Automotive Car Loan Analysis

The attached analysis dives in to Auto Loan patterns over a 10-year period, broken down by quarter; to answer the questions below.

1. Are the number of people taking out loans increasing or decreasing?
2. Has the number of people purchasing cars increased or decreased?
3. Do trends in loan approvals correlate with trends in car purchases?

Data Sources

To ensure we had consistency among all our data sets, we first reviewed the timeframe we wanted to focus on. To work with a large enough timeframe that we could see visible trends in the data. Since data for Population for 2024 was not yet available, we chose to end our pull parameter for all the data at Q4 2023. FRED (Federal Reserve Bank of St. Louis) enables a pull parameter to be set for each data set. Once we determined our range, we extracted the relevant .csv files into our GitHub repository. The files we chose are noted below:

•Balance Sheet: Total Assets: Loans to Individuals: Other Loans to Individuals: Auto Loans = https://fred.stlouisfed.org/seriesBeta/QBPBSTASLNINDVOLNINDCARLN
•Average Finance Rate of New Car Loans at Finance Companies, Amount of Finance Weighted = https://fred.stlouisfed.org/series/RIELPCFANNM
•Average Amount Financed for New Car Loans at Finance Companies = https://fred.stlouisfed.org/series/DTCTLVENANM
•Total Vehicle Sales = https://fred.stlouisfed.org/series/TOTALSA
•Auto Inventory/Sales Ratio = https://fred.stlouisfed.org/series/AISRSA •Population = https://fred.stlouisfed.org/seriesBeta/POPTHM
•API - Population Data = https://api.census.gov/data/2019/pep/charagegroups?get=NAME,POP&for=us:*&key={census_key}
•API 2 - Population Data = https://datausa.io/api/data?drilldowns=Nation&measures=Population

---

Data cleaning, importing and coding in "VS Code"

```
#Import Dependencies
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import requests
import time
import json
from scipy.stats import linregress
from census import Census
from config import census_key
```

Import identified data sources from FRED.

```
# Store filepaths into variables
total_asset_loan_value_csv = Path("Resources/Total Loan Assets.csv")
average_rate_of_financed_loan_csv = Path("Resources/Average Loan Finance.csv")
average_amount_financed_csv= Path("Resources/Average Amount Financed.csv")
total_vehicle_sales_csv = Path("Resources/Total Vehicle Sales.csv")
total_population_csv = Path("Resources/Total Population USA.csv")
```

```
# Read the CSV files
total_asset_loan_value_df = pd.read_csv(total_asset_loan_value_csv)
average_rate_of_financed_loan_df = pd.read_csv(average_rate_of_financed_loan_csv)
average_amount_financed_df = pd.read_csv(average_amount_financed_csv)
total_vehicle_sales_df = pd.read_csv(total_vehicle_sales_csv)
total_population_df = pd.read_csv(total_population_csv)
```

Question 1 Analysis:

Are the number of people taking out loans increasing or decreasing?

Using VS Code, we created a Jupyter Notebook to analyze and clean the datasets we planned to work with. Our analysis began with the Total Asset Loan Value file.

```
#Dispay the data
total_asset_loan_value_df.head()
```

	DATE	QBPBSTASLNINDVOLNINDCARLN
0	2014-07-01	379531.304
1	2014-10-01	385156.262
2	2015-01-01	389633.848
3	2015-04-01	397633.220
4	2015-07-01	407873.676

Our first step was to rename the column in the data set to be clear and easier to utilize within our code.

```
#Rename Columns & Change to Float
total_asset_loan_value_df = total_asset_loan_value_df.rename(columns={"QBPBSTASLNINDVOLNINDCARLN": "Total Loan Amount"})
total_asset_loan_value_df.head()
```

	DATE	Total Loan Amount
0	2014-07-01	379531.304
1	2014-10-01	385156.262
2	2015-01-01	389633.848
3	2015-04-01	397633.220
4	2015-07-01	407873.676

Our next step focused on formatting the number to match the unit of measure which was “Millions of U.S. Dollars”

```
total_asset_loan_value_df = pd.DataFrame(total_asset_loan_value_df)
total_asset_loan_value_df["Total Loan Amount Millions"] = total_asset_loan_value_df["Total Loan Amount"] * 1000000
total_asset_loan_value_df = total_asset_loan_value_df.dropna()
total_asset_loan_value_df["Total Loan Amount Millions"] = total_asset_loan_value_df["Total Loan Amount Millions"].astype('int64')
total_asset_loan_value_df.head()
```

	DATE	Total Loan Amount	Total Loan Amount Millions
0	2014-07-01	379531.304	379531304000
1	2014-10-01	385156.262	385156262000
2	2015-01-01	389633.848	389633848000
3	2015-04-01	397633.220	397633220000
4	2015-07-01	407873.676	407873676000

The following step was to incorporate Loan Rates into the data frame by cleaning the data: renaming the column, removing "NaN" values, and calculating the quarterly rate by multiplying the monthly rate by 3.

```
#Rename Columns and calculate on a quarterly rate in percentage format.
average_rate_of_financed_loan_df = average_rate_of_financed_loan_df.rename(columns={"RIELPCFANNM": "Average Loan Rate"})
average_rate_of_financed_loan_df['Average Loan Rate'] = pd.to_numeric(average_rate_of_financed_loan_df['Average Loan Rate'], errors='coerce').fillna(0.0)
average_rate_of_financed_loan_df["Average Loan Rate % by Quatre"] = average_rate_of_financed_loan_df["Average Loan Rate"] / 100 * 3
average_rate_of_financed_loan_df.head()
```

✓ 0.0s 📓 Open 'average_rate_of_financed_loan_df' in Data Wrangler

	DATE	Average Loan Rate	Average Loan Rate % by Quatre
0	2014-07-01	4.68	0.1404
1	2014-10-01	4.82	0.1446
2	2015-01-01	5.19	0.1557
3	2015-04-01	5.38	0.1614
4	2015-07-01	4.92	0.1476

We then focused on the data set for Average Financed Amount and cleaned the data.

```
average_amount_financed_df=average_amount_financed_df.rename(columns={"DTCTLVENANM": "Average Financed Amount"})
average_amount_financed_df['Average Financed Amount'] = pd.to_numeric(average_amount_financed_df['Average Financed Amount'], errors='coerce').fillna(0.0)
average_amount_financed_df.head()
```

✓ 0.0s 📊 Open 'average_amount_financed_df' in Data Wrangler

	DATE	Average Financed Amount
0	2014-07-01	26370.63
1	2014-10-01	26755.30
2	2015-01-01	27272.36
3	2015-04-01	26932.27
4	2015-07-01	27697.60

To create a comprehensive data frame for analysis, we began merging the data using the DATE column with the pd.merge function.

```
#Merge the dataframes
revenue_rate_df = pd.merge(total_asset_loan_value_df, average_rate_of_financed_loan_df, on="DATE")
revenue_rate_df.head()
```

✓ 0.0s 📊 Open 'revenue_rate_df' in Data Wrangler

	DATE	Total Loan Amount	Total Loan Amount Millions	Average Loan Rate	Average Loan Rate % by Quatre
0	2014-07-01	379531.304	379531304000	4.68	0.1404
1	2014-10-01	385156.262	385156262000	4.82	0.1446
2	2015-01-01	389633.848	389633848000	5.19	0.1557
3	2015-04-01	397633.220	397633220000	5.38	0.1614
4	2015-07-01	407873.676	407873676000	4.92	0.1476

```
#Merge the dataframes
clean_df = pd.merge(revenue_rate_df, average_amount_financed_df, on="DATE")
clean_df.head()
```

✓ 0.0s 📊 Open 'clean_df' in Data Wrangler

	DATE	Total Loan Amount	Total Loan Amount Millions	Average Loan Rate	Average Loan Rate % by Quatre	Average Financed Amount
0	2014-07-01	379531.304	379531304000	4.68	0.1404	26370.63
1	2014-10-01	385156.262	385156262000	4.82	0.1446	26755.30
2	2015-01-01	389633.848	389633848000	5.19	0.1557	27272.36
3	2015-04-01	397633.220	397633220000	5.38	0.1614	26932.27
4	2015-07-01	407873.676	407873676000	4.92	0.1476	27697.60

To ensure these values can be utilized effectively, we validated their data types to ensure they were formatted correctly.

```
# verify that our data is the correct type in order to be used in calculations
clean_df.dtypes
```

✓ 0.0s

```
DATE                          object
Total Loan Amount            float64
Total Loan Amount Millions     int64
Average Loan Rate            float64
Average Loan Rate % by Quatre    float64
Average Financed Amount          float64
dtype: object
```

We can now calculate the Average Loan Amount by first determining the interest earned each quarter and then adding it to the average financed amount. We believe this approach provides a more accurate representation of the value that can be used to divide against the Total Loan Values on the balance sheet, helping to estimate the number of people receiving loans.

```
#Calculate average amount financed by and multiply by the average rate of that period to give us new total of revenue generated during that quarter
clean_df["Average Loan Amount"] = clean_df["Average Financed Amount"] * clean_df["Average Loan Rate % by Quatre"] + clean_df["Average Financed Amount"]
clean_df.head()
```

✓ 0.0s 哪 Open 'clean_df' in Data Wrangler

	DATE	Total Loan Amount	Total Loan Amount Millions	Average Loan Rate	Average Loan Rate % by Quatre	Average Financed Amount	Average Loan Amount
0	2014-07-01	379531.304	379531304000	4.68	0.1404	26370.63	30073.066452
1	2014-10-01	385156.262	385156262000	4.82	0.1446	26755.30	30624.116380
2	2015-01-01	389633.848	389633848000	5.19	0.1557	27272.36	31518.666452
3	2015-04-01	397633.220	397633220000	5.38	0.1614	26932.27	31279.138378
4	2015-07-01	407873.676	407873676000	4.92	0.1476	27697.60	31785.765760

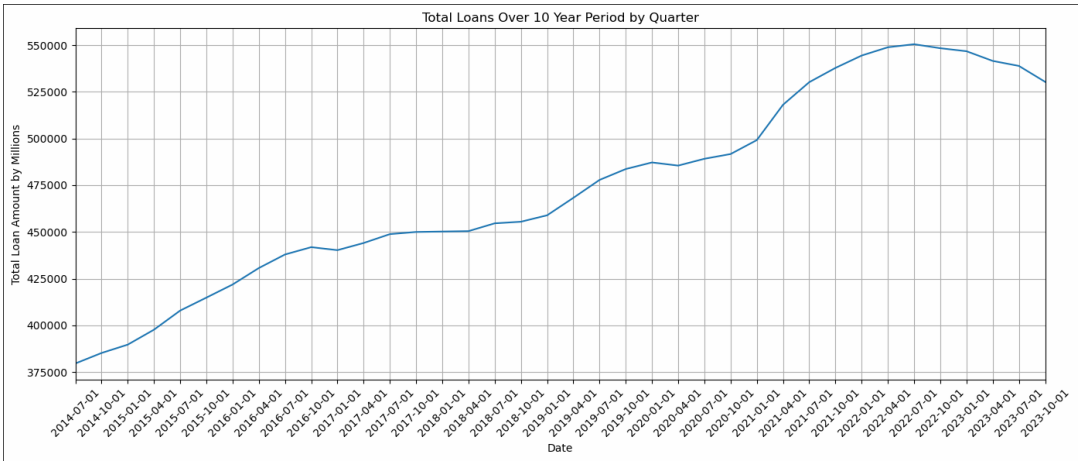
```
# Calculate the Total People getting loans by dividing Total Loan Amount (Millions) by our recaluclated amount for Revenue Generated
clean_df["Total People Getting Loans"] = clean_df["Total Loan Amount Millions"] / clean_df["Average Loan Amount"]
clean_df["Total People Getting Loans"] = clean_df["Total People Getting Loans"].astype(int)
clean_df.head()
```

✓ 0.0s 哪 Open 'clean_df' in Data Wrangler

	DATE	Total Loan Amount	Total Loan Amount Millions	Average Loan Rate	Average Loan Rate % by Quatre	Average Financed Amount	Average Loan Amount	Total People Getting Loans
0	2014-07-01	379531.304	379531304000	4.68	0.1404	26370.63	30073.066452	12620306
1	2014-10-01	385156.262	385156262000	4.82	0.1446	26755.30	30624.116380	12576893
2	2015-01-01	389633.848	389633848000	5.19	0.1557	27272.36	31518.666452	12362002
3	2015-04-01	397633.220	397633220000	5.38	0.1614	26932.27	31279.138378	12712409
4	2015-07-01	407873.676	407873676000	4.92	0.1476	27697.60	31785.765760	12831960

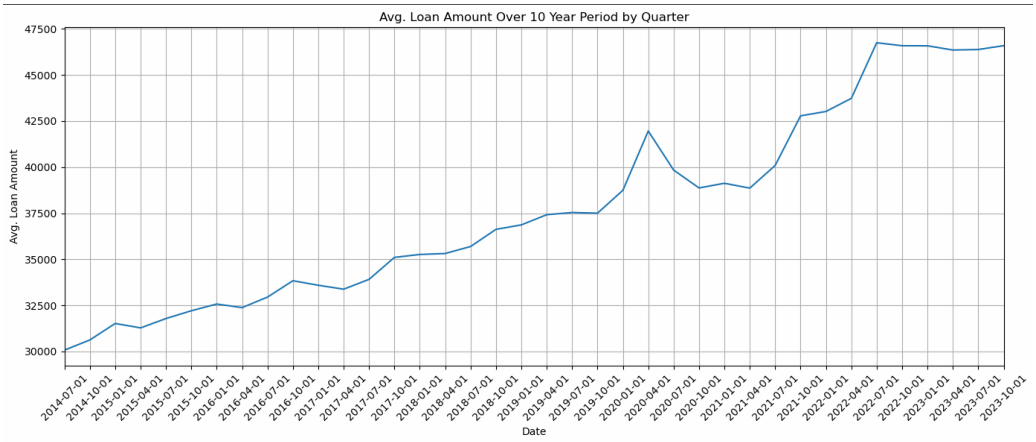
We began by constructing graphs, starting with the Total Loans over a 10-year period.

```
date = clean_df["DATE"]
Total_Loan_Amount = clean_df["Total Loan Amount"]
plt.figure(figsize=(14,6))
plt.plot(date, Total_Loan_Amount)
plt.title("Total Loans Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Total Loan Amount by Millions")
plt.xlim(clean_df["DATE"].min(),clean_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



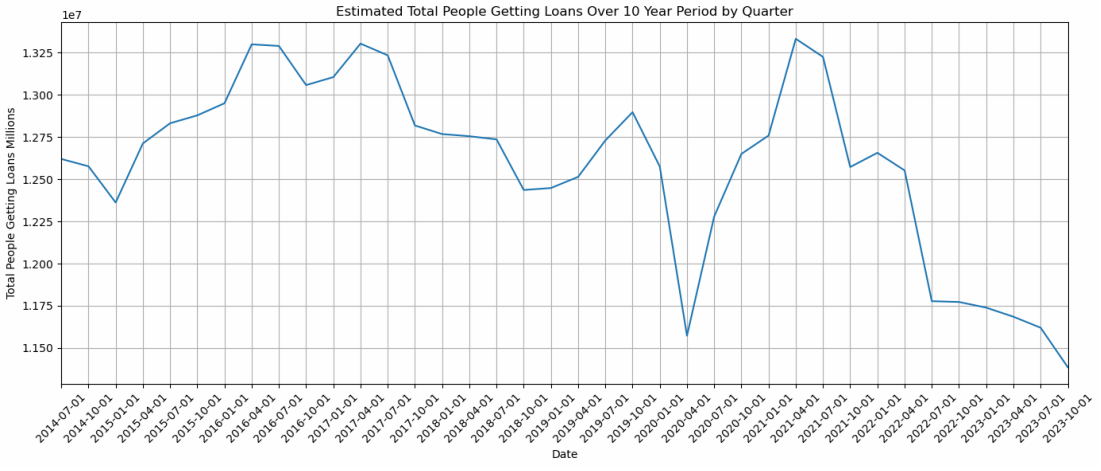
The subsequent step involved plotting the Average Loan Amount over the 10-year period

```
date = clean_df["DATE"]
Revenue_Generated = clean_df["Average Loan Amount"]
plt.figure(figsize=(14,6))
plt.plot(date, Revenue_Generated)
plt.title("Avg. Loan Amount Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Avg. Loan Amount")
plt.xlim(clean_df["DATE"].min(),clean_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Finally, we plotted the graph representing the Total People Getting Loans, which provided the insights needed to answer the question we posed.

```
date = clean_df["DATE"]
Total_People = clean_df["Total People Getting Loans"]
plt.figure(figsize=(14,6))
plt.plot(date, Total_People)
plt.title("Estimated Total People Getting Loans Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Total People Getting Loans Millions")
plt.xlim(clean_df["DATE"].min(),clean_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Analysis: To answer the question, "Are the number of people taking out loans increasing or decreasing?" our calculations of the estimated total number of people getting loans, show a sharp increase following the COVID-19 pandemic, followed by an equally significant decline in the subsequent quarters. Looking at these trends over the past 10 years, we can confidently conclude that the number of people taking out loans is rapidly decreasing.

Question 2 Analysis:

Has the number of people purchasing cars increased or decreased??

To answer this question, we analyzed Total Car Sales data over a 10-year period. However, we recognized that this data alone didn't fully address the question. With population growth across the U.S., simply calculating total car sales wouldn't account for the impact of population increases on car purchases. Instead, we focused on the average number of car purchases per person, as this approach would more accurately reflect trends in car buying behavior.

Our first step was to obtain population metrics. We initially attempted to source this data from the U.S. Census via API. However, upon reviewing the data, we encountered issues with the timeframe, as the available Census data only provided population figures for 2019. We attempted to use a "For" loop to populate the different years within our scope, but we quickly realized 2019 was the only viable year in this dataset.

```
# Initialize the Census object
url = f"https://api.census.gov/data/2019/pep/charagegroups?get=NAME,POP&for=us:*&key={census_key}"
response = requests.get(url)

census_data = response.json()

print (census_data)
```

✓ 0.7s

```
[['NAME', 'POP', 'us'], ['United States', '328239523', '1']]
```

We then turned to a secondary source, Data USA, to obtain the population data via API. We successfully retrieved the data and broke it down into quarterly segments.

```
url = "https://datausa.io/api/data?drilldowns=Nation&measures=Population"
data = requests.get(url).json()

years = []
population = []

for i in data['data']:
    years.append(i['Year'])
    population.append(i['Population'])

population_df = pd.DataFrame({"Year": years, "Population": population})

quarterly_years = []
quarterly_population = []

for i in range(1, len(population_df)):
    current_year = population_df.iloc[i]['Year']
    current_population = population_df.iloc[i]['Population']
    previous_population = population_df.iloc[i-1]['Population']


    yearly_increase = current_population - previous_population


    quarterly_increase = yearly_increase / 4


    for quarter in range(1, 5):

        population_at_quarter = previous_population + (quarterly_increase * quarter)
        quarterly_years.append(f"{current_year} Q{quarter}")
        quarterly_population.append(int(population_at_quarter))

quarterly_df = pd.DataFrame({"Year_Quarter": quarterly_years, "Population": quarterly_population})

quarterly_df
```

Unfortunately, upon reviewing our data frame, we realized the data ended in 2021. While we wanted to effectively utilize an API for this metric, we felt ending the data this early would fail to represent post COVID trends accurately.

	Year_Quarter	Population
0	2021 Q1	330754565
1	2021 Q2	330411537
2	2021 Q3	330068509
3	2021 Q4	329725481
4	2020 Q1	328936437
5	2020 Q2	328147394
6	2020 Q3	327358351
7	2020 Q4	326569308
8	2019 Q1	326101429
9	2019 Q2	325633551
10	2019 Q3	325165673
11	2019 Q4	324697795
12	2018 Q1	324249103
13	2018 Q2	323800412
14	2018 Q3	323351721
15	2018 Q4	322903030
16	2017 Q1	322428374
17	2017 Q2	321953718
18	2017 Q3	321479062
19	2017 Q4	321004407
20	2016 Q1	320392845
21	2016 Q2	319781284
22	2016 Q3	319169723
23	2016 Q4	318558162
24	2015 Q1	318047376
25	2015 Q2	317536591
26	2015 Q3	317025806
27	2015 Q4	316515021
28	2014 Q1	315913036
29	2014 Q2	315311052
30	2014 Q3	314709068
31	2014 Q4	314107084
32	2013 Q1	313464461
33	2013 Q2	312821839
34	2013 Q3	312179216
35	2013 Q4	311536594

Based on the results from our API attempts, we inevitably decided to source the data from FRED as we were able to get the annual population data within our desired timeframe. We pulled the file into our data set as indicated in our VS Code.

```
# Store filepaths into variables
total_asset_loan_value_csv = Path("Resources/Total Loan Assets.csv")
average_rate_of_financed_loan_csv = Path("Resources/Average Loan Finance.csv")
average_amount_financed_csv= Path("Resources/Average Amount Financed.csv")
total_vehicle_sales_csv = Path("Resources/Total Vehicle Sales.csv")
total_population_csv = Path("Resources/Total Population USA.csv")
```

Our first step was to rename the column in the data set to be clear and easier to utilize within our code and to link with pd.merge against “DATE” vs “Year”

```
total_population_df = total_population_df.rename(columns={"Year": "DATE"})
total_population_df.head()
```

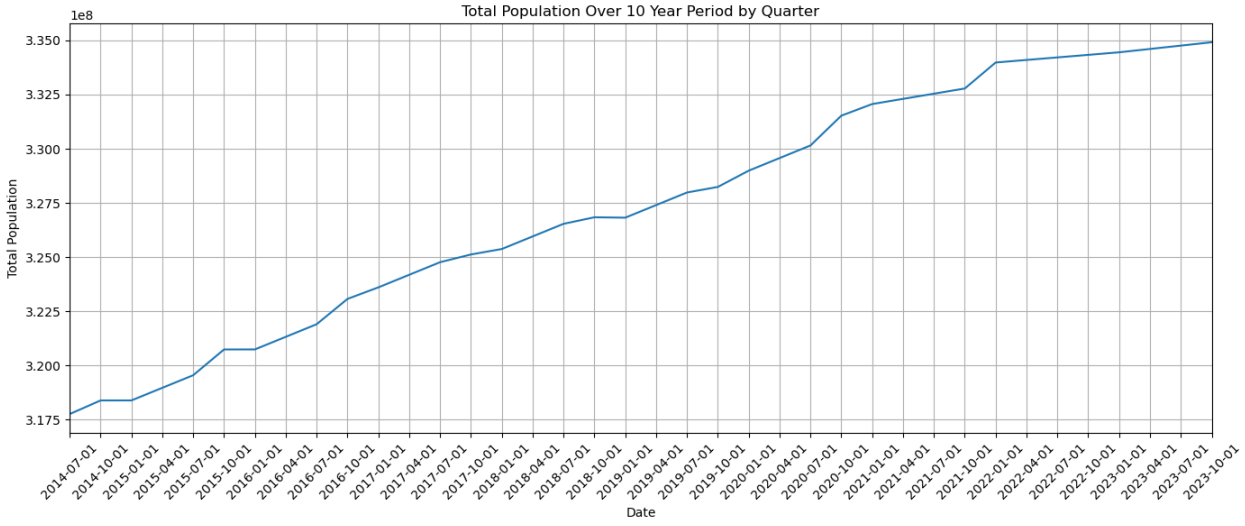
✓ 0.0s 🐼 Open 'total_population_df' in Data Wrangler

	DATE	Population
0	2014-07-01	317758058
1	2014-10-01	318386329
2	2015-01-01	318387509
3	2015-04-01	318968455
4	2015-07-01	319549401

To provide context for our total vehicle sales, we decided to include a graph showing the population increase over the same timeframe.

```
#Plot Total Population Over 10 Year Period by Quarter
date = total_population_df["DATE"]
Total_Population = total_population_df["Population"]
plt.figure(figsize=(14,6))
plt.plot(date, Total_Population)
plt.title("Total Population Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Total Population")
plt.xlim(total_population_df["DATE"].min(),total_population_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

✓ 0.3s



The following step was to incorporate Total Vehicle Sales into the data frame by cleaning the data: renaming the column, removing "NaN" values, and calculating the quarterly rate by multiplying the monthly rate by 1,000,000 as that was the accurate unit of measure indicated by the FRED dataset.

```
#Display the data
total_vehicle_sales_df.head()
```

✓ 0.0s Open 'total_vehicle_sales_df' in Data Wrangler

	DATE	TOTALSA
0	2014-07-01	51.475
1	2014-10-01	50.883
2	2015-01-01	51.697
3	2015-04-01	53.511
4	2015-07-01	54.990

```
#Rename Columns
total_vehicle_sales_df= total_vehicle_sales_df.rename(columns={"TOTALSA": "Total Vehicle Sales"})
total_vehicle_sales_df.head()
```

✓ 0.0s Open 'total_vehicle_sales_df' in Data Wrangler

	DATE	Total Vehicle Sales
0	2014-07-01	51.475
1	2014-10-01	50.883
2	2015-01-01	51.697
3	2015-04-01	53.511
4	2015-07-01	54.990

```
#Convert to millions
total_vehicle_sales_df = pd.DataFrame(total_vehicle_sales_df)
total_vehicle_sales_df["Total Vehicle Sales Millions"] = total_vehicle_sales_df["Total Vehicle Sales"] * 1000000
total_vehicle_sales_df = total_vehicle_sales_df.dropna()
total_vehicle_sales_df["Total Vehicle Sales Millions"] = total_vehicle_sales_df["Total Vehicle Sales Millions"].astype(int)
total_vehicle_sales_df.head()
```

✓ 0.0s Open 'total_vehicle_sales_df' in Data Wrangler

	DATE	Total Vehicle Sales	Total Vehicle Sales Millions
0	2014-07-01	51.475	51475000
1	2014-10-01	50.883	50883000
2	2015-01-01	51.697	51697000
3	2015-04-01	53.511	53511000
4	2015-07-01	54.990	54990000

Similarly in our previous data frame creation, to create a complete data frame for analysis, we began merging the data using the DATE column with the pd.merge function.

```
#Merge the dataframes
avg_car_per_person_df = pd.merge(total_population_df, total_vehicle_sales_df, on="DATE")
avg_car_per_person_df.head()
```

✓ 0.0s Open 'avg_car_per_person_df' in Data Wrangler

	DATE	Population	Total Vehicle Sales	Total Vehicle Sales Millions
0	2014-07-01	317758058	51.475	51475000
1	2014-10-01	318386329	50.883	50883000
2	2015-01-01	318387509	51.697	51697000
3	2015-04-01	318968455	53.511	53511000
4	2015-07-01	319549401	54.990	54990000

We can now calculate the Average Car Sales Per Person by dividing the total vehicle sales (in millions) by the population for each corresponding period. This calculation allows us to determine the average number of cars sold per person, providing a clearer view of car purchasing trends in relation to population growth over time. By examining this metric, we can better understand whether car purchases are increasing or decreasing on a per capita basis, accounting for changes in population.

```
avg_car_per_person_df["Average Car Sales Per Person"] = avg_car_per_person_df["Total Vehicle Sales Millions"] / avg_car_per_person_df["Population"]
avg_car_per_person_df["Average Car Sales Per Person"] = avg_car_per_person_df["Average Car Sales Per Person"].astype(float)
avg_car_per_person_df.head()
```

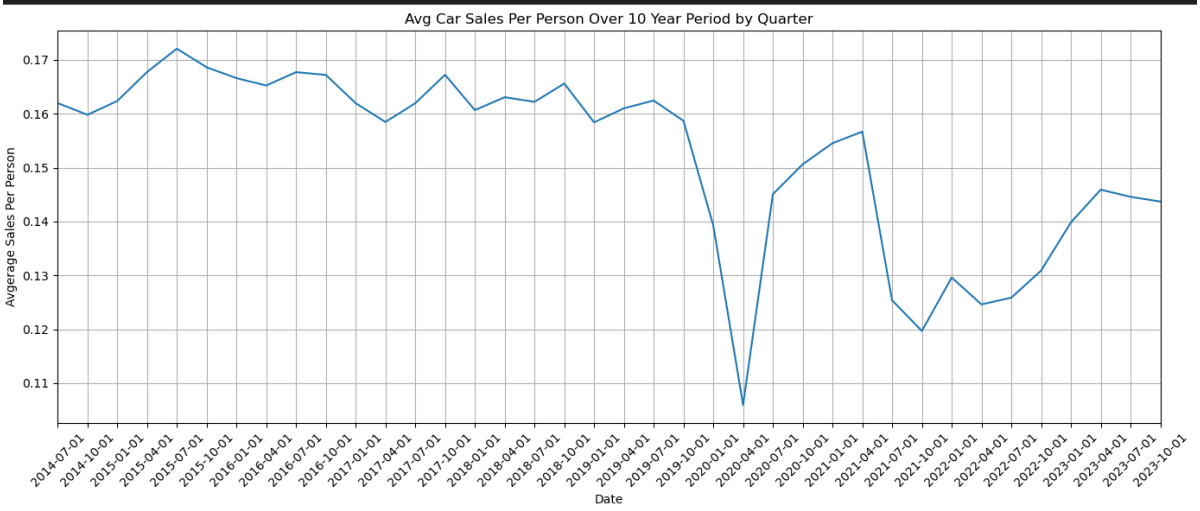
✓ 0.0s Open 'avg_car_per_person_df' in Data Wrangler

	DATE	Population	Total Vehicle Sales	Total Vehicle Sales Millions	Average Car Sales Per Person
0	2014-07-01	317758058	51.475	51475000	0.161994
1	2014-10-01	318386329	50.883	50883000	0.159815
2	2015-01-01	318387509	51.697	51697000	0.162371
3	2015-04-01	318968455	53.511	53511000	0.167763
4	2015-07-01	319549401	54.990	54990000	0.172086

With the newly calculated metrics for Average Car Sales Per Person, we can now plot the data to draw effective insights.

```
#Plot Avg Car sales per person Over 10 Year Period by Quarter
date = avg_car_per_person_df["DATE"]
Avgerage_Sales_Per_Person = avg_car_per_person_df["Average Car Sales Per Person"]
plt.figure(figsize=(14,6))
plt.plot(date, Avgerage_Sales_Per_Person)
plt.title("Avg Car Sales Per Person Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Avgerage Sales Per Person")
plt.xlim(avg_car_per_person_df["DATE"].min(),avg_car_per_person_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

✓ 0.4s

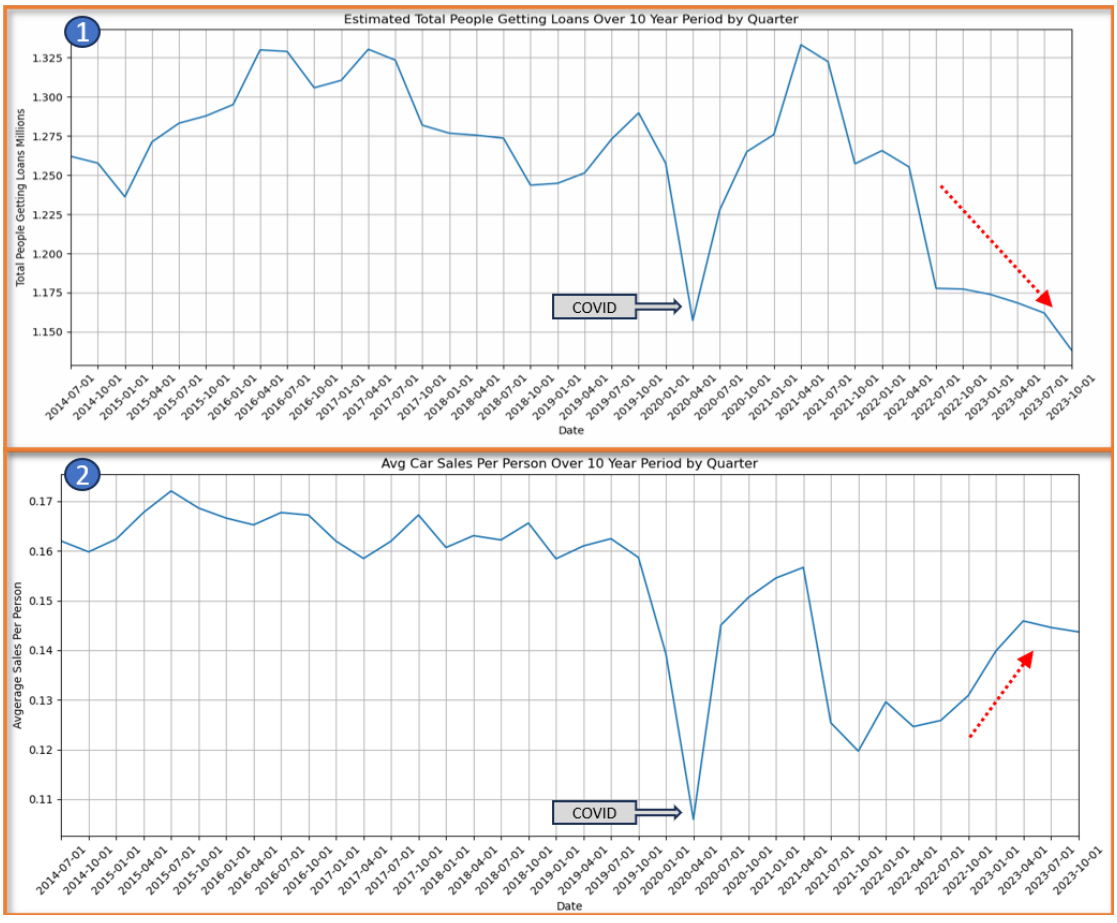


Analysis: To answer the question, "Has the number of people purchasing cars increased or decreased?" our calculations of the Average Car Sales Per Person show a sharp increase following the COVID-19 pandemic. Unlike the trend observed in loan data, which saw a continued decline after an initial drop, car purchases experienced a slight decline in the subsequent quarters before rebounding and trending upwards. Analyzing these trends over the past 10 years, we can confidently conclude that the average number of car purchases per person is increasing and returning to pre-pandemic levels.

Question 3 Analysis:

<u>Do trends in loan approvals correlate with trends in car purchases?</u>

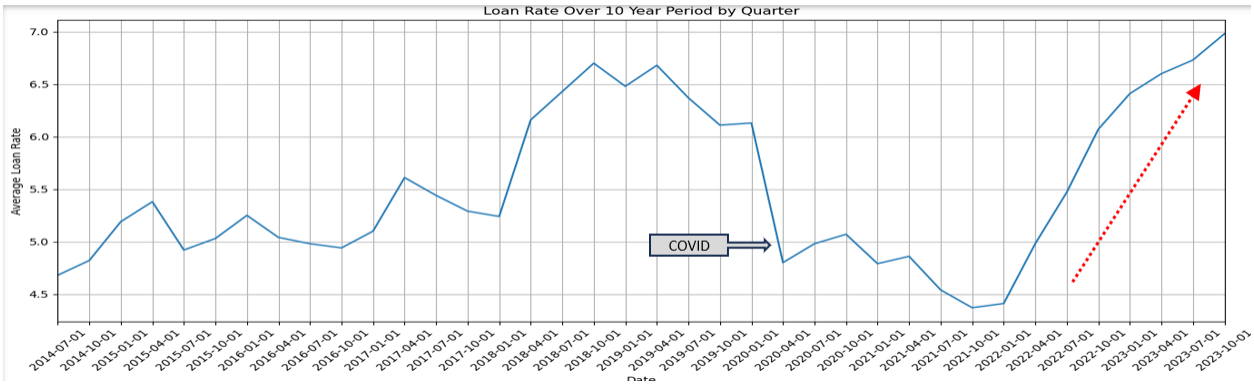
Based on the graphs we created to address our first two questions; we sought to analyze why the Estimated Total People Getting Loans did not follow the same pattern as the Average Car Sales Per Person. Initially, we hypothesized that these two metrics would show similar movements. However, upon reviewing the post-pandemic data, we observed a contrasting trend and realized further investigation was needed to understand the factors driving this opposite correlation.

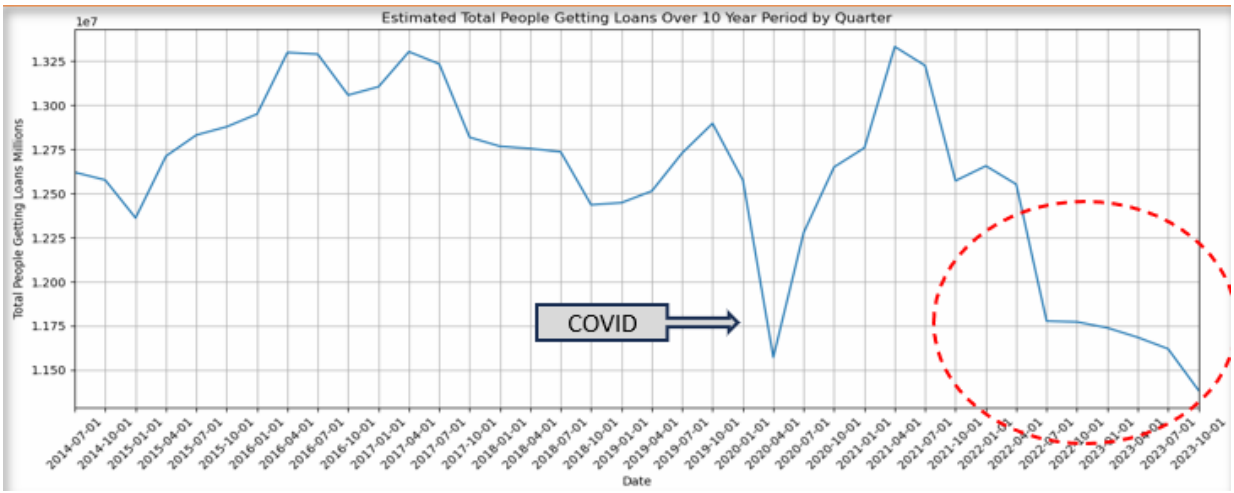


It didn't take long to identify the disparity between car buying trends and auto loan acquisition, and we concluded that fluctuating loan rates were likely a significant factor influencing this difference.

We decided to plot the Loan Rate over the 10-year period to examine the data further to see if it supports our hypothesis.

```
date = clean_df["DATE"]
Average_Loan_Rate = clean_df["Average Loan Rate"]
plt.figure(figsize=(14,6))
plt.plot(date, Average_Loan_Rate)
plt.title("Loan Rate Over 10 Year Period by Quarter")
plt.xlabel("Date")
plt.ylabel("Average Loan Rate")
plt.xlim(clean_df["DATE"].min(),clean_df["DATE"].max())
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```





After comparing the graphs together, we can clearly identify the periods of increasing rates typically corresponding to a decrease in the number of people obtaining auto loans. Making this the primary factor for people deciding to get loans.

Analysis: To answer the question, "Do trends in loan approvals correlate with trends in car purchases?" we observed similar patterns in our calculations of Average Car Sales Per Person and Estimated Total People Getting Loans before the COVID-19 pandemic. However, as loan rates began to vary post-pandemic, we saw a direct correlation between these changes and our Estimated Total People Getting Loans, while Average Car Sales followed a different trajectory. Although we initially assumed that car purchases would influence the number of people taking out loans, we concluded that loan rates played a more significant role in shaping the trend for this particular metric.

Conclusion

In conclusion, despite the current high inflation, there is a noticeable trend towards consumers opting to purchase cars outright rather than relying on long-term loans. As highlighted in the summary, the correlation between loan approvals and car purchases indicates that loan rates significantly influence the decision to finance vehicle purchases through loans.

This relationship is particularly evident in the post-COVID period, where loan amounts dropped to their lowest point, coinciding with a substantial increase in loan applications. Similarly, following the first quarter of 2022, as loan rates increased, there was a corresponding decline in the number of consumers choosing loan options over outright purchases.

Given these observations, it is reasonable to anticipate that as loan rates continue to rise, the trend of declining loan applications will persist. To mitigate this and generate additional revenue from loan services, car manufacturers would be well-advised to consider reducing interest rates to make long-term loan options more attractive to potential buyers.