Ensuring Successful Retirement

Group 2

Our
Main Question

Hana W.

How do different circumstances affect the success of Americans' retirements?

How would we measure a retirement's success?

Earlier Age of Retirement?

No, lacking financial security.

- FIRE versus Barista-FIRE

Benefits from Social Security

No, lacking financial security.

- Benefits =\= Pre-Retirement Income

Income and Spending?

Yes, measuring financial security.

Strategy and Metrics for Our Source

“success of. . .retirement”

One variable at least…

…should show income and spending.

“differents circumstances”

Variables…

…should be plentiful and distinct from one another.

“Americans”’

Set’s sample…

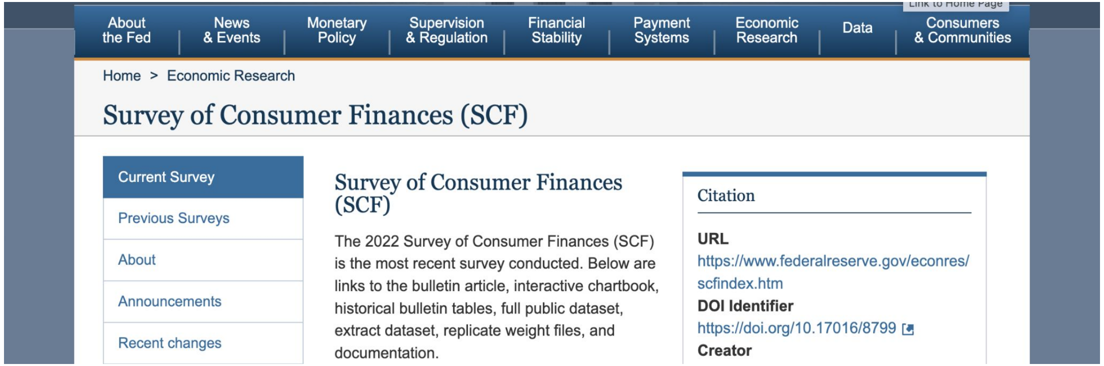
…should be large.

Cleaning the Data

Amanda D.

- Source Data
- Jupyter Notebook
- Final Output

Source Data: www.federalreserve.gov



Original Dataset

	YY1	Y1	WGT	HHSEX	AGE	AGECL	EDUC	EDCL	MARRIED	KIDS	...	NWCAT	INCCAT	A
0	1	11	3027.956120	2	70	5	9	3	2	2	...	4	2	
1	1	12	3054.900065	2	70	5	9	3	2	2	...	4	2	
2	1	13	3163.637766	2	70	5	9	3	2	2	...	4	2	
3	1	14	3166.228463	2	70	5	9	3	2	2	...	3	2	

Consolidated Dataframe

```
data_df = data[['AGE',
                'AGECL',
                'EDCL',
                'FAMSTRUCT',
                'KIDS',
```

Jupyter Notebook

.rename ( ) method

```
#Rename column headers from Code to Description
data_df = data_df.rename(columns={"AGE": "Age",
                                  "AGECL": "Age Group",
                                  "EDCL": "Education",
```

mapping dictionary and .replace ( ) method

```
# Map row numerical codes to descriptions
age_map = {1: "< 35", 2: "35-44", 3: "45-54", 4: "55-64", 5: "65-74", 6: ">=75"}
education_map = {1: "no high school", 2: "high school diploma or GED", \
                 3: "some college or Assoc. degree", \
                 4: "Bachelors degree or higher"}
family_structure_map = {1: "not married/LWP + children", \
                        2: "not married/LWP + children + reference person under 55", \
                        3: "not married/LWP + no children + reference person 55 or older", \
                        4: "married/LWP+ children", 5: "married/LWP + no children"}
```

```
#Replace row numberical codes to descriptions
data_df['Age Group'] = data_df['Age Group'].replace(age_map)
data_df['Education'] = data_df['Education'].replace(education_map)
data_df['Family Structure'] = data_df['Family Structure'].replace(family_structure_map)
```

mapping dictionary and .replace ( ) method within a for loop

```
#Create mapping dictionary for Yes / No columns and apply to specified columns
columns_to_convert = ['Household declared bankruptcy, last 5 yrs', \
                      'Respondant had foreclosure, last 5 yrs', \
                      'Pension exists for reference person/spouse', \
```

```
yes_no_mapping = {1: 'Yes', 0: 'No'}

for column in columns_to_convert:
    data_df[column] = data_df[column].replace(yes_no_mapping)
```

Find_reasons function and .apply ( ) method to consolidate multiple columns into a new column

```
#Create a new column for Reason for Saving and consolidate 5 separate Yes/No
#reason for savings columns into one with only the reason.
reason_columns = ["Can't Save", 'Saving for Home', 'Saving for Retirement', \
                  'Saving for Liquidity/Future', 'Saving for Investment']

#Check for yes columns to find reason for saving. If no reason listed, add "None Listed".
def find_reasons(row):
    reasons = []
    for col in reason_columns:
        if row[col] == 'Yes':
            reasons.append(col)
    if not reasons:
        return 'None Listed'
    return ', '.join(reasons)
```

Original Dataset

	YY1	Y1	WGT	HHSEX	AGE	AGECL	EDUC	EDCL	MARRIED	KIDS	...	NWCAT	INCCAT	A
0	1	11	3027.956120	2	70	5	9	3	2	2	...	4	2	
1	1	12	3054.900065	2	70	5	9	3	2	2	...	4	2	
2	1	13	3163.637766	2	70	5	9	3	2	2	...	4	2	
3	1	14	3166.228463	2	70	5	9	3	2	2	...	3	2	

Final Dataset

	Age	Age Group	Education	Family Structure	Kids	Marital Status	Occupation Category	Occupation Class	Race/Ethnicity	Life Cycle
0	70	65-74	some college or Assoc. degree	not married/LWP + children	2	neither married nor living with partner	retired/disabled + (student/homemaker/misc. no...	not working	white non-Hispanic	55 or older and not working
1	70	65-74	some college or Assoc. degree	not married/LWP + children	2	neither married nor living with partner	retired/disabled + (student/homemaker/misc. no...	not working	white non-Hispanic	55 or older and not working
2	70	65-74	some college or Assoc. degree	not married/LWP + children	2	neither married nor living with partner	retired/disabled + (student/homemaker/misc. no...	not working	white non-Hispanic	55 or older and not working

Our Graphs, Part #1

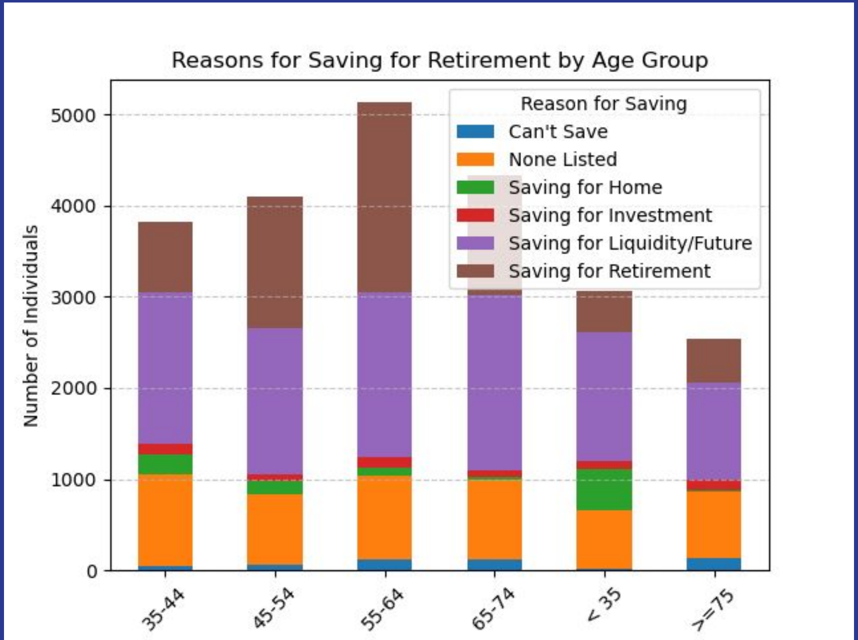
Jessica V.

Retrieving the Data:

- The following slides primarily describe aspects related to retirement savings behaviors and their correlation with different demographics

1^{st} Graph:

Which age group saves the most for retirement?



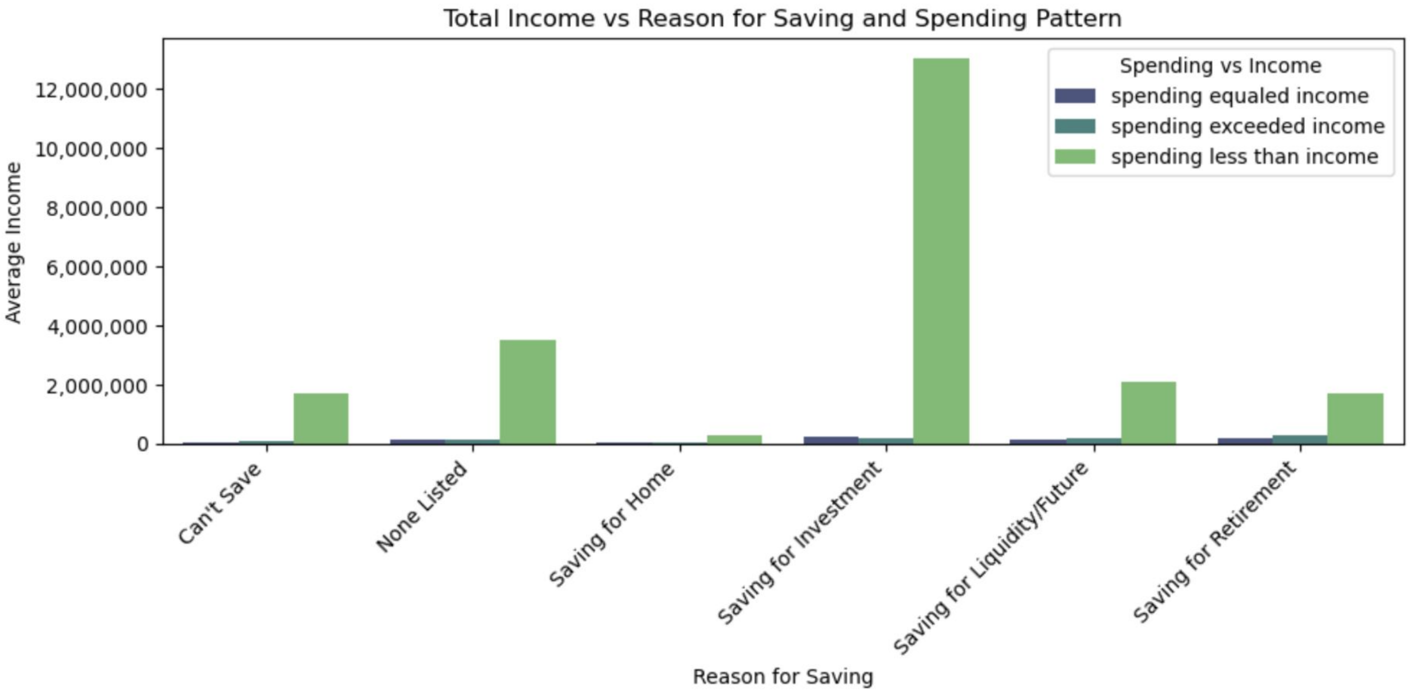
The code for Graph #1

```
#Reasons for savings for retirement by age group
age_reason_counts = data_df.groupby(['Age Group', 'Reason for Saving']).size().unstack().fillna(0)

plt.figure(figsize=(12, 7))
age_reason_counts.plot(kind='bar', stacked=True)
plt.title('Reasons for Saving for Retirement by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Individuals')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='Reason for Saving')
plt.show()
```

Jupyter Notebook

2nd Graph:

How does total income vary by reason for saving and spending patterns?



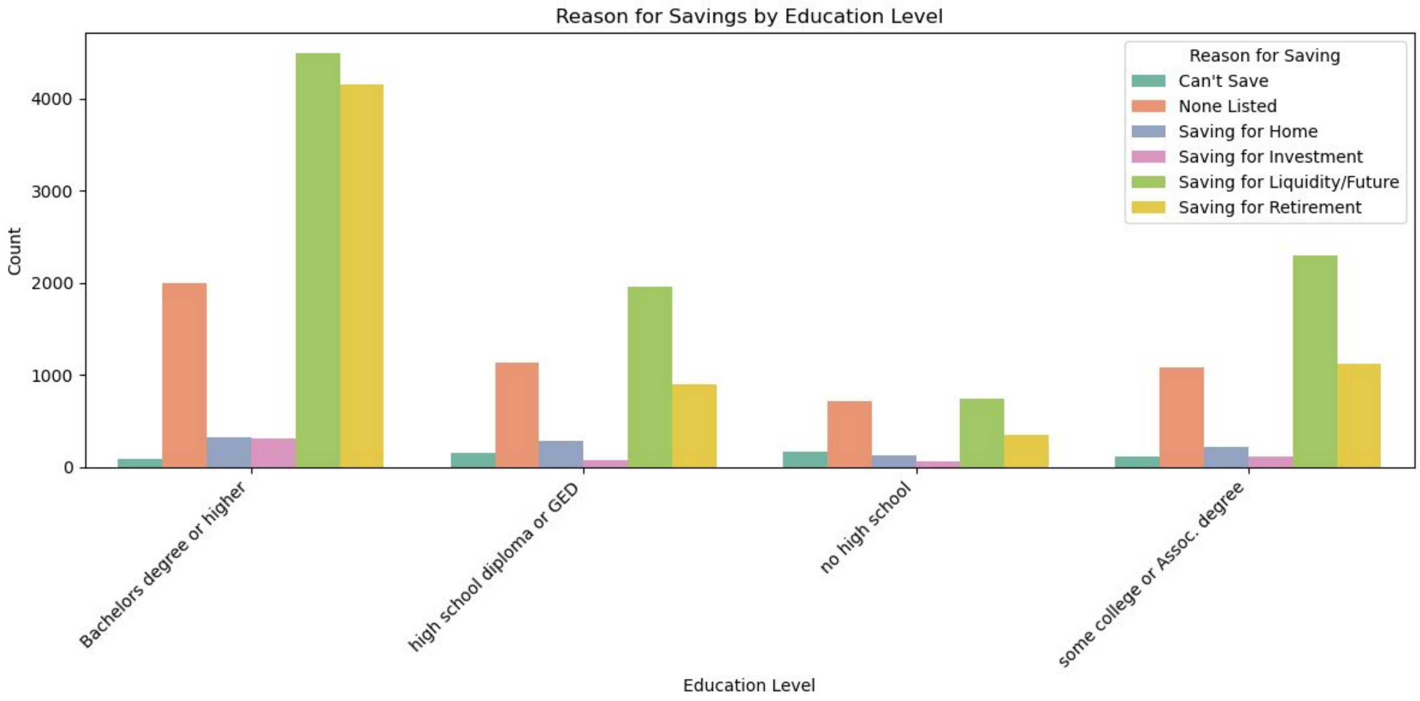
The code for Graph #2

```
# Total income versus reason for saving and spending pattern
plot_df = data_df.groupby(['Reason for Saving', 'Spending vs Income, last 12 mo'])['Total Income 2019'].mean().reset_index()


plt.figure(figsize=(10, 5))
sns.barplot(data=plot_df, x='Reason for Saving', y='Total Income 2019', hue='Spending vs Income, last 12 mo', palette='viridis')
plt.title('Total Income vs Reason for Saving and Spending Pattern')
plt.xlabel('Reason for Saving')
plt.ylabel('Average Income')
plt.xticks(rotation=45, ha='right')

ax = plt.gca()  # Get current axis
ax.get_yaxis().set_major_formatter(mtick.FuncFormatter(lambda x, p: format(int(x), ',')))

plt.legend(title='Spending vs Income')
plt.tight_layout()
plt.show()
```

Jupyter Notebook

3rd Graph:

How do reasons for saving vary across different education levels?



The code for Graph #3

```
#Education versus reason for savings
education_saving_df = data_df.groupby(['Education', 'Reason for Saving']).size().reset_index(name='Count')

plt.figure(figsize=(12, 6))
sns.barplot(data=education_saving_df, x='Education', y='Count', hue='Reason for Saving', palette='Set2')

plt.title('Reason for Savings by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Reason for Saving')
plt.tight_layout()

plt.show()
```

Jupyter Notebook

Our Graphs, Part #2

Rebekah R.

Retrieving the Data:

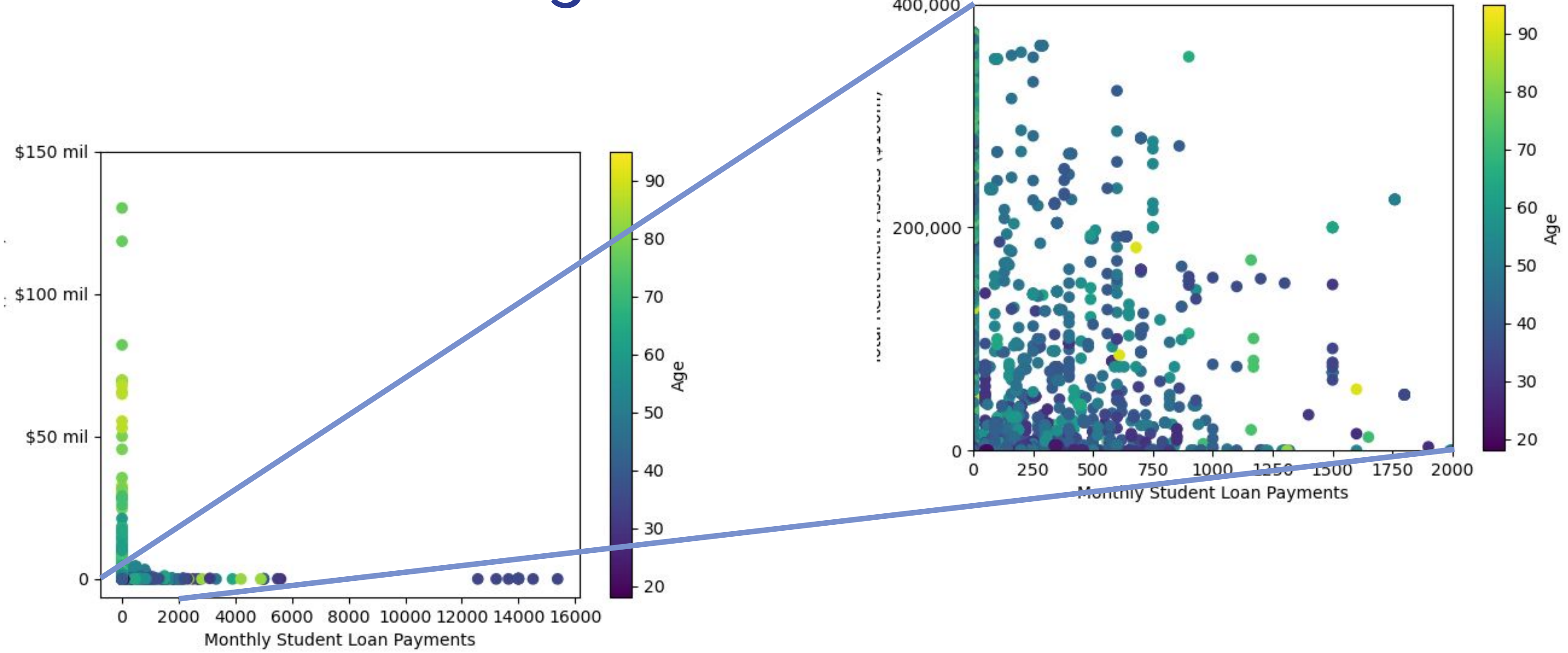
- All retirement figures represent quasi-liquid retirement assets only: No physical assets or checking/saving accounts are included in “retirement” figures.

The Outliers

```python
# Data Wrangling: Find and remove outliers
# Original Dataset max/min:
max_ret = max(data_df['Total Equity: Quasi-Liquid Retirement Assets 2019'])
min_ret = min(data_df['Total Equity: Quasi-Liquid Retirement Assets 2019'])

# Finding the Quartiles & IQR
quartiles = data_df['Total Equity: Quasi-Liquid Retirement Assets 2019'].quantile([.25, .5, .75])
lowerq = quartiles[0.25]
upperq = quartiles[0.75]
iqr = upperq - lowerq

# Finding the Bounds
lower_bound = lowerq - (1.5 * iqr)
upper_bound = upperq + (1.5 * iqr)

# Finding the outliers
outliers = data_df[(data_df['Total Equity: Quasi-Liquid Retirement Assets 2019'] < lower_bound) | \
    (data_df['Total Equity: Quasi-Liquid Retirement Assets 2019'] > upper_bound)]

# Defining a new dataframe without the outliers
final_data_df = data_df[~data_df['Total Equity: Quasi-Liquid Retirement Assets 2019']\
    .isin(outliers['Total Equity: Quasi-Liquid Retirement Assets 2019'])]

# Display the original min/max as compared to the new bounds
print(f"Original Min Retirement: ${min_ret:,.2f};  Original Max Retirement: ${max_ret:,.2f}")
print(f"Lower Bound: ${lower_bound:,.2f};  Upper Bound:  ${upper_bound:,.2f}")
```

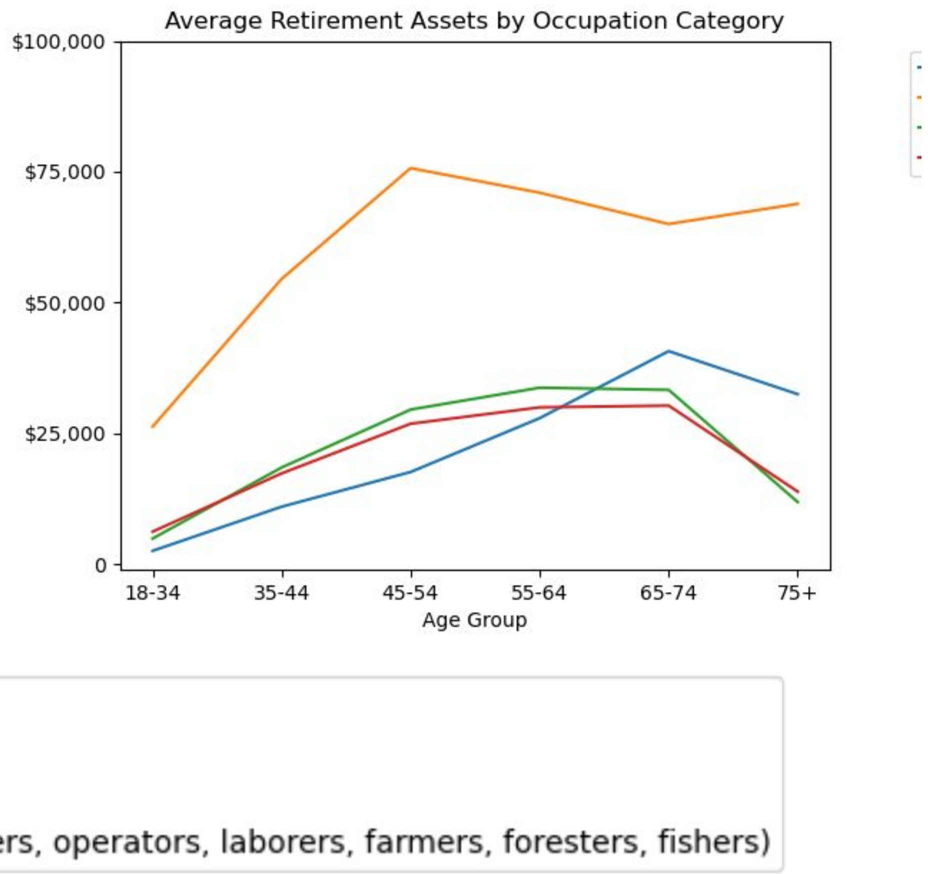
```
Original Min Retirement: $0.00;  Original Max Retirement: $130,122,000.00
Lower Bound: $-225,000.00;  Upper Bound:  $375,000.00
```

4th Graph: How do student loan payments affect retirement savings?



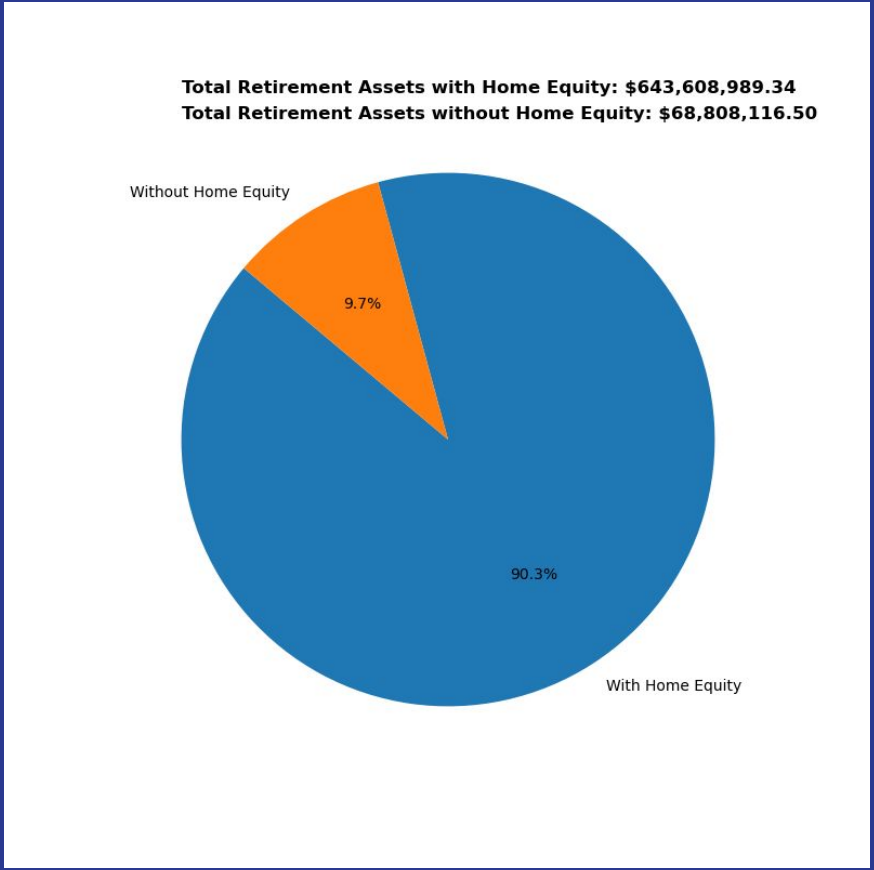
5th Graph:

How does occupation type affect retirement savings?

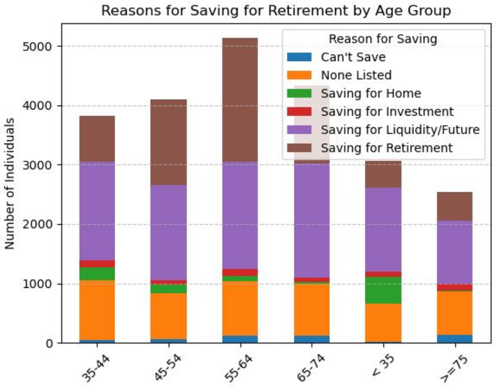


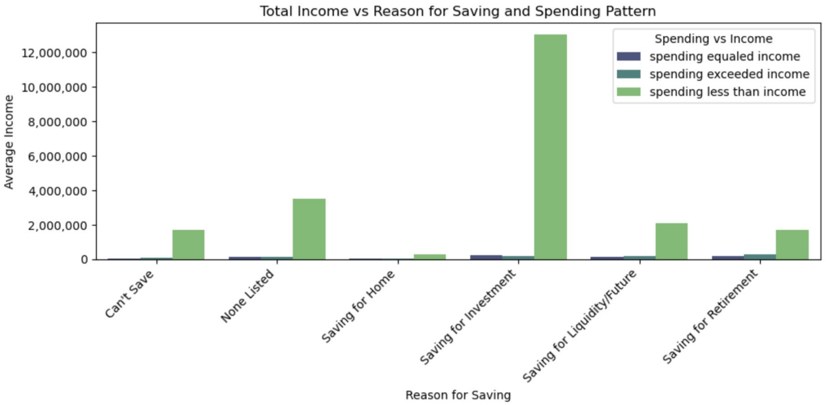
6^{th} Graph:

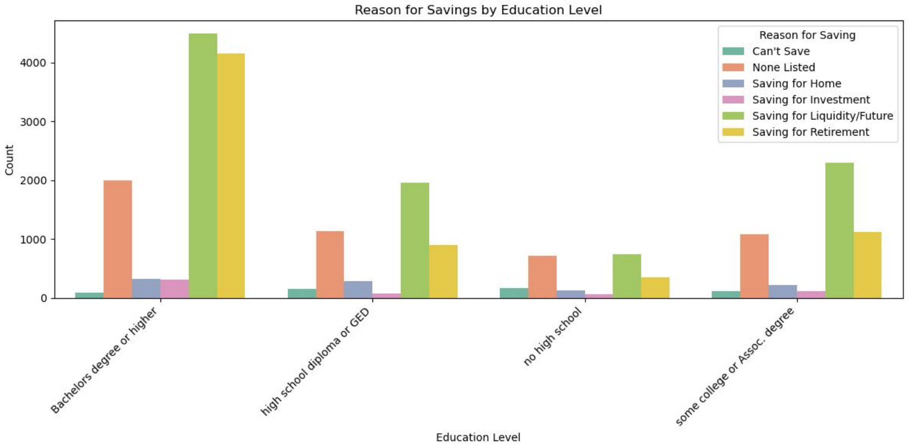
How does home ownership affect total retirement savings?



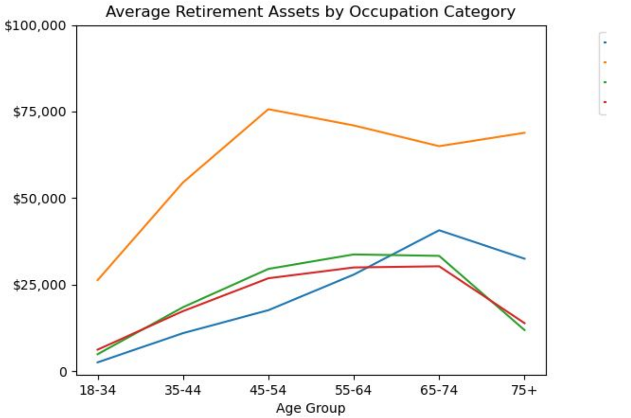
Agustín R.'s Analysis, Part #1

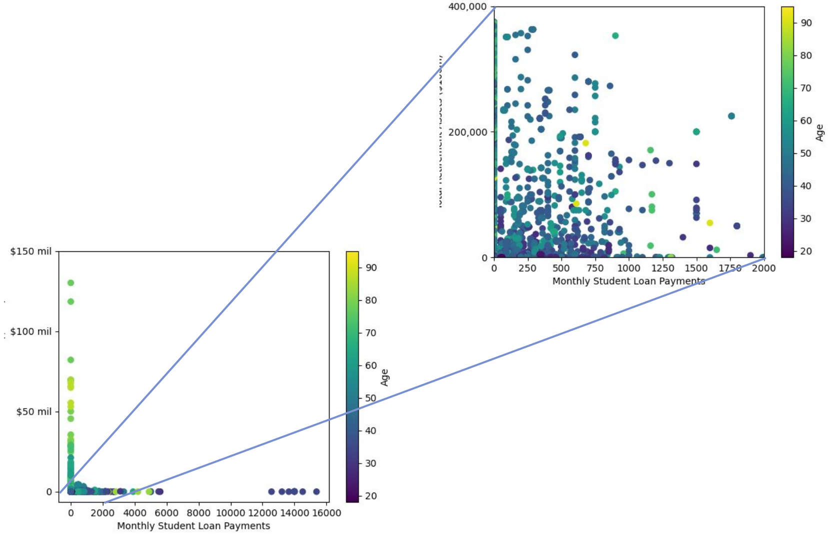


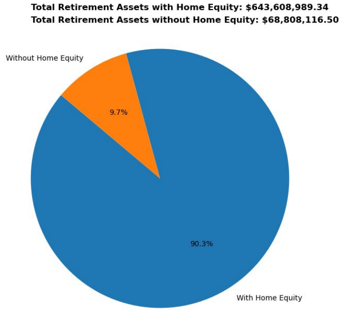




Agustín R.’s Analysis, Part #2







Our Conclusions

Hana W.

- Findings' Implications

Thanks for listening to our presentation!