HWK5 Part 1: Baby Boomers Through Time

Name: Dyab Asdi

UT EID: DA32435

Name:

UT EID:

Name:

UT EID:

Date:

Creating Interactive Charts to Visualize Population Shifts over Time with Altair

Baby boomers (often shortened to boomers) are the demographic cohort following the Silent Generation and preceding Generation X. The generation is generally defined as people born from 1946 to 1964, during the post–World War II baby boom. The term is also used outside the United States but the dates, the demographic context and the cultural identifiers may vary. The baby boom has been described variously as a "shockwave"and as "the pig in the python." Baby boomers are often parents of late Gen Xers and Millennials. from wikipedia.

Let us explore this "shockwave" by examining the US Census data available via the vega datasets package. We'll start by doing some data engineering to add a column in our population data to denote generational membership, then we will juxtapose the sex distribution of the population using a brush and linking technique we studied in the lab. Finally we will add a slider to animate the transition through time.

In [13]:
```
# Import the necessary libraries and data
import altair as alt
import pandas as pd

df_pop = pd.read_json('population.json')
```

In [14]:
```
df_pop.head()
```

Out[14]:

	year	age	sex	people
0	1850	0	1	1483789
1	1850	0	2	1450376
2	1850	5	1	1411067
3	1850	5	2	1359668
4	1850	10	1	1260099

In [15]:
```
df_pop.shape
```

Out[15]: (570, 4)

In [16]:
```
df_pop.describe()
```

Out[16]:

	year	age	sex	people
count	570.000000	570.000000	570.000000	5.700000e+02
mean	1927.333333	45.000000	1.500000	3.428937e+06
std	46.726717	27.410182	0.500439	3.101098e+06
min	1850.000000	0.000000	1.000000	5.259000e+03
25%	1880.000000	20.000000	1.000000	6.742840e+05
50%	1930.000000	45.000000	1.500000	2.548015e+06
75%	1970.000000	70.000000	2.000000	5.466410e+06
max	2000.000000	90.000000	2.000000	1.163565e+07

In [17]:
```
df_pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 570 entries, 0 to 569
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   year    570 non-null    int64
 1   age     570 non-null    int64
 2   sex     570 non-null    int64
 3   people  570 non-null    int64
dtypes: int64(4)
memory usage: 17.9 KB
```

Q1 (10 points) - Add in the "Boomer" label

As we can see from inspecting the dataframe, our data only gives us information for:

- year
- age
- sex
- people

But, we want to be able to highlight just the people born between 1946 - 1964 as a separate group. To accomplish this, we want to create a new categorical attribute - `Generation`

Using pandas data manipulation techniques, add a new column to `df_pop` named `Generation` that either has the value `Baby Boomer` or `Other`.

In [18]:
```
# your code here
df_pop['Generation'] = df_pop.apply(
    lambda row: 'Baby Boomer' if (row['year'] - row['age'] >= 1946 and row['year'] - row[
    axis=1
```

```
)
df_pop.head()
```

Out[18]:

	year	age	sex	people	Generation
0	1850	0	1	1483789	Other
1	1850	0	2	1450376	Other
2	1850	5	1	1411067	Other
3	1850	5	2	1359668	Other
4	1850	10	1	1260099	Other

Q2 (10 points) - Change the encoding for `sex`

As in our lab in class, the sex "Male" is encoded as the number `1` and "Female" is encoded as `2`. Modify the dataframe `df_pop` to replace the encoding with the string so when we create our plots this will automatically have the legend come out correctly (note, you can map numbers to labels in Altair as well).

In [19]:
```
# your code here
df_pop['sex'] = df_pop['sex'].replace({1: 'Male', 2: 'Female'})
```

In [20]:
```
df_pop.head()
```

Out[20]:

	year	age	sex	people	Generation
0	1850	0	Male	1483789	Other
1	1850	0	Female	1450376	Other
2	1850	5	Male	1411067	Other
3	1850	5	Female	1359668	Other
4	1850	10	Male	1260099	Other

Q3 (10 points) - Show the Population Change Over Time with a Slider

Now, we have a snapshot of 2 different years next to each other, but what about creating a crude animation by controlling the the year displayed with a slider?

Create a slider using this example to help guide you. Our plot will look similar, except we have not split our bar chart up by `sex` yet. Name the slider 'Select Year:' (this in controlled in `binding_range`, and not in the `selection_single` parameters).

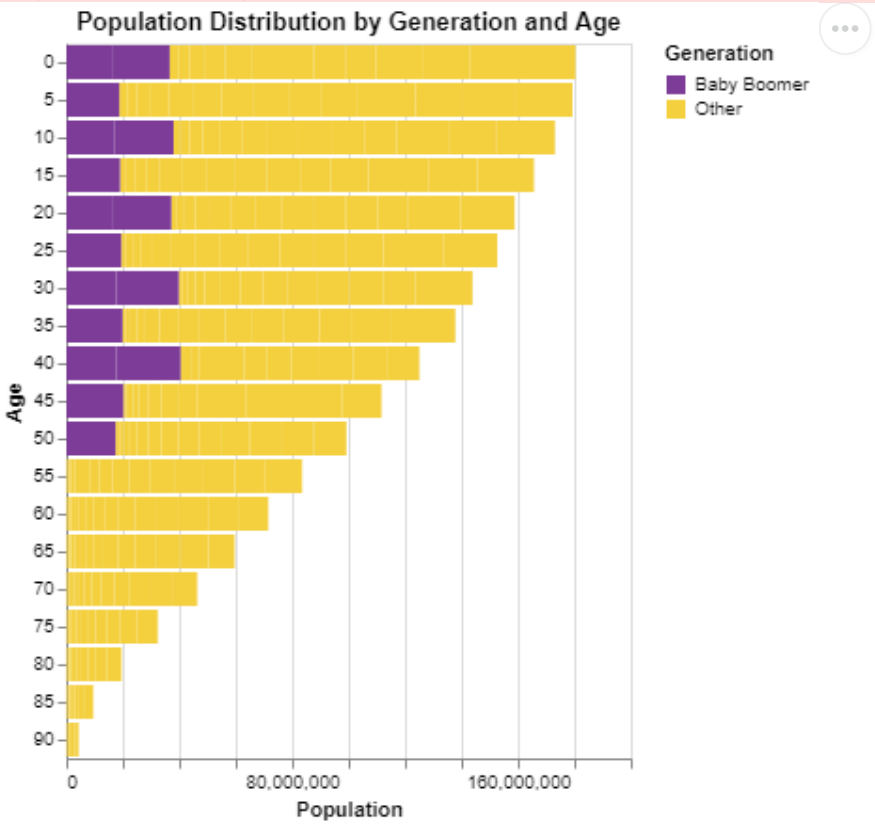
Encode membership of the Baby Boomer generation with color using "#7D3C98" (purple) for the boomers "#F4D03F" (gold) for other.

Start the slider at 1900.

In [21]:
```
# your code here
slider = alt.binding_range(min=1900, max=2025, step=1, name='Select Year:')
selection = alt.selection_single(fields=['year'], bind=slider)
```

```
chart = alt.Chart(df_pop).mark_bar().encode(
    x=alt.X('sum(people):Q', title='Population'),
    y=alt.Y('age:O', title='Age'),
    color=alt.Color('Generation:N', scale=alt.Scale(domain=['Baby Boomer', 'Other'], rang
    tooltip=['year', 'Generation', 'sum(people)']
).add_selection(
    selection
).transform_filter(
    selection
).properties(
    title='Population Distribution by Generation and Age'
)

chart.show()
```

```
C:\Users\dyaba\AppData\Local\Temp\ipykernel_9412\3313691713.py:3: AltairDeprecationWarnin
g:
Deprecated since `altair=5.0.0`. Use selection_point instead.
  selection = alt.selection_single(fields=['year'], bind=slider)
C:\Users\dyaba\AppData\Local\Temp\ipykernel_9412\3313691713.py:9: AltairDeprecationWarnin
g:
Deprecated since `altair=5.0.0`. Use add_params instead.
  ).add_selection(
```



Select Year:

Q4 (15 points) - Linking

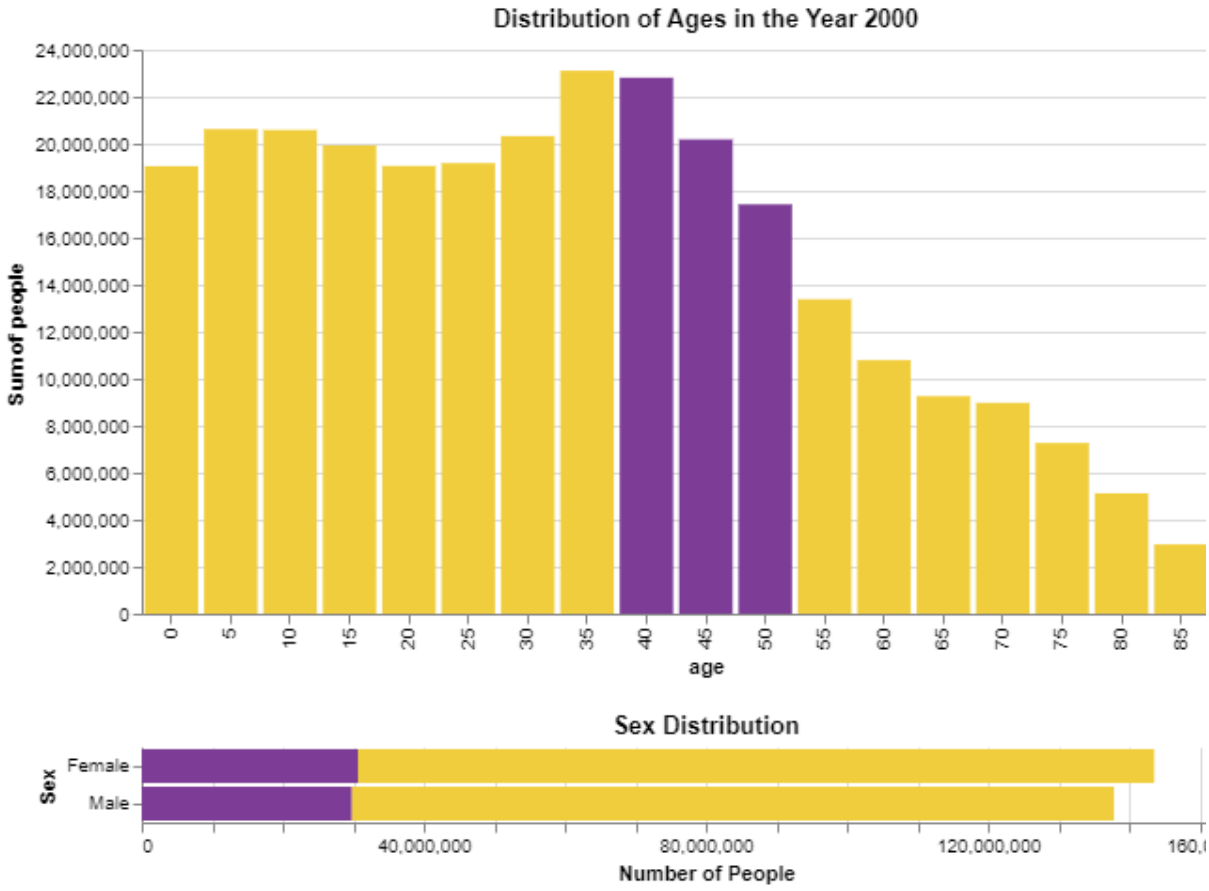
Let us take a closer look at just the year 2000 data, and find what the distribution of sex is for each individual age grouping. Plot the distribution of ages as a bar chart for just the year 2000, and link a histogram that will plot the distribution of sex for the current selection. It should default to no age group selected (as in, all bars are colored, meaning none are selected). The histogram for the sex distribution should appear below the year 2000 data (vertically concatenated). When a bar on the

top chart is selected, indicate its selection by turning the other bars light gray. The histogram of the sex distribution below it should be a horizontal bar chart.

Encode membership of the Baby Boomer generation with color using "#7D3C98" (purple) for the boomers, and "#F4D03F" (gold) for other.

In [27]:
```
# your code here
df_2000 = df_pop[df_pop['year'] == 2000].copy()

df_2000['Generation'] = df_2000['age'].apply(
    lambda x: 'Baby Boomer' if 1946 <= (2000 - x) <= 1964 else 'Other'
)

generation_color = alt.Scale(
    domain=['Baby Boomer', 'Other'],
    range=['#7D3C98', '#F4D03F']
)

age_selection = alt.selection_point(fields=['age'])

age_chart = alt.Chart(df_2000).mark_bar().encode(
    x='age:O',
    y='sum(people):Q',
    color=alt.condition(
        age_selection,
        alt.Color('Generation:N')
            .scale(domain=['Baby Boomer', 'Other'], range=['#7D3C98', '#F4D03F'])
            .legend(title="Generation"),
        alt.value('lightgray')
    ),
    tooltip=['age', 'sum(people)', 'Generation']
).properties(
    title='Distribution of Ages in the Year 2000',
    width=600
).add_params(
    age_selection
)

sex_chart = alt.Chart(df_2000).mark_bar().encode(
    y=alt.Y('sex:N', title='Sex'),
    x=alt.X('sum(people):Q', title='Number of People'),
    color=alt.Color('Generation:N')
        .scale(domain=['Baby Boomer', 'Other'], range=['#7D3C98', '#F4D03F'])
        .legend(title="Generation"),
    tooltip=['sex', 'sum(people)', 'Generation']
).transform_filter(
    age_selection
).properties(
    title='Sex Distribution',
    width=600
)

final_chart = alt.vconcat(age_chart, sex_chart).configure_view(stroke=None)

final_chart
```

Out[27]:



Q5 (15 points) - Combine Q3 and Q4 to One Chart

In Q4, we linked the distribution of sex to the age selection for just the year 2000. Let us visualize all the data by incorporating the year selection slider from Q3 so that you can select which year of data you are viewing. Retain the ability to just select one age group for the sex distribution, and default to no age group selected.

Add a tooltip so you can see exactly how many people are in the age range for the top "Distribution of Ages for the Selected Year" histogram.

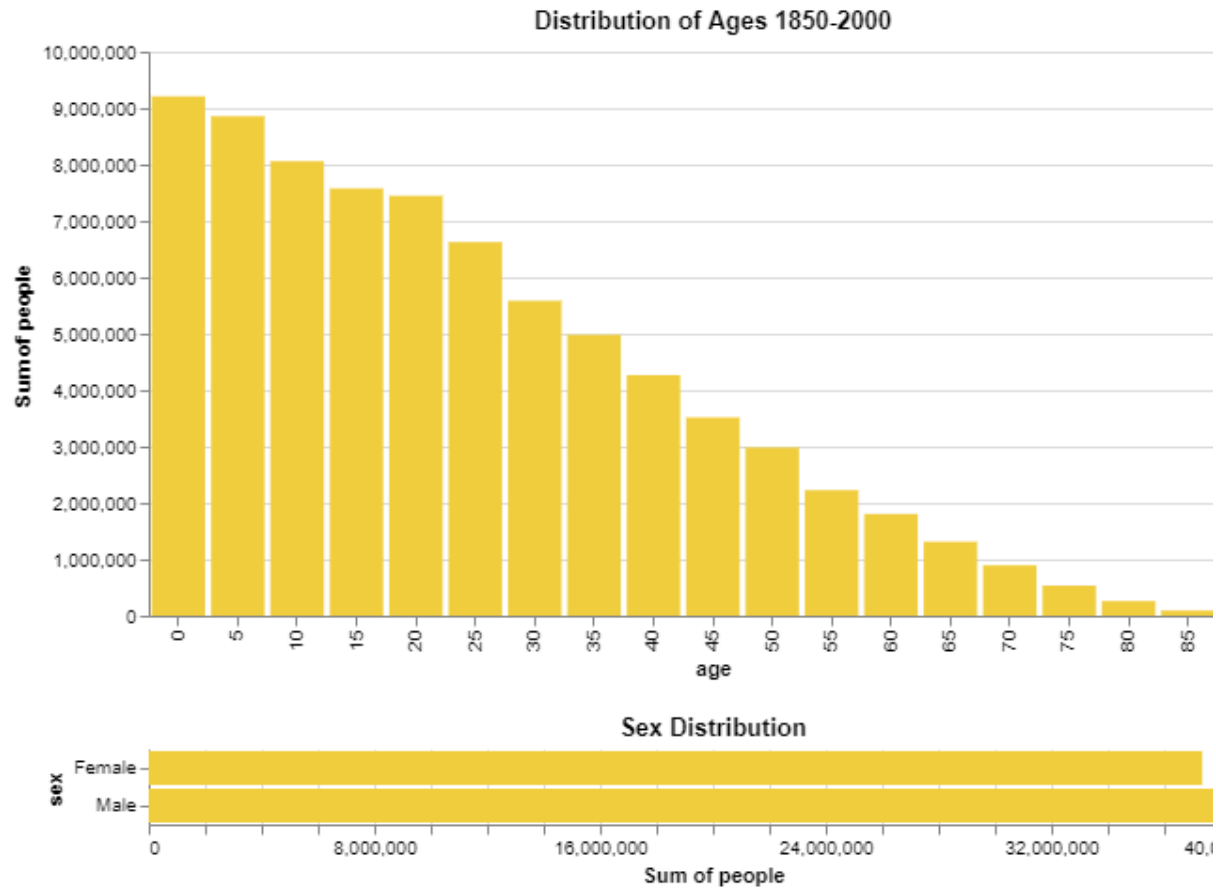
Encode membership of the Baby Boomer generation with color using "#7D3C98" (purple) for the boomers and "#F4D03F" (gold) for other.

In [29]:

```
# your code here
year_selection = alt.selection_point(
    fields=['year'],
    bind=alt.binding_range(min=1850, max=2000, step=10, name='Select Year:'),
    value=1900
)
age_chart = alt.Chart(df_pop).mark_bar().encode(
    x='age:O',
    y='sum(people):Q',
    color=alt.condition(
        age_selection,
        alt.Color('Generation:N')
            .scale(domain=['Baby Boomer', 'Other'], range=['#7D3C98', '#F4D03F'])
            .legend(title="Generation"),
```

```
        alt.value('lightgray')
    ),
    tooltip=['age', 'sum(people)', 'Generation']
).properties(
    title='Distribution of Ages 1850-2000',
    width=600
).add_params(
    age_selection,
    year_selection
).transform_filter(
    year_selection
)

sex_chart = alt.Chart(df_pop).mark_bar().encode(
    y='sex:N',
    x='sum(people):Q',
    color=alt.Color('Generation:N')
        .scale(domain=['Baby Boomer', 'Other'], range=['#7D3C98', '#F4D03F'])
        .legend(title="Generation"),
    tooltip=['sex', 'sum(people)', 'Generation']
).transform_filter(
    age_selection,
    year_selection
).properties(
    title='Sex Distribution',
    width=600
)

final_chart = alt.vconcat(age_chart, sex_chart).configure_view(stroke=None)

final_chart
```

Out[29]:



Select Year: 1900