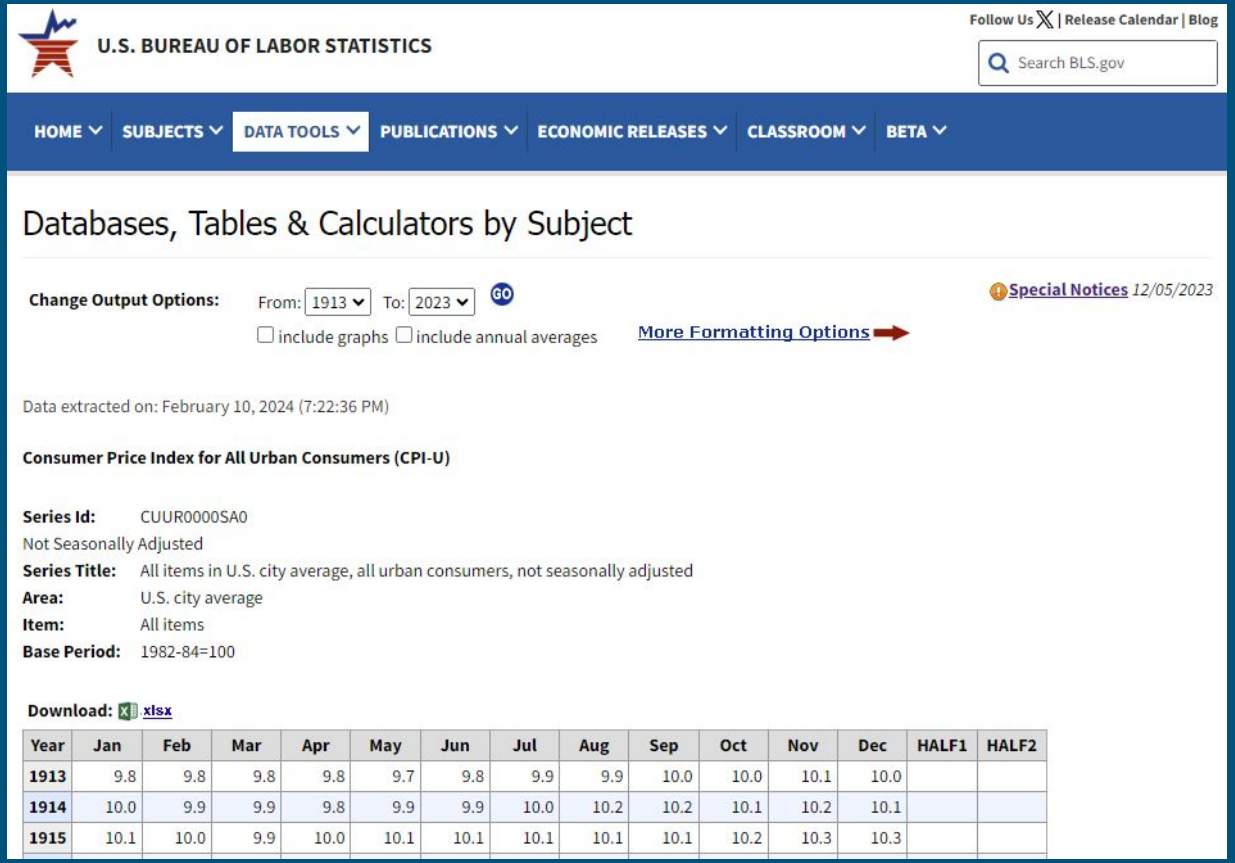
How much money do you need to retire today?

Project 4

Grant Amundson, Shruti Palur Mallampalli,
Xuexuan Xu and Vanessa Dumlao

Consumer Price Index aka CPI



U.S. BUREAU OF LABOR STATISTICS

Follow Us | Release Calendar | Blog

Search BLS.gov

HOME | SUBJECTS | DATA TOOLS | PUBLICATIONS | ECONOMIC RELEASES | CLASSROOM | BETA

Databases, Tables & Calculators by Subject

Change Output Options: From: 1913 To: 2023 GO

include graphs include annual averages More Formatting Options

Special Notices 12/05/2023

Data extracted on: February 10, 2024 (7:22:36 PM)

Consumer Price Index for All Urban Consumers (CPI-U)

Series Id: CUUR0000SA0
Not Seasonally Adjusted
Series Title: All items in U.S. city average, all urban consumers, not seasonally adjusted
Area: U.S. city average
Item: All items
Base Period: 1982-84=100

Download: xlsx

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	HALF1	HALF2
1913	9.8	9.8	9.8	9.8	9.7	9.8	9.9	9.9	10.0	10.0	10.1	10.0		
1914	10.0	9.9	9.9	9.8	9.9	9.9	10.0	10.2	10.2	10.1	10.2	10.1		
1915	10.1	10.0	9.9	10.0	10.1	10.1	10.1	10.1	10.1	10.2	10.3	10.3		

Period Life Table

Downloadable Files — Period Life Tables

*(Projected values contained in these files are based on the intermediate assumptions of the **2017 Trustees Report**)*

In each file, the first column gives the year, the second column gives the age, and the remaining columns are the life table values.

Click on the "Text" link to get a preview of the file.

Males

- Historical (1900-2014)
  Text, CSV, Excel
- Projected (2015-2095)
  Text, CSV, Excel

Females

- Historical (1900-2014)
  Text, CSV, Excel
- Projected (2015-2095)
  Text, CSV, Excel

Period Life Table, 2020, as used in the 2023 Trustees Report

Exact age	Male			Female		
	Death probability [a]	Number of lives [b]	Life expectancy	Death probability [a]	Number of lives [b]	Life expectancy
0	0.005837	100,000	74.12	0.004907	100,000	79.78
1	0.000410	99,416	73.55	0.000316	99,509	79.17
2	0.000254	99,376	72.58	0.000196	99,478	78.19
3	0.000207	99,350	71.60	0.000160	99,458	77.21
4	0.000167	99,330	70.62	0.000129	99,442	76.22
5	0.000141	99,313	69.63	0.000109	99,430	75.23
6	0.000123	99,299	68.64	0.000100	99,419	74.24
7	0.000113	99,287	67.65	0.000096	99,409	73.25

Cost of Living by State

Cost of Living & Disposable Incomes by State

Search in table

Page 1 of 5

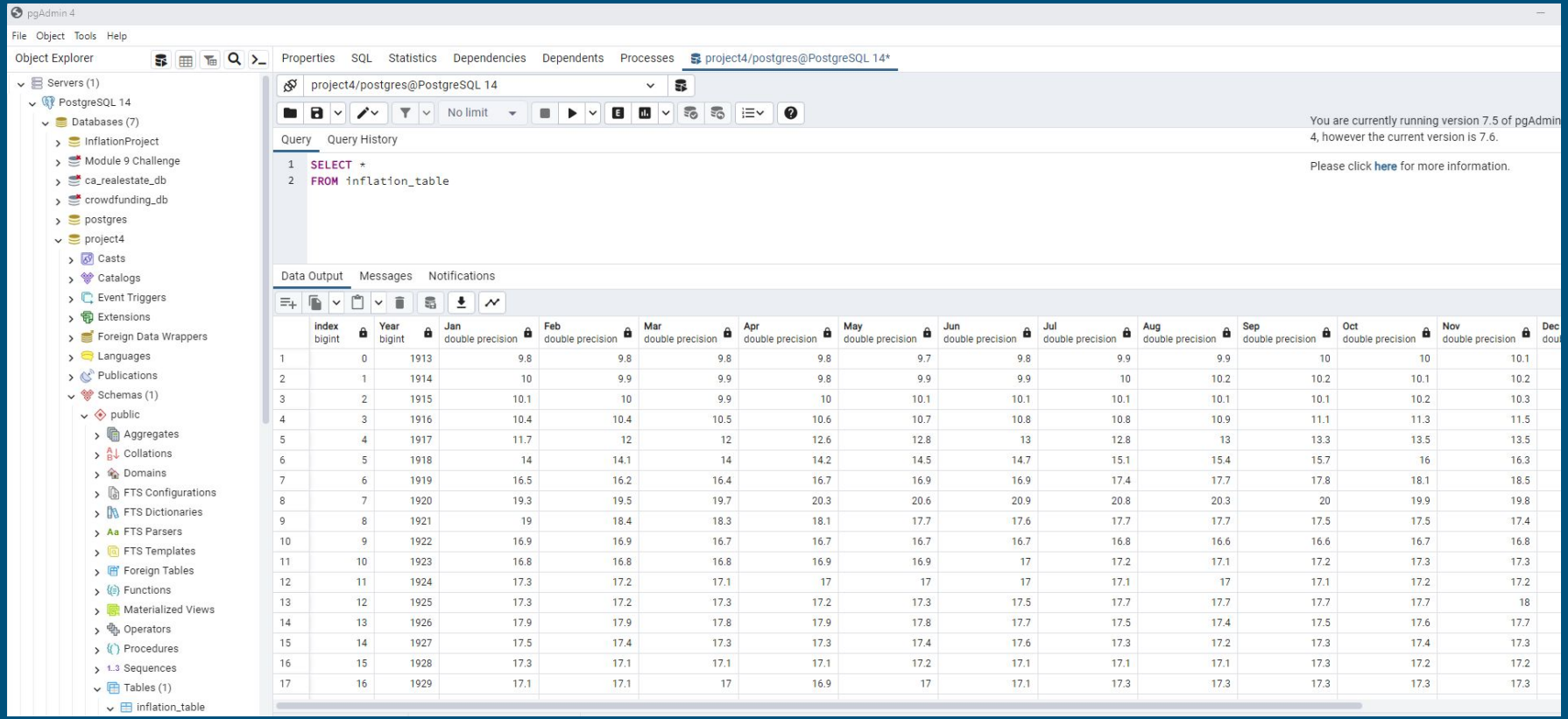
State	Total Cost of Living[1]	Total Cost Of Living Ranking	Total Disposable Income[2]	Disposable Income Ranking
Hawaii	$55,491	1	$5,929	50
Massachusetts	$53,860	2	$22,740	3
California	$53,171	3	$20,049	15
New York	$49,623	4	$25,247	1
New Jersey	$49,511	5	$21,379	10
Alaska	$48,670	6	$17,460	29
Maryland	$48,235	7	$21,515	9
Washington	$47,231	8	$25,119	2
Connecticut	$46,912	9	$22,398	6
Oregon	$46,193	10	$16,487	36

1. To determine the total cost of living, we factored in yearly expenses for housing, healthcare, taxes, food and transportation. Data sources include C2ER, KFF, MIT Living Wage Calculator and the U.S. Census Bureau.
2. To calculate the disposable income, we subtracted the total cost of living from the average salary.

Source: Forbes Advisor Analysis • Get the data • Embed

Forbes ADVISOR

Postgres Project4: inflation_table



	index bigint	Year bigint	Jan double precision	Feb double precision	Mar double precision	Apr double precision	May double precision	Jun double precision	Jul double precision	Aug double precision	Sep double precision	Oct double precision	Nov double precision	Dec
1	0	1913	9.8	9.8	9.8	9.8	9.7	9.8	9.9	9.9	10	10	10.1	
2	1	1914	10	9.9	9.9	9.8	9.9	9.9	10	10.2	10.2	10.1	10.2	
3	2	1915	10.1	10	9.9	10	10.1	10.1	10.1	10.1	10.1	10.2	10.3	
4	3	1916	10.4	10.4	10.5	10.6	10.7	10.8	10.8	10.9	11.1	11.3	11.5	
5	4	1917	11.7	12	12	12.6	12.8	13	12.8	13	13.3	13.5	13.5	
6	5	1918	14	14.1	14	14.2	14.5	14.7	15.1	15.4	15.7	16	16.3	
7	6	1919	16.5	16.2	16.4	16.7	16.9	16.9	17.4	17.7	17.8	18.1	18.5	
8	7	1920	19.3	19.5	19.7	20.3	20.6	20.9	20.8	20.3	20	19.9	19.8	
9	8	1921	19	18.4	18.3	18.1	17.7	17.6	17.7	17.7	17.5	17.5	17.4	
10	9	1922	16.9	16.9	16.7	16.7	16.7	16.7	16.8	16.6	16.6	16.7	16.8	
11	10	1923	16.8	16.8	16.8	16.9	16.9	17	17.2	17.1	17.2	17.3	17.3	
12	11	1924	17.3	17.2	17.1	17	17	17	17.1	17	17.1	17.2	17.2	
13	12	1925	17.3	17.2	17.3	17.2	17.3	17.5	17.7	17.7	17.7	17.7	18	
14	13	1926	17.9	17.9	17.8	17.9	17.8	17.7	17.5	17.4	17.5	17.6	17.7	
15	14	1927	17.5	17.4	17.3	17.3	17.4	17.6	17.3	17.2	17.3	17.4	17.3	
16	15	1928	17.3	17.1	17.1	17.1	17.2	17.1	17.1	17.1	17.3	17.2	17.2	
17	16	1929	17.1	17.1	17	16.9	17	17.1	17.3	17.3	17.3	17.3	17.3	

Pulling data from sql

Preparing the data

```python
# Create a list of date that represent the date the dataframe has
years = df_CPI['Year']
date = []
thirty_days_months = [4, 6, 9, 11]
for year in years:
    # The reason we assign it to the last day of the month is for the prophet model
    # since the prophet model has its own function for create future date
    # and the future dates are the last day of the month
    for j in range(1,13):
        if j == 2 and (year-1912) % 4 ==0:
            date.append(datetime.date(year, j, 29))
        elif j == 2 and (year-1912) % 4 !=0:
            date.append(datetime.date(year, j, 28))
        elif j in thirty_days_months:
            date.append(datetime.date(year, j, 30))
        else:
            date.append(datetime.date(year, j, 31))
```

✓ 0.0s Python

```python
# Get the CPI values as a list
cpi_value= np.array([df_CPI.drop(columns=["index","Year", "Annual", "HALF1", "HALF2"])]).reshape(-1,1)
```

✓ 0.0s Python

```python
# Create a reorganize dataframe for model use
df_reorganized_CPI = pd.DataFrame(date, columns=["Date"])
df_reorganized_CPI["CPI Value"] = cpi_value
df_reorganized_CPI
```

✓ 0.0s Python

	Date	CPI Value
0	1913-01-31	9.800
1	1913-02-28	9.800
2	1913-03-31	9.800
3	1913-04-30	9.800
4	1913-05-31	9.700
...	...	...
1327	2023-08-31	307.026
1328	2023-09-30	307.789
1329	2023-10-31	307.671
1330	2023-11-30	307.051
1331	2023-12-31	306.746

1332 rows × 2 columns

```python
# Check the data type
df_reorganized_CPI.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1332 entries, 0 to 1331
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1332 non-null   object
 1   CPI Value  1332 non-null   float64
dtypes: float64(1), object(1)
memory usage: 20.9+ KB
```

```python
# Transform the date value to datetime form
df_reorganized_CPI["Date"] = pd.DatetimeIndex(df_reorganized_CPI["Date"])
```

✓ 0.0s Python

```python
# Check the data type
df_reorganized_CPI.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1332 entries, 0 to 1331
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1332 non-null   datetime64[ns]
 1   CPI Value  1332 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 20.9 KB
```

```python
# Rename the columns for the Prophet model
df_reorganized_CPI = df_reorganized_CPI.rename(columns={"Date": "ds", "CPI Value": "y"})
```

✓ 0.0s Python

Prophet

Build the Prophet Model with Train data

```python
# Create a model with scikit-learn
model = Prophet()
```

✓ 0.2s Python

```python
# Fit the data into the model
model.fit(df_reorganized_CPI)
```

✓ 0.2s Python

```
16:57:44 - cmdstanpy - INFO - Chain [1] start processing
16:57:45 - cmdstanpy - INFO - Chain [1] done processing

<prophet.forecaster.Prophet at 0x234219068f0>
```

```python
# Assign the date column to X for test the r2 value
X = df_reorganized_CPI["ds"]
```

✓ 0.0s Python

```python
# Transform the X to dataframe for the model use
X = pd.DataFrame(df_reorganized_CPI["ds"])
```

✓ 0.0s Python

```python
# Get the CPI value from dataframe for testing the accuracy values
y = df_reorganized_CPI["y"]
```

✓ 0.0s Python

Assess the Prophet Model

```python
# Make predictions using the X set
train_X_forecast = model.predict(X)
```

✓ 0.2s Python

```python
# Get the predicted train CPI values from the forecast
predicted_train_y_values = train_X_forecast["yhat"]
```

✓ 0.0s Python

```python
# Compute metrics for the linear regression model: score, r2, mse, rmse, std
r2_LR = r2_score(y, predicted_train_y_values)
mse = mean_squared_error(y, predicted_train_y_values)
rmse = np.sqrt(mse)
std = np.std(y)

# Print relevant metrics.
print(f"The r2 is {r2_LR}.")
print(f"The mean squared error is {mse}.")
print(f"The root mean squared error is {rmse}.")
print(f"The standard deviation is {std}.")
```

✓ 0.0s Python

```
The r2 is 0.9983182452592712.
The mean squared error is 12.234888027871618.
The root mean squared error is 3.497840480621096.
The standard deviation is 85.29403680892736.
```

Predict the CPI for Future 80 Years

```
# Define X_future as the year values from 2024 to 2103
X_future = model.make_future_dataframe(periods=960, freq="M", include_history=False)
X_future
```

✓ 0.0s Python

	ds
0	2024-01-31
1	2024-02-29
2	2024-03-31
3	2024-04-30
4	2024-05-31
...	...
955	2103-08-31
956	2103-09-30
957	2103-10-31
958	2103-11-30
959	2103-12-31

960 rows × 1 columns

```
# Predict the CPI values for future 80 years using the model
predicted_future_forecast = model.predict(X_future)
```

✓ 0.1s Python

```
# Show the predicted future 80 years forecast
# ds: the date
# yhat: the predicted y value
# yhat_lower: the lower bound of the predicted y value
# yhat_upper: the upper bound of the predicted y value
predicted_future_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
```

✓ 0.0s Python

	ds	yhat	yhat_lower	yhat_upper
0	2024-01-31	284.799084	280.656883	289.211150
1	2024-02-29	285.586674	281.020454	289.839619
2	2024-03-31	286.146847	281.758390	290.761673
3	2024-04-30	286.640645	282.054231	291.171830
4	2024-05-31	287.142381	282.379346	291.499708
...	...	...	...	...
955	2103-08-31	675.368510	502.712390	835.853327
956	2103-09-30	675.847434	502.524307	836.860459
957	2103-10-31	676.329503	503.010000	836.682654
958	2103-11-30	676.935295	504.785530	837.619760
959	2103-12-31	677.209029	502.796024	839.700695

960 rows × 4 columns

```
# Get the predicted CPI values for future 80 years from the forecast
predicted_y = predicted_future_forecast["yhat"]
```

✓ 0.0s Python

Create Dataframe for Use

```python
# Create the dataframe from 1913 to 2023
df_to_2023 = df_reorganized_CPI[["ds", "y"]]
```

✓ 0.0s Python

```python
# Create the dataframe from 1913 to 2023 with the predicted value
df_to_2023_predict = df_to_2023.copy()
df_to_2023_predict["Predicted CPI"] = predicted_train_y_values
```

✓ 0.0s Python

```python
# Create the dataframe from 2024 to 2103
df_to_2103 = pd.DataFrame(X_future, columns=["ds"])
df_to_2103["y"] = predicted_y
```

✓ 0.0s Python

```python
df_to_2103_copy = df_to_2103.copy()
```

✓ 0.0s Python

```python
df_to_2103_copy['year'] = df_to_2103_copy["ds"].dt.year
df_to_2103_copy.head()
```

✓ 0.0s Python

	ds	y	year
0	2024-01-31	284.799084	2024
1	2024-02-29	285.586674	2024
2	2024-03-31	286.146847	2024
3	2024-04-30	286.640645	2024
4	2024-05-31	287.142381	2024

```python
df_to_2103_annual = df_to_2103_copy.groupby('year')['y'].mean().reset_index()
df_to_2103_annual.head()
```

✓ 0.0s Python

	year	y
0	2024	287.652767
1	2025	292.611108
2	2026	297.509082
3	2027	302.406971
4	2028	307.249524

```python
df_to_2103_annual.to_csv("./Resources/df_to_2103_annual.csv", index=False)
```

✓ 0.1s Python

```python
# Create a dataframe from 2024 to 2103 which CPI values are annual
future_years = np.array(range(2024, 2104))
```

✓ 0.0s Python

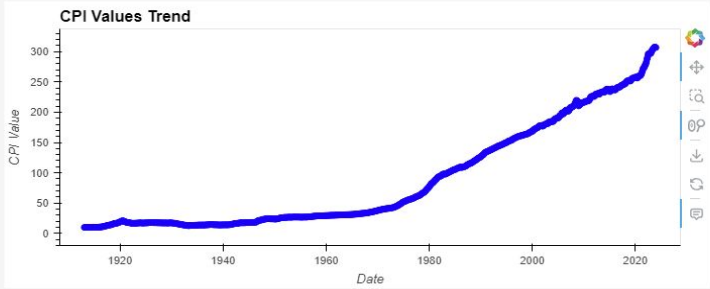
```python
# Create a dataframe that include the year and CPI value from 1913 to 2103
df_all_year = pd.concat([df_to_2023, df_to_2103])
```

✓ 0.0s Python

Visualize the Data

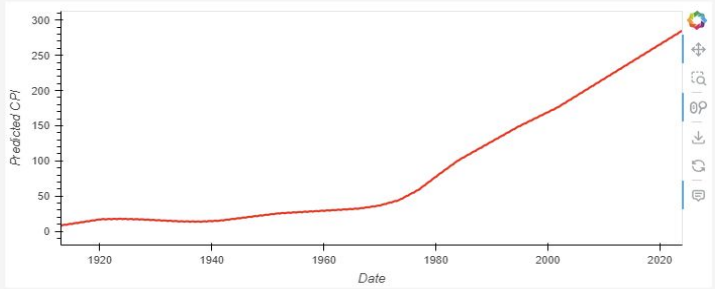
```python
# Create a scatter plot of year versus the CPI value
scatter_plot_for_year_vs_CPI = df_to_2023.hvplot.scatter(
    x="ds",
    xlabel="Date",
    y="y",
    ylabel="CPI Value",
    title="CPI Values Trend",
    color="blue"
)
scatter_plot_for_year_vs_CPI
```

0.1s Python




```python
# Create a line plot to show the model
line_plot_model = df_to_2023_predict.hvplot.line(
    x="ds",
    xlabel="Date",
    y="Predicted CPI",
    color="red"
)
line_plot_model
```

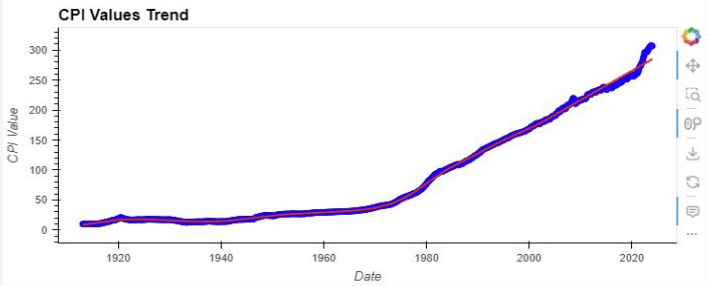
0.1s Python



Visualize the Data (Cont.)

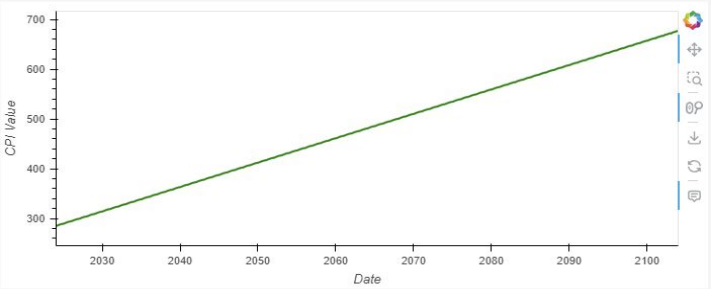
```
# See the model predict with the origin data
scatter_plot_for_year_vs_CPI * line_plot_model
```

✓ 0.1s Python




```
# Create the line plot for future 80 years
line_plot_future = df_to_2103.hvplot.line(
    x="ds",
    xlabel="Date",
    y="y",
    ylabel="CPI Value",
    color = "green"
)
line_plot_future
```

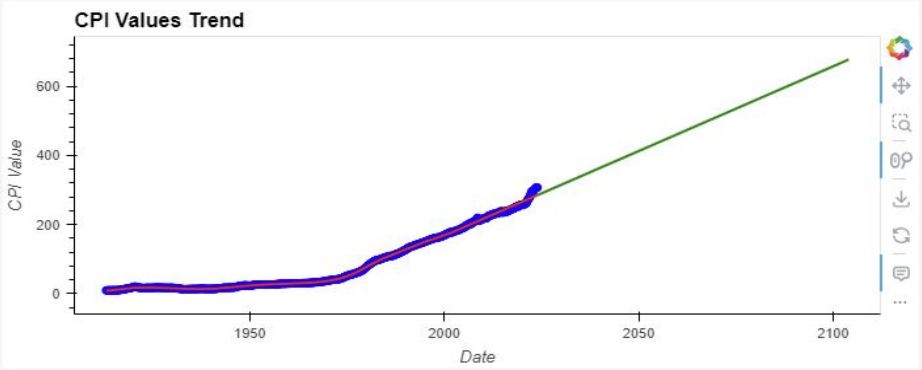
✓ 0.0s Python



Visualize the Data (Cont.)

```
# Combine the plots
scatter_plot_for_year_vs_CPI * line_plot_model * line_plot_future
```

✓ 0.1s Python



Cost of Living

```python
import pandas as pd

file = '/Users/shrutim/Downloads/data-FDgz9.csv'
```

Python

```python
cost_living_df = pd.read_csv(file)
cost_living_df.head()
```

Python

	State	Total Cost of Living¹	Total Cost Of Living Ranking	Total Disposable Income²	Disposable Income Ranking
0	Hawaii	$55,491	1	$5,929	50
1	Massachusetts	$53,860	2	$22,740	3
2	California	$53,171	3	$20,049	15
3	New York	$49,623	4	$25,247	1
4	New Jersey	$49,511	5	$21,379	10

```python
cost_living_df = cost_living_df.drop(['Total Cost Of Living Ranking',"Total Disposable Income<sup>2</sup>",
cost_living_df = cost_living_df.rename(columns={'Total Cost of Living<sup>1</sup>': 'cost_of_living'})
cost_living_df.head()
```

Python

	State	cost_of_living
0	Hawaii	$55,491
1	Massachusetts	$53,860
2	California	$53,171
3	New York	$49,623
4	New Jersey	$49,511

```python
for state in range(len(cost_living_df['cost_of_living'])):
    cost_living_df['cost_of_living'][state] = cost_living_df['cost_of_living'][state].replace('$', '')
    cost_living_df['cost_of_living'][state] = cost_living_df['cost_of_living'][state].replace(',', '')
    cost_living_df['cost_of_living'][state] = int(cost_living_df['cost_of_living'][state])

cost_living_df.head()
```

Python

	State	cost_of_living
0	Hawaii	55491
1	Massachusetts	53860
2	California	53171
3	New York	49623
4	New Jersey	49511

```python
cost_living_df.to_csv('/Users/shrutim/Downloads/cost_living_df.csv', index=False)
```

Python

Life Expectancy

```python
import pandas as pd
male_file = '/Users/shrutim/Downloads/PerLifeTables_M_Alt2_TR2017.csv'
female_file = '/Users/shrutim/Downloads/PerLifeTables_F_Alt2_TR2017.csv'
```

Python

```python
male_df = pd.read_csv(male_file, header = 4)
male_df.head()

female_df = pd.read_csv(female_file, header = 4)
female_df.head()
```

Python

	Year	x	q(x)	l(x)	d(x)	L(x)	T(x)	e(x)	D(x)	M(x)	A(x)	N(x)	a(x)	12a(x)
0	2015	0	0.005145	100000	515	99547	8136028	81.36	100000	12622	0.1262	3323611	33.2361	393.33
1	2015	1	0.000334	99485	33	99469	8036481	80.78	96870	12121	0.1251	3223611	33.2777	393.83
2	2015	2	0.000218	99452	22	99441	7937012	79.81	94292	12089	0.1282	3126741	33.1603	392.42
3	2015	3	0.000163	99431	16	99423	7837571	78.82	91793	12069	0.1315	3032450	33.0358	390.93
4	2015	4	0.000122	99414	12	99408	7738148	77.84	89365	12055	0.1349	2940657	32.9061	389.37

```python
print(male_df.columns)
```

Python

```
Index(['Year', 'x', 'q(x)', 'l(x)', 'd(x)', 'L(x)', 'T(x)', 'e(x)', 'D(x)',
       'M(x)', 'A(x)', 'N(x)', 'a(x)', '12a(x)'],
      dtype='object')
```

```python
male_df = male_df.drop(['q(x)','l(x)', 'd(x)', 'L(x)', 'T(x)', 'D(x)', 'M(x)', 'A(x)', 'N(x)','a(x)','12a
male_df.head()

female_df = female_df.drop(['q(x)','l(x)', 'd(x)', 'L(x)', 'T(x)', 'D(x)', 'M(x)', 'A(x)', 'N(x)','a(x)',
female_df.head()
```

Python

	Year	x	e(x)
0	2015	0	81.36
1	2015	1	80.78
2	2015	2	79.81
3	2015	3	78.82
4	2015	4	77.84

```python
male_df = male_df.rename(columns={'e(x)' : 'male_life_expectancy'})
female_df = female_df.rename(columns={'e(x)' : 'female_life_expectancy'})
```

Python

```python
male_df = male_df[male_df['Year'] == 2024]
male_df = male_df.reset_index(drop= True)
male_df.tail()
```

Python

	Year	x	male_life_expectancy
115	2024	115	0.95
116	2024	116	0.89
117	2024	117	0.82
118	2024	118	0.76
119	2024	119	0.71

Life Expectancy (Cont.)

```python
female_df = female_df[female_df['Year'] == 2024]
female_df = female_df.reset_index(drop= True)
female_df.tail()
```

Python

	Year	x	female_life_expectancy
115	2024	115	0.95
116	2024	116	0.89
117	2024	117	0.82
118	2024	118	0.76
119	2024	119	0.71

```python
life_expentancy_df = pd.merge(female_df, male_df, on="x")
life_expentancy_df = life_expentancy_df.drop('Year_y', axis= 1)
life_expentancy_df = life_expentancy_df.rename(columns={'Year_x' : 'year', 'x': 'age'})
life_expentancy_df.head()
```

Python

	year	age	female_life_expectancy	male_life_expectancy
0	2024	0	82.30	77.91
1	2024	1	81.66	77.31
2	2024	2	80.68	76.34
3	2024	3	79.70	75.35
4	2024	4	78.71	74.37

```python
life_expentancy_df.to_csv('/Users/shrutim/Downloads/life_expectancy_df.csv', index=False)
```

Python

Retirement Calculator

```python
#importing dependancies
from datetime import datetime as dt
import pandas as pd

#importing data
inflation_file = 'Resources/df_to_2103_annual.csv'
col_file = 'Resources/cost_living_df.csv'
life_file = 'Resources/life_expectancy_df.csv'
```

✓ 2.8s Python

```python
#creating dataframes for future inflation, cost of living, and life expectancy data
future_inflation_df = pd.read_csv(inflation_file)
cost_living_df = pd.read_csv(col_file)
life_expectancy_df = pd.read_csv(life_file)
```

✓ 0.0s Python

```python
#getting input data for age, gender, saving, and state
age = int(input("How old are you?"))
gender_input = input("What is your biological sex?")
savings = int(input("How many money in USD dollars do you have in savings for retirement right now?"))
state_input = input("What state do you live in?")
```

✓ 2m 9.1s Python

```python
#setting default gender to male
gender = 'male'

#checking for f in input to detect if user meant female
#accounting for spelling errors
for i in range(len(gender_input)):
        if gender_input[i] == 'f':
            gender = 'female'

#according to gender, calling the right life expentancy data
life_expectancy_column = gender + '_life_expectancy'

#grabbing the value of the number of years left in life based on age
life_expectancy_age_gender = life_expectancy_df[life_expectancy_column][age]

#getting the index of the state
#does not count for spelling errors
for state in range(len(cost_living_df['State'])):
    if state_input == cost_living_df['State'][state]:
        state_index = state

#pulling the average cost of living based on the state
cost_of_living = cost_living_df['cost_of_living'][state_index]


#print('age:',age,', gender_input:', gender_input,', gender:', gender,', life_expectancy: ',life_expectancy_
```

✓ 0.0s Python

Retirement Calculator (Cont.)

```python
#sets current year to real time
current = dt.now().year

#rounding years left and adding an extra year for emergency
year_left = round(life_expectancy_age_gender) + 1

#finding year of death
death_year = year_left + current

#initializing total cost
total_cost = 0

#grabbing predicted inflation CPI value for current year
current_inflation = future_inflation_df["y"][current - 2024]

for year in range(current, death_year):
    index_value = year - current
    #getting inflation CPI for each year in future
    inflation = future_inflation_df["y"][index_value]

    #calculating the total cost of each year in current money value by accounting for inflation
    yearly_cost = cost_of_living * inflation/(current_inflation)
    #adding yearly cost to total cost
    total_cost = total_cost + yearly_cost

#removing savings from total cost to find the remainder of money saved
money_needed = round(total_cost - savings, 2)

#printing remaining money needed to retire right now

if money_needed <= 0:
    print("You can retire right now!!! YAY!!!!")
else:
    print("Remaining money needed to retire right now: $",money_needed)
```

✓ 0.0s Python

Demo

Q&A

Thank you
The end.