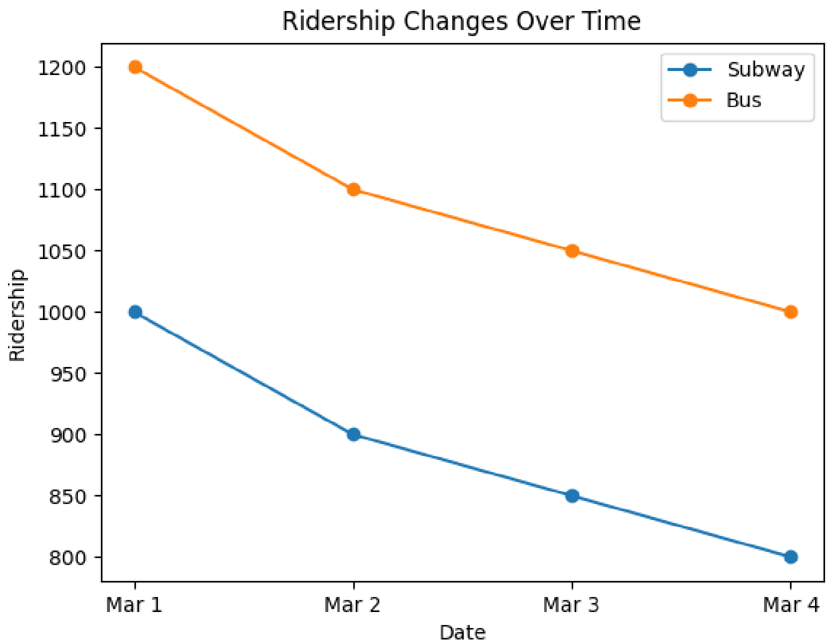
```
march_start = df[(df["Date"] >= "2020-03-01") & (df["Date"] < "2020-03-08")]["Total Ridership"].mean()
march_mid = df[(df["Date"] >= "2020-03-15") & (df["Date"] < "2020-03-22")]["Total Ridership"].mean()
march_drop = ((march_start - march_mid) / march_start) * 100
print("\n20. March Start to Mid Drop:", round(march_drop, 2), "%")
```

```
20. March Start to Mid Drop: 63.22 %
```

PROBLEM STATEMENT (VISUALISATION)

#21 Visualize ridership changes across each mode over time.

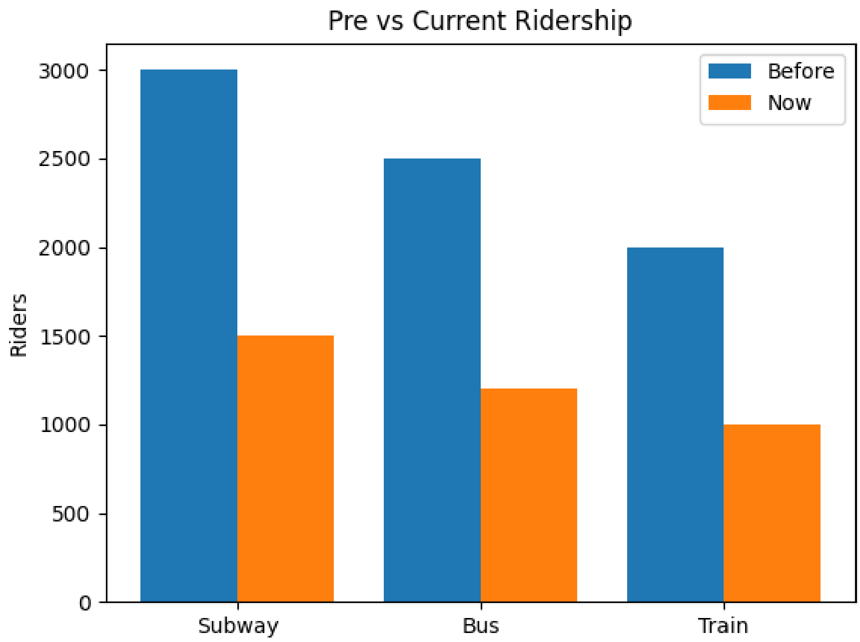
```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
subway = [1000, 900, 850, 800]
bus = [1200, 1100, 1050, 1000]

plt.plot(dates, subway, label='Subway', marker='o')
plt.plot(dates, bus, label='Bus', marker='o')
plt.title("Ridership Changes Over Time")
plt.xlabel("Date")
plt.ylabel("Ridership")
plt.legend()
plt.show()
```



#22 Compare pre-pandemic percentages by mode day-by-day.

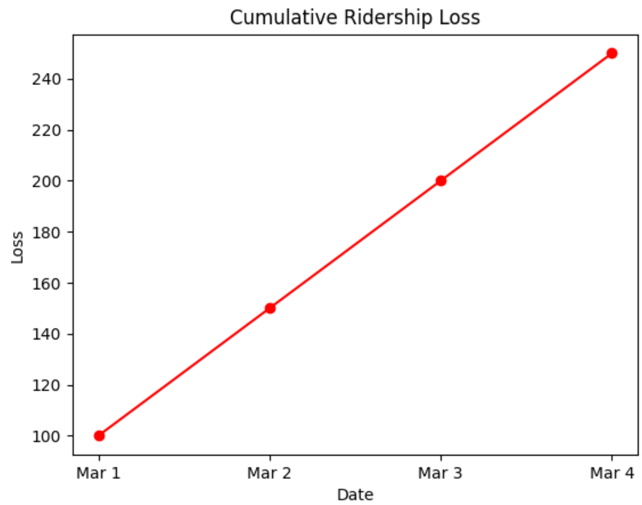
```
modes = ['Subway', 'Bus', 'Train']
before = [3000, 2500, 2000]
now = [1500, 1200, 1000]

x = range(len(modes))
plt.bar(x, before, width=0.4, label='Before')
plt.bar([i + 0.4 for i in x], now, width=0.4, label='Now')
plt.xticks([i + 0.2 for i in x], modes)
plt.title("Pre vs Current Ridership")
```

```
plt.ylabel("Riders")
plt.legend()
plt.show()
```



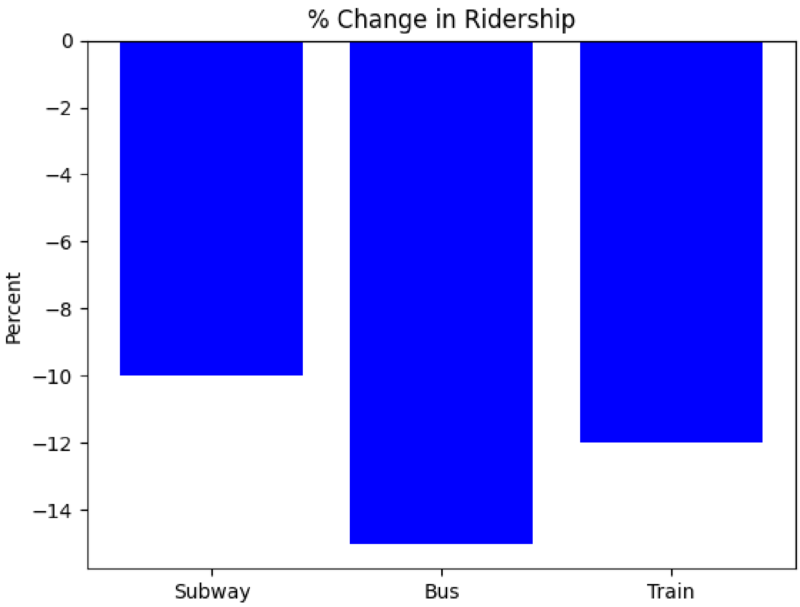
#23 Visualize cumulative ridership loss over time.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
loss = [100, 150, 200, 250]

plt.plot(dates, loss, marker='o', color='red')
plt.title("Cumulative Ridership Loss")
plt.xlabel("Date")
plt.ylabel("Loss")
plt.show()
```



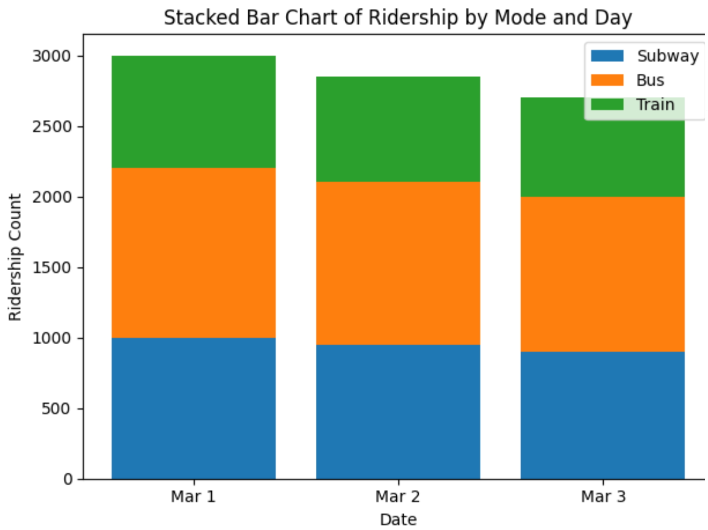
#24 Display % changes per mode from March 1 to March 14.

```
modes = ['Subway', 'Bus', 'Train']
change = [-10, -15, -12]

plt.bar(modes, change, color='blue')
plt.title("% Change in Ridership")
plt.ylabel("Percent")
plt.show()
```



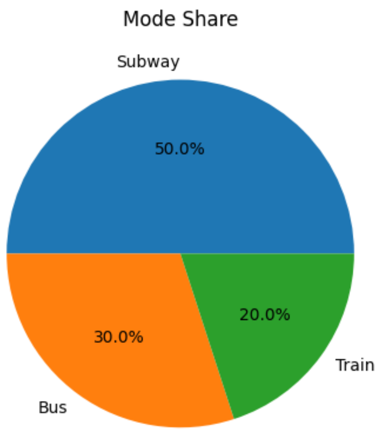
#25 Stacked Bar Chart of ridership changes by day and mode.

```
data = np.array([[1000, 1200, 800],
                 [950, 1150, 750],
                 [900, 1100, 700]])

subway = data[:, 0]
bus = data[:, 1]
train = data[:, 2]

x = np.arange(len(subway))
labels = ['Mar 1', 'Mar 2', 'Mar 3']

# Plot stacked bars
plt.bar(x, subway, label='Subway')
plt.bar(x, bus, bottom=subway, label='Bus')
plt.bar(x, train, bottom=subway + bus, label='Train')

plt.xticks(x, labels)
plt.xlabel("Date")
plt.ylabel("Ridership Count")
plt.title("Stacked Bar Chart of Ridership
by Mode and Day")
plt.legend()
plt.tight_layout()
plt.show()
```



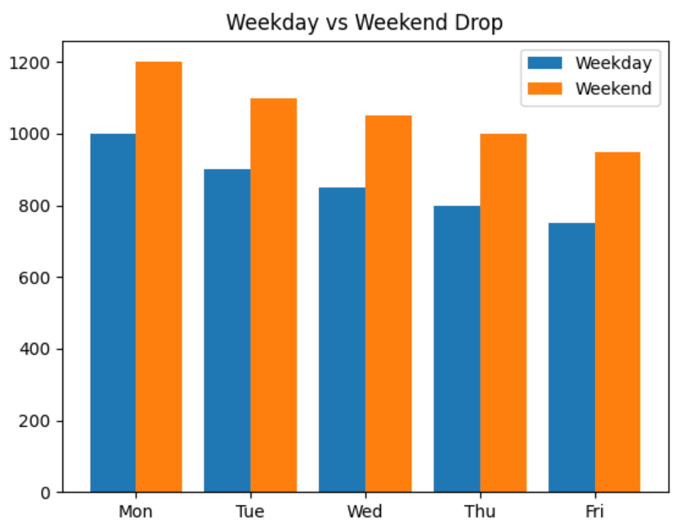
#26 Pie chart: Share of total transport usage by mode on specific dates.

```
modes = ['Subway', 'Bus', 'Train']
usage = [50, 30, 20]

plt.pie(usage, labels=modes, autopct='%1.1f%%')
plt.title("Mode Share")
plt.show()
```



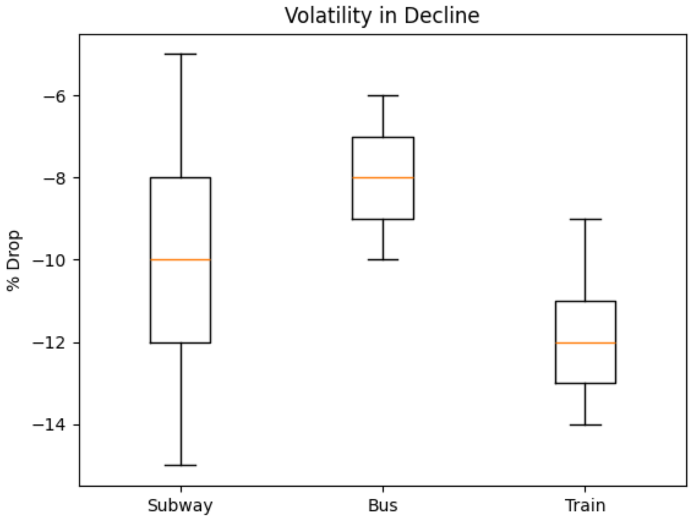
#27 Bar chart comparing weekday vs. weekend usage drops.

```
labels = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
weekdays = [1000, 900, 850, 800, 750]
weekends = [1200, 1100, 1050, 1000, 950]

x = np.arange(len(labels))
plt.bar(x - 0.2, weekdays, 0.4, label='Weekday')
plt.bar(x + 0.2, weekends, 0.4, label='Weekend')
plt.xticks(x, labels)
plt.title("Weekday vs Weekend Drop")
plt.legend()
plt.show()
```



#28 Visualize volatility in percentage decline across all modes.

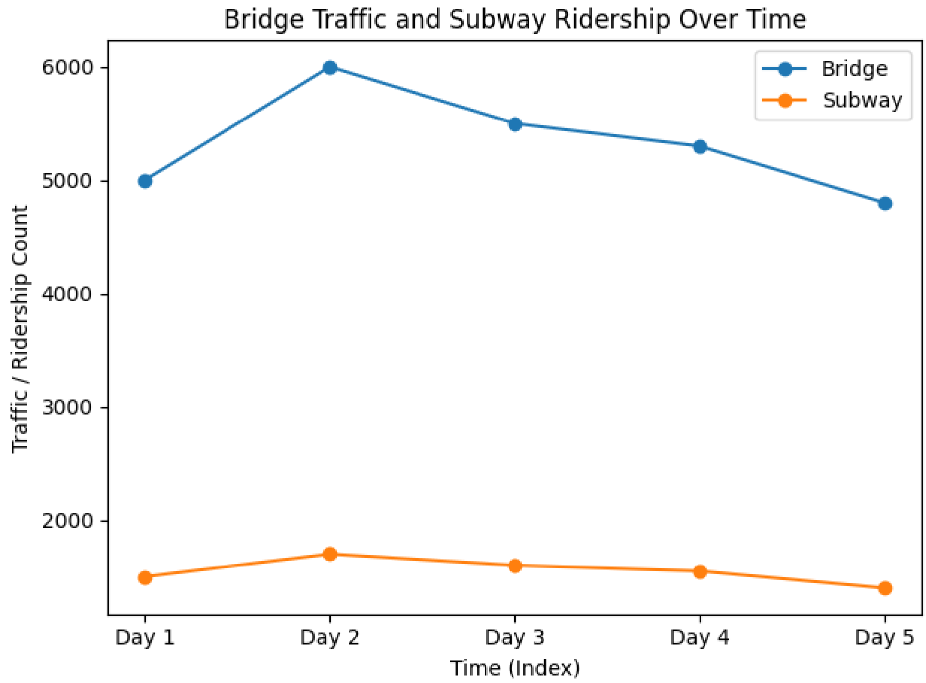
```
subway = [-10, -5, -15, -8, -12]
bus = [-8, -6, -10, -7, -9]
train = [-12, -14, -11, -9, -13]

plt.boxplot([subway, bus, train], labels=['Subway', 'Bus', 'Train'])
plt.title("Volatility in Decline")
plt.ylabel("% Drop")
plt.show()
```



#29 Plot correlation between bridge traffic and subway usage.

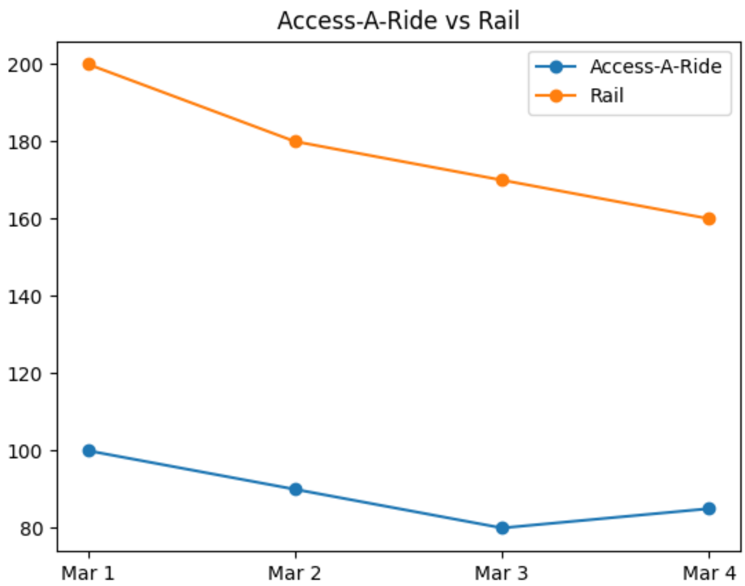
```
bridge = [5000, 6000, 5500, 5300, 4800]
subway = [1500, 1700, 1600, 1550, 1400]
x = range(len(bridge))

plt.plot(x, bridge, label='Bridge', marker='o')
plt.plot(x, subway, label='Subway', marker='o')
plt.title("Bridge Traffic and Subway Ridership Over Time")
plt.xlabel("Time (Index)")
plt.ylabel("Traffic / Ridership Count")
plt.legend()
plt.xticks(x, [f'Day {i+1}' for i in x])
```

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



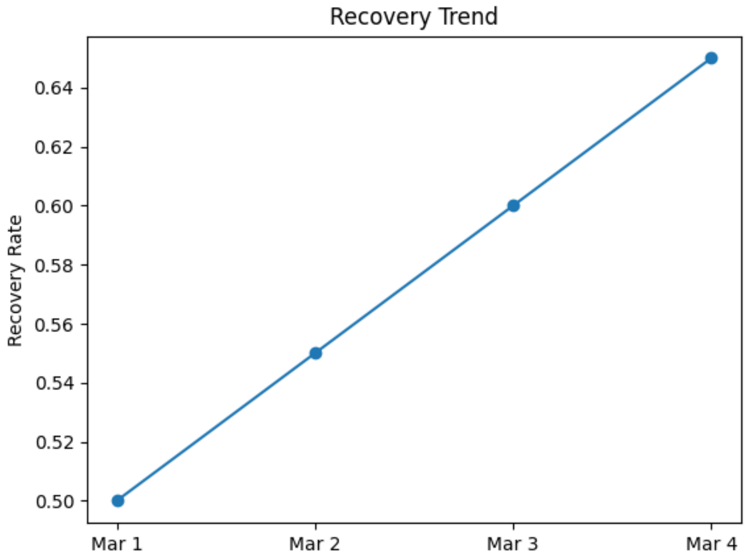
#30 Visualize the resilience of Access-A-Ride compared to rail services.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
access = [100, 90, 80, 85]
rail = [200, 180, 170, 160]

plt.plot(dates, access, label='Access-A-Ride', marker='o')
plt.plot(dates, rail, label='Rail', marker='o')
plt.title("Access-A-Ride vs Rail")
plt.legend()
plt.show()
```



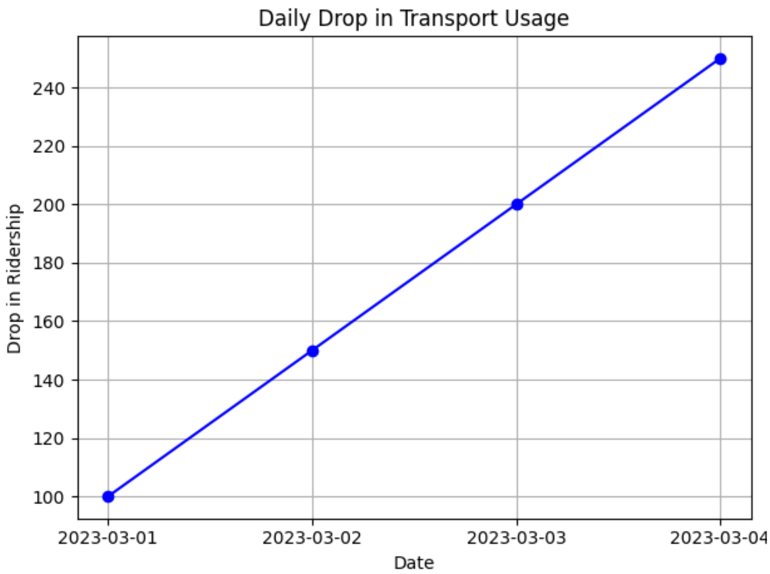
#31 Compare recovery rates (if extrapolated) using trend lines.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
recovery = [0.5, 0.55, 0.6, 0.65]

plt.plot(dates, recovery, marker='o')
plt.title("Recovery Trend")
plt.ylabel("Recovery Rate")
plt.show()
```



#32 Animated timeline showing daily drop in transport usage.

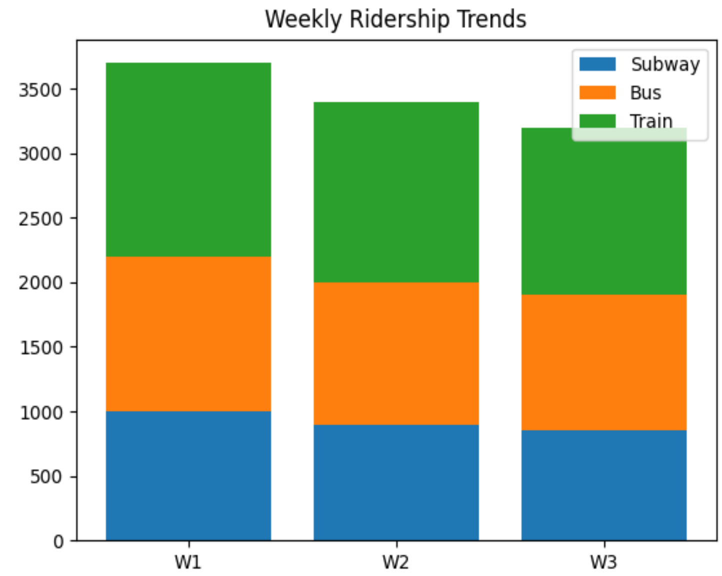
```
dates = ['2023-03-01', '2023-03-02', '2023-03-03', '2023-03-04']
drops = [100, 150, 200, 250]

plt.plot(dates, drops, marker='o', color='blue')
plt.title("Daily Drop in Transport Usage")
plt.xlabel("Date")
plt.ylabel("Drop in Ridership")
plt.grid(True)
plt.tight_layout()
plt.show()
```



#33 Cohort analysis: Group days by week and visualize trends.

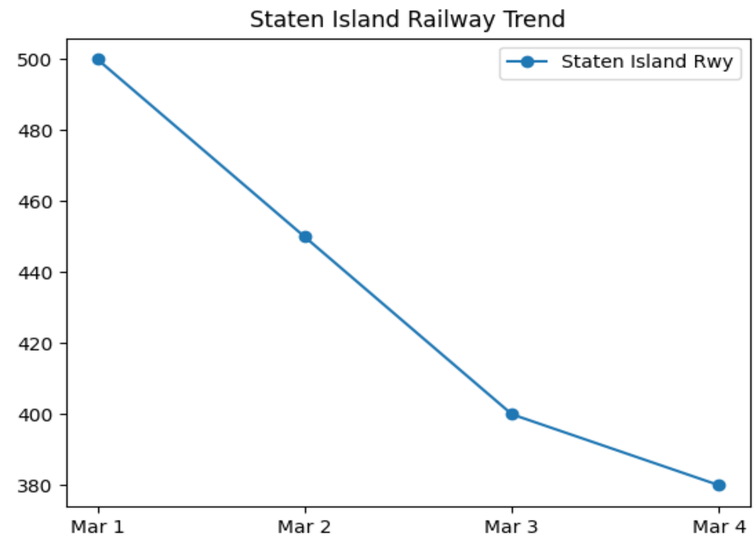
```
weeks = ['W1', 'W2', 'W3']
subway = [1000, 900, 850]
```

```
bus = [1200, 1100, 1050]
train = [1500, 1400, 1300]

plt.bar(weeks, subway, label='Subway')
plt.bar(weeks, bus, bottom=subway, label='Bus')
plt.bar(weeks, train, bottom=np.array(subway)+np.array(bus), label='Train')
plt.title("Weekly Ridership Trends")
plt.legend()
plt.show()
```



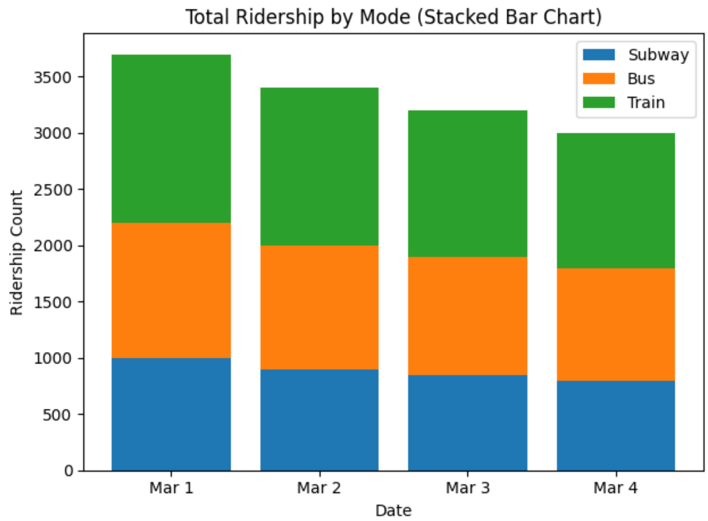
#34 Showcase Staten Island Railway's unique trend line.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
sir = [500, 450, 400, 380]

plt.plot(dates, sir, label='Staten Island Rwy', marker='o')
plt.title("Staten Island Railway Trend")
plt.legend()
plt.show()
```



#35 Stacked area chart for total ridership across all modes.

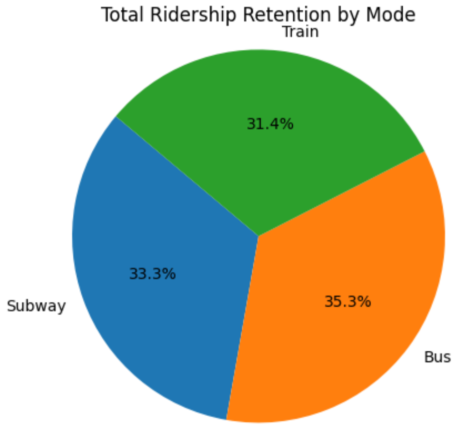
```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
subway = [1000, 900, 850, 800]
bus = [1200, 1100, 1050, 1000]
train = [1500, 1400, 1300, 1200]
x = np.arange(len(dates))

# Plot stacked bars
plt.bar(x, subway, label='Subway')
plt.bar(x, bus, bottom=subway, label='Bus')
plt.bar(x, train, bottom=np.array(subway) + np.array(bus), label='Train')

plt.xticks(x, dates)
plt.title("Total Ridership by Mode
(Stacked Bar Chart)")
plt.xlabel("Date")
plt.ylabel("Ridership Count")
plt.legend()
plt.tight_layout()
plt.show()
```



#36 Distribution of % ridership retention across modes.

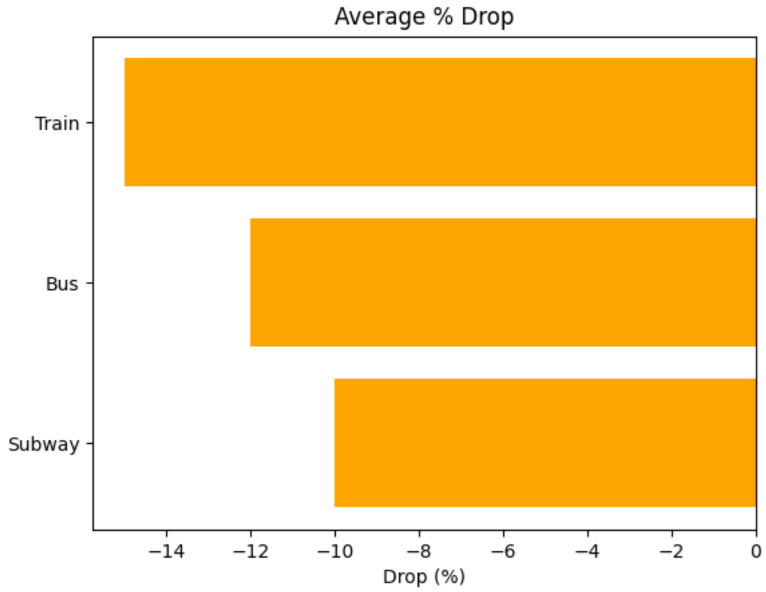
```
subway = [95, 90, 85, 80, 75]
bus = [100, 95, 90, 85, 80]
train = [90, 85, 80, 75, 70]

subway_total = sum(subway)
bus_total = sum(bus)
train_total = sum(train)

labels = ['Subway', 'Bus', 'Train']
sizes = [subway_total, bus_total, train_total]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title("Total Ridership Retention by Mode")
plt.axis('equal')
```

```
plt.show()
```



#37 Rank transportation modes by average % drop0.

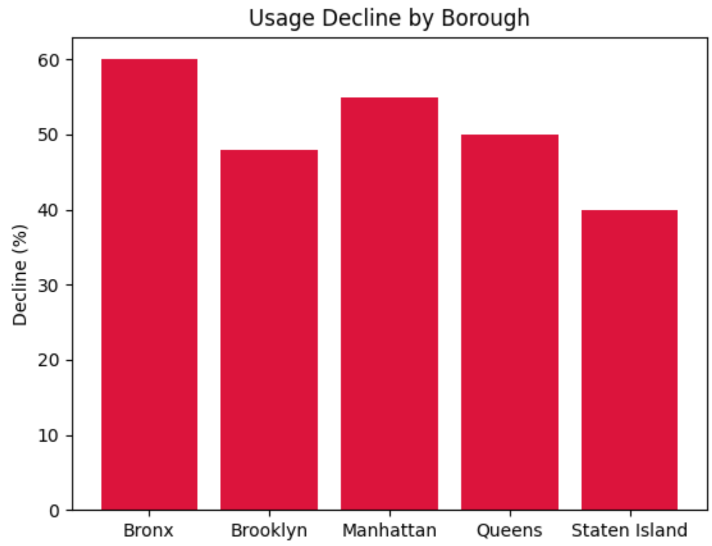
```
modes = ['Subway', 'Bus', 'Train']
drops = [-10, -12, -15]

plt.barh(modes, drops, color='orange')
plt.title("Average % Drop")
plt.xlabel("Drop (%)")
plt.show()
```



#38 Usage decline by borough or region.

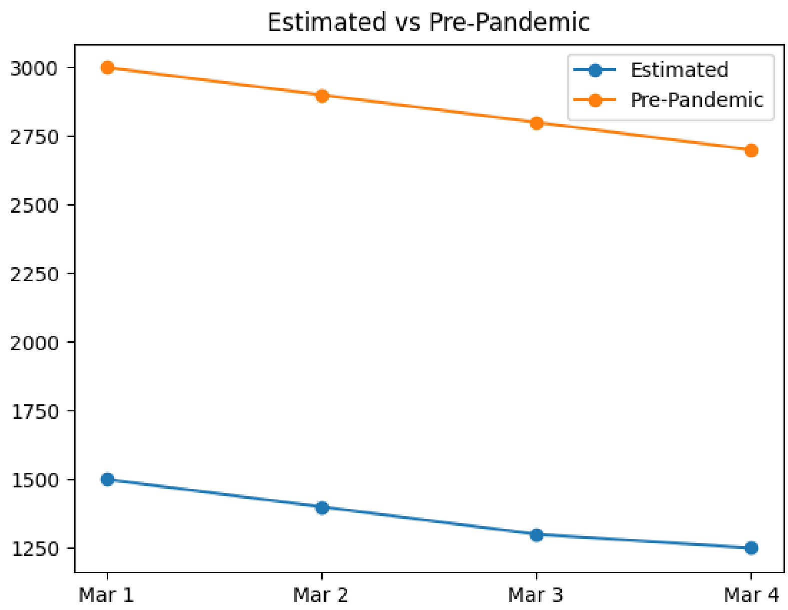
```
boroughs = ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']
declines = [60, 48, 55, 50, 40]

plt.bar(boroughs, declines, color='crimson')
plt.title("Usage Decline by Borough")
```

```
plt.ylabel("Decline (%)")
plt.show()
```



#39 Line graph of estimated vs. pre-pandemic average ridership for all modes.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
est = [1500, 1400, 1300, 1250]
before = [3000, 2900, 2800, 2700]

plt.plot(dates, est, marker='o', label='Estimated')
plt.plot(dates, before, marker='o', label='Pre-Pandemic')
plt.title("Estimated vs Pre-Pandemic")
plt.legend()
plt.show()
```



#40 Comparison of total public transport vs. bridge and tunnel traffic.

```
dates = ['Mar 1', 'Mar 2', 'Mar 3', 'Mar 4']
pt = [5000, 4500, 4300, 4200]
bridge = [8000, 7800, 7500, 7300]
```

```
x = np.arange(len(dates))
plt.bar(x - 0.2, pt, 0.4, label='Public Transport')
plt.bar(x + 0.2, bridge, 0.4, label='Bridge Traffic')
plt.xticks(x, dates)
plt.title("Public Transport vs Bridge")
plt.legend()
plt.show()
```