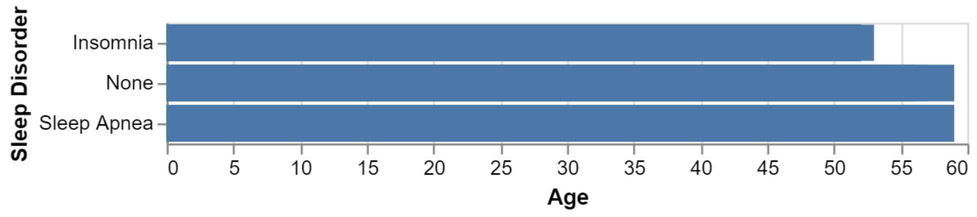
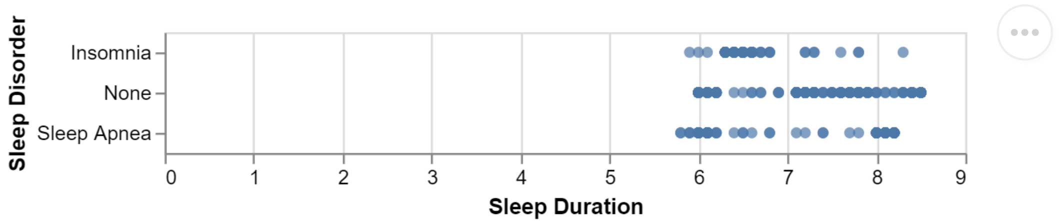
```
# Import our data processing library (note: you may have to install this!)
import pandas as pd

# Let's use this to upload a sample dataset and show the start of the dataset
# Note that you need to download the dataset and make sure it's in the same
# directory as your notebook
data= pd.read_csv("/content/Sleep_health_and_lifestyle_dataset.csv")
data.head()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level
0	1	Male	27	Software Engineer	6.1	6	42	6
1	2	Male	28	Doctor	6.2	6	60	8
2	3	Male	28	Doctor	6.2	6	60	8
3	4	Male	28	Sales Representative	5.9	4	30	8
4	5	Male	28	Sales Representative	5.9	4	30	8

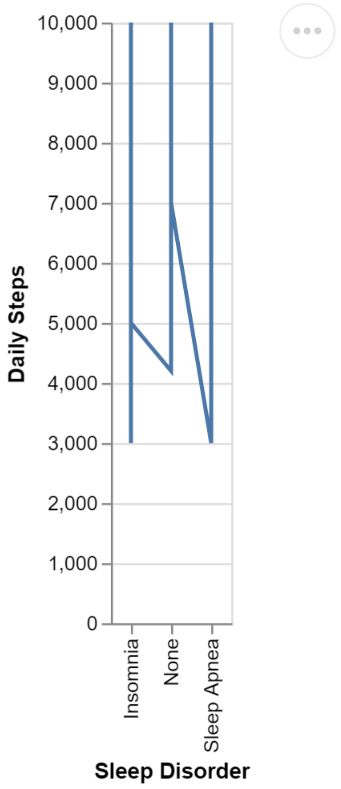
```
# Now let's visualize the data
import altair as alt

alt.Chart(data).mark_bar().encode(x="Age", y="Sleep Disorder")
```



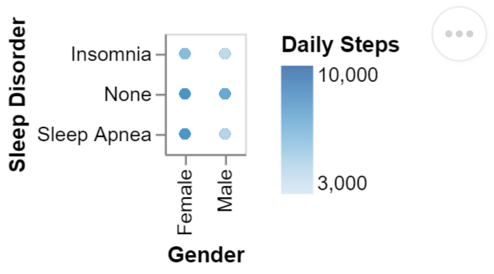
```
alt.Chart(data).mark_circle().encode(x="Sleep Duration", y="Sleep Disorder")
```



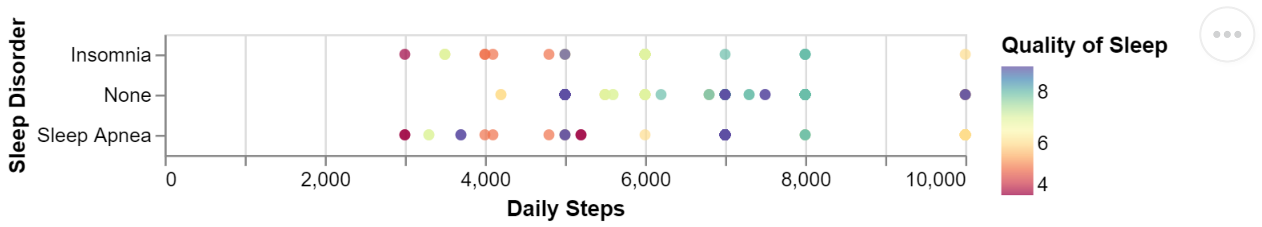
```
alt.Chart(data).mark_line().encode(
    x='Sleep Disorder',
    y='Daily Steps'
)
```



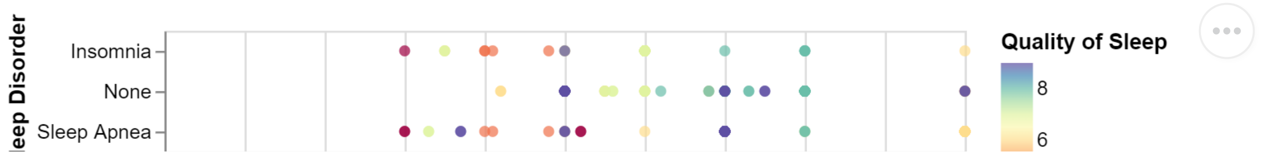
```
alt.Chart(data).mark_circle().encode(
    x = "Gender",
    y = "Sleep Disorder",
    color="Daily Steps"
)
```



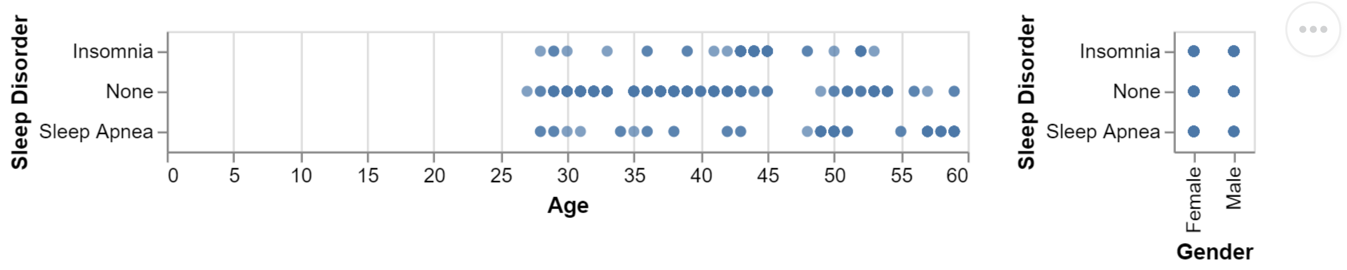
```
alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('Quality of Sleep', scale=alt.Scale(scheme='spectral'))
)
```



```
alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('Quality of Sleep', scale=alt.Scale(scheme='spectral')),
    tooltip=["Gender", "Sleep Disorder"]
)
```



```
c1 = alt.Chart(data).mark_circle().encode(
    x = "Age",
    y = "Sleep Disorder",
)

c2 = alt.Chart(data).mark_circle().encode(
    x = "Gender",
    y = "Sleep Disorder",
)

c1|c2
```

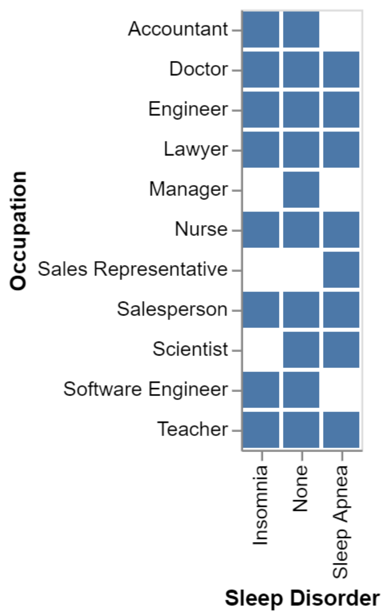


```
# Build a SPLOM
alt.Chart(data).mark_circle().encode(
    alt.X(alt.repeat("column"), type="quantitative"),
    alt.Y(alt.repeat("row"), type="quantitative"),
    color="Sleep Disorder",
    tooltip=["Age", "Sleep Disorder"]
).properties(
    width=125,
    height=125
).repeat(
    row=["Sleep Duration","Quality of Sleep","Physical Activity Level","Daily Steps"],
    column=["Sleep Duration","Quality of Sleep","Physical Activity Level","Daily Steps"]
)
```

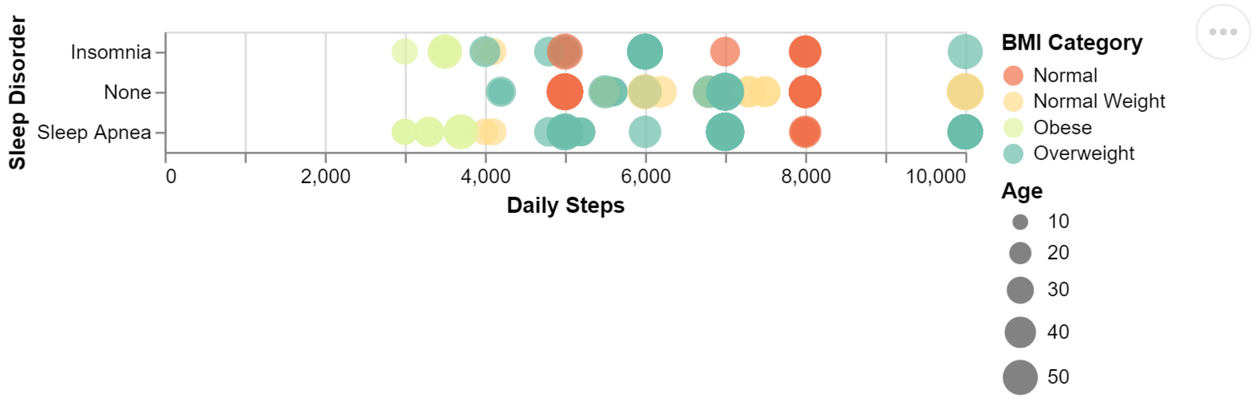
```
# Build a parallel coordinates plot
alt.Chart(data).transform_window(
    index="count()"
).transform_fold(
    ["Sleep Duration","Quality of Sleep","Physical Activity Level","Stress Level"]
).mark_line().encode(
    x="key:N",
    y="value:Q",
    detail="index:N",
    opacity=alt.value(0.5),
    color=alt.Color("Heart Rate", scale=alt.Scale(scheme="Magma")),
    tooltip=["Sleep Disorder"]
).properties(width=700).interactive()
```

```
# Store the SPLOM
chart = alt.Chart(data).mark_circle().encode(
    alt.X(alt.repeat("column"), type="quantitative"),
    alt.Y(alt.repeat("row"), type="quantitative"),
    color="Sleep Disorder",
    tooltip=["Occupation", "Sleep Disorder"]
).properties(
    width=125,
    height=125
).repeat(
    row=["Sleep Duration","Quality of Sleep","Physical Activity Level","Stress Level"],
    column=["Sleep Duration","Quality of Sleep","Physical Activity Level","Stress Level"]
).interactive()

chart.save('/content/webchart.html', embed_options={'renderer':'svg'})
```

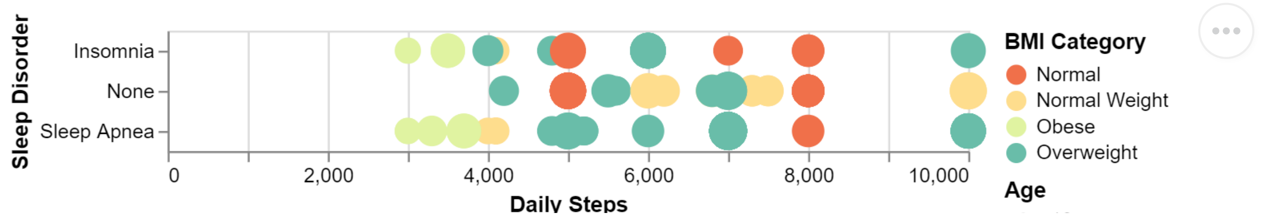
```
alt.Chart(data).mark_bar().encode(x="Sleep Disorder", y="Occupation")
```



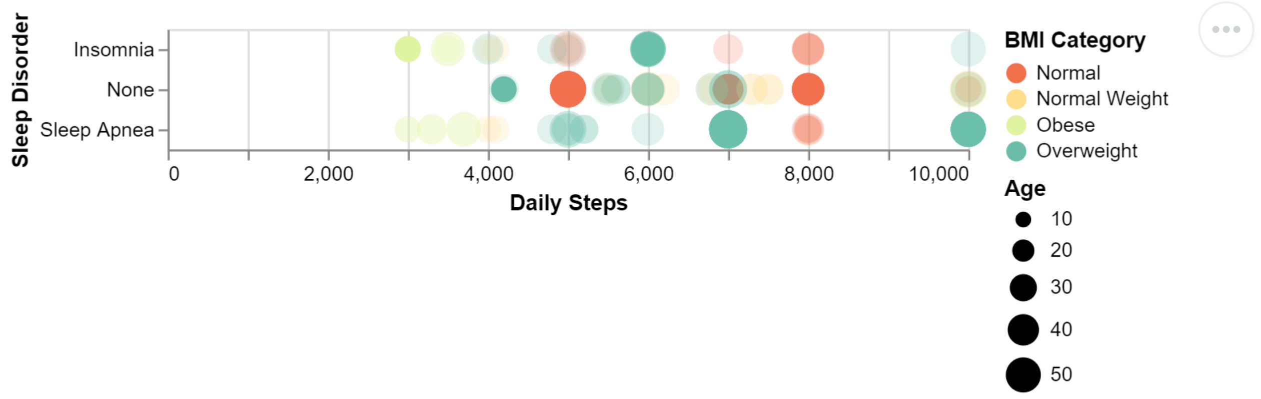
```
alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation","Sleep Disorder"]
)
```



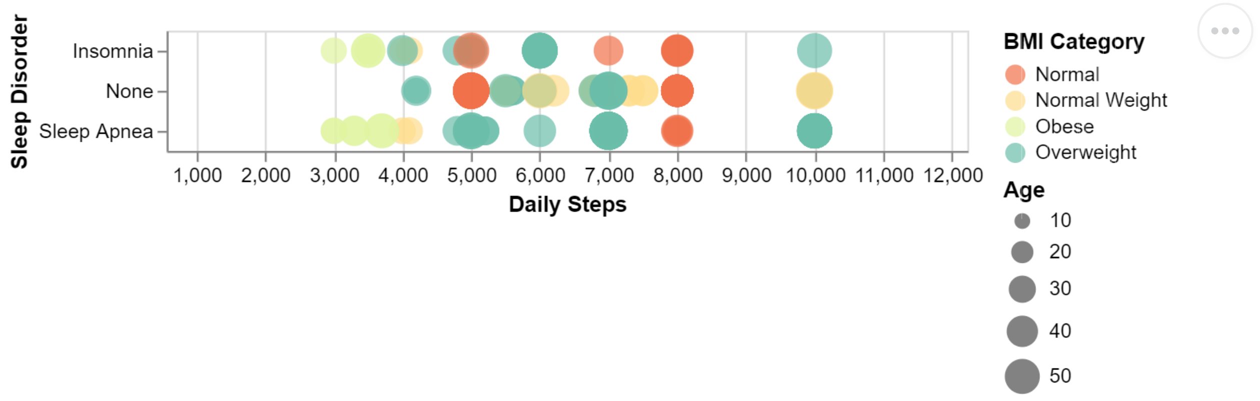
```
# Implementing selection
selection = alt.selection(type='multi', fields=['Occupation'])

alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation", "Sleep Disorder"],
    opacity=alt.condition(selection,alt.value(1),alt.value(.2))
).add_selection(selection)
```



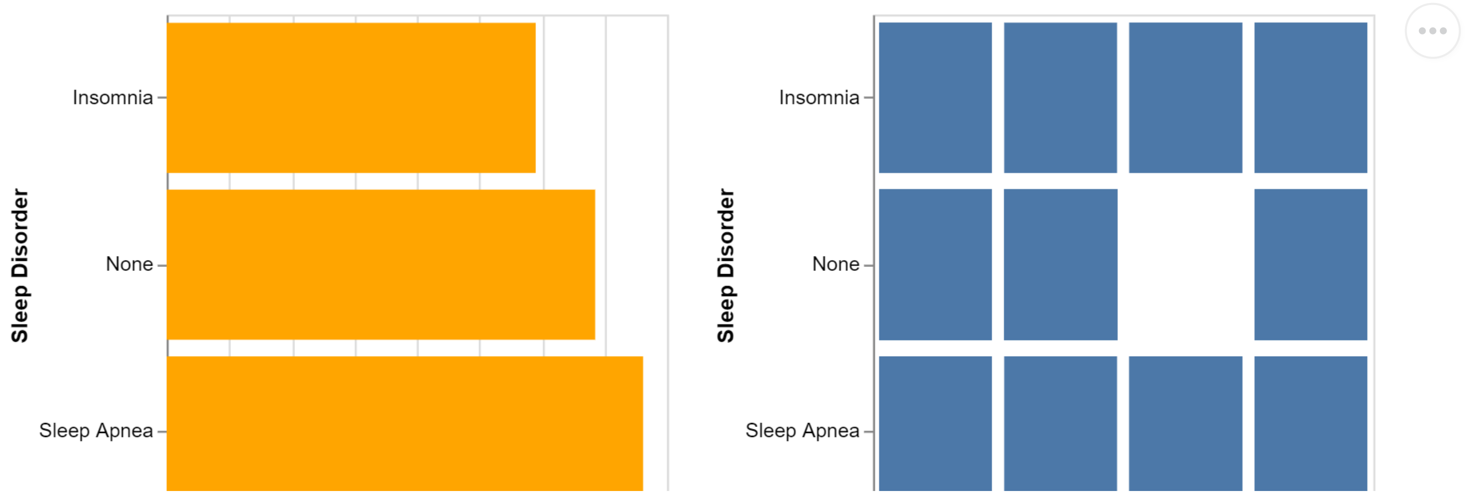
```
selection = alt.selection(type='multi', fields=['Occupation'], on='mouseover', nearest=True)

alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation", "Sleep Disorder"],
    opacity=alt.condition(selection,alt.value(1),alt.value(.2))
).add_selection(selection)
```



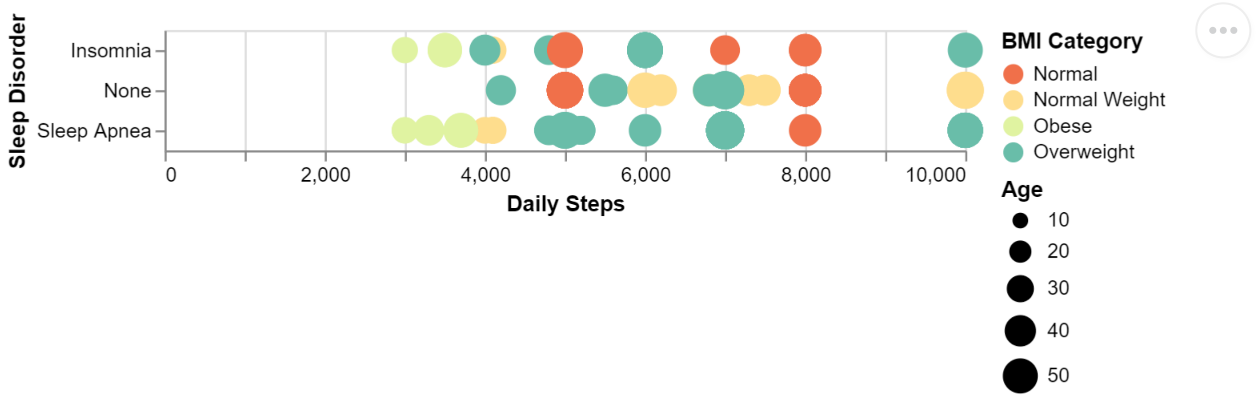
```
alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation", "Sleep Disorder"]
).interactive()
```



```python
# Let's implement filtering using dynamic queries.
selection = alt.selection(type="multi", fields=["Region"])

# Create a container for our two different views
base =  alt.Chart(data).properties(width=500, height=250)

# Let's specify our overview chart
overview = alt.Chart(data).mark_bar().encode(
    y = "Sleep Disorder",
    x = "mean(Daily Steps)",
    color=alt.condition(selection, alt.value("orange"), alt.value("lightgrey"))
).add_selection(selection).properties(height=250, width=250)

# Create a detail chart
detail = hist = base.mark_bar().encode(
    y = "Sleep Disorder",
    x = "BMI Category"
).transform_filter(selection).properties(height=250, width=250)

overview | detail
```

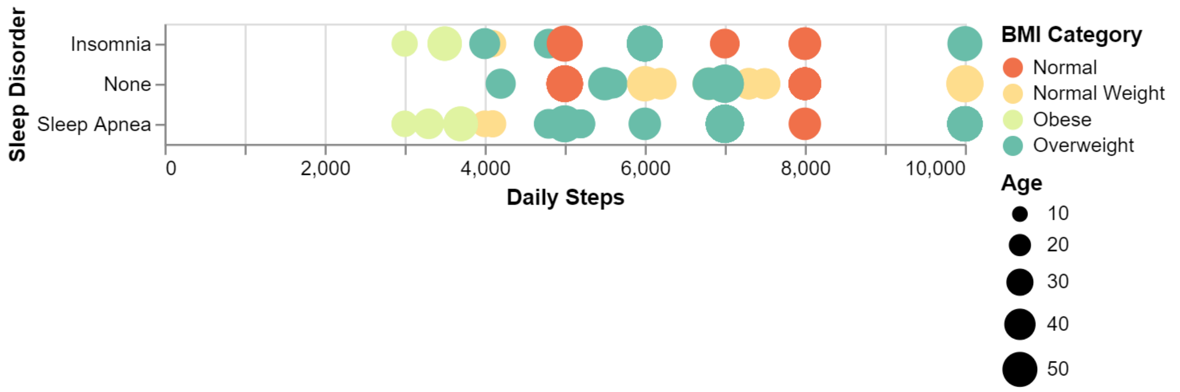


```
selection = alt.selection(type='multi', fields=['BMI Category'], bind='legend')
alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation", "Sleep Disorder"],
    opacity=alt.condition(selection,alt.value(1),alt.value(.2))
).add_selection(selection)
```

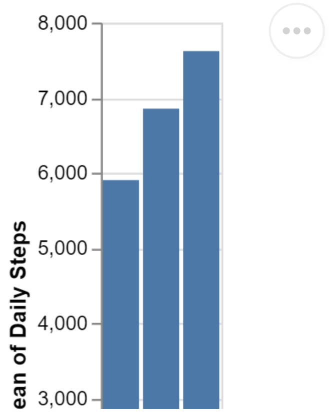


```
# dropdown = alt.binding_select (options=data["Occupation"].unique(),name="Select a BMI Category:")
```

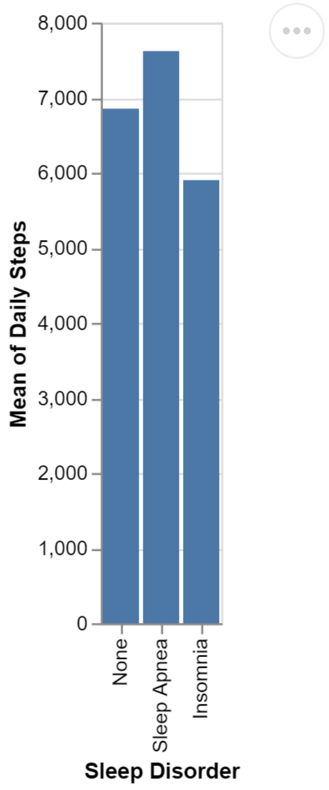
```
selection = alt.selection(type='multi', fields=['BMI Category'], bind='legend')

alt.Chart(data).mark_circle().encode(
    x = "Daily Steps",
    y = "Sleep Disorder",
    color=alt.Color('BMI Category', scale=alt.Scale(scheme='spectral')),
    size="Age",
    tooltip=["Occupation", "Sleep Disorder"],
    opacity=alt.condition(selection,alt.value(1),alt.value(.2))
).add_selection(selection)
```



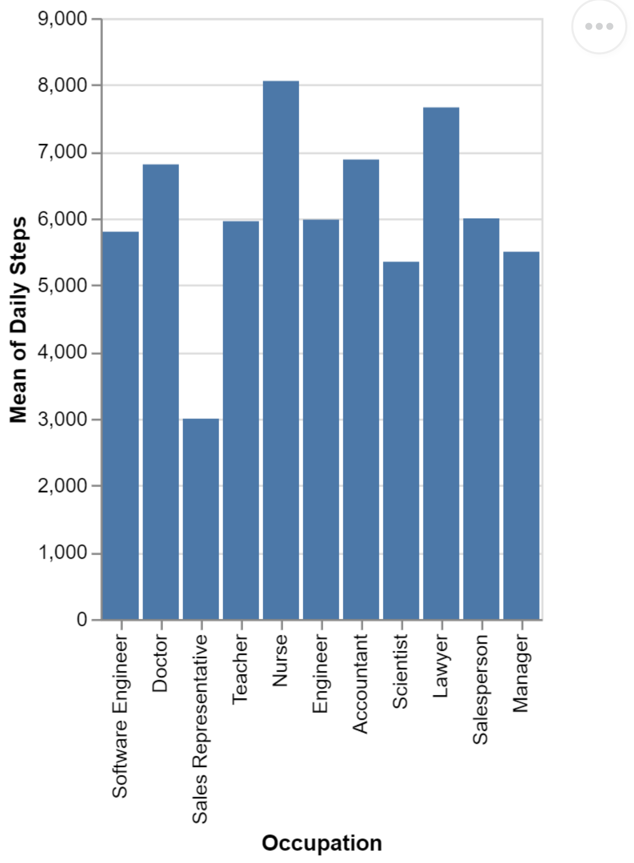
```
alt.Chart(data).mark_bar().encode(
    y = "mean(Daily Steps)",
    x = "Sleep Disorder"
)
```



```
alt.Chart(data).mark_bar().encode(
    y = "mean(Daily Steps)",
    x = alt.X(field='Sleep Disorder', type='nominal', sort=alt.EncodingSortField(field='BMI Category', op='mean'))
)
```

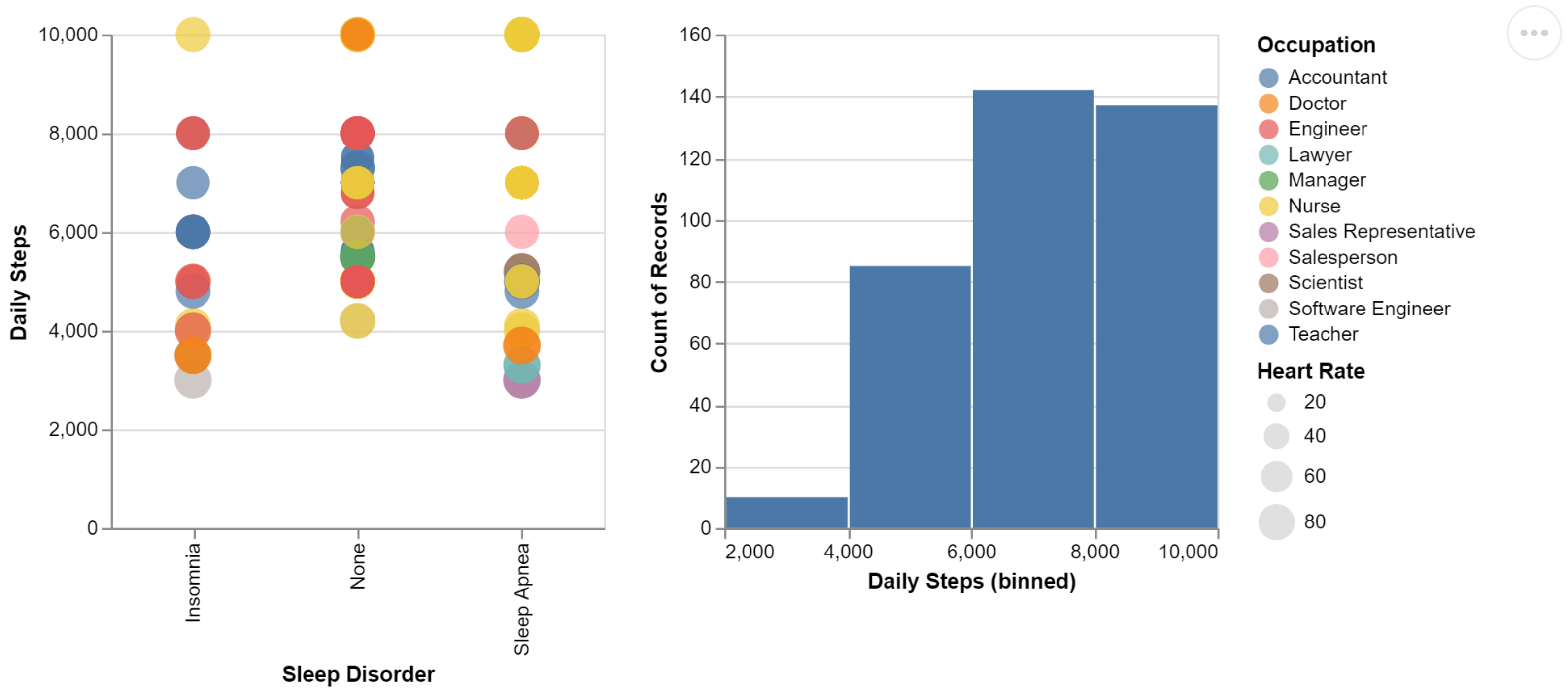


```
alt.Chart(data).mark_bar().encode(
    y = "mean(Daily Steps)",
    x = alt.X(field='Occupation', type='nominal', sort=alt.EncodingSortField(field='BMI Category', op='mean'))
)
```



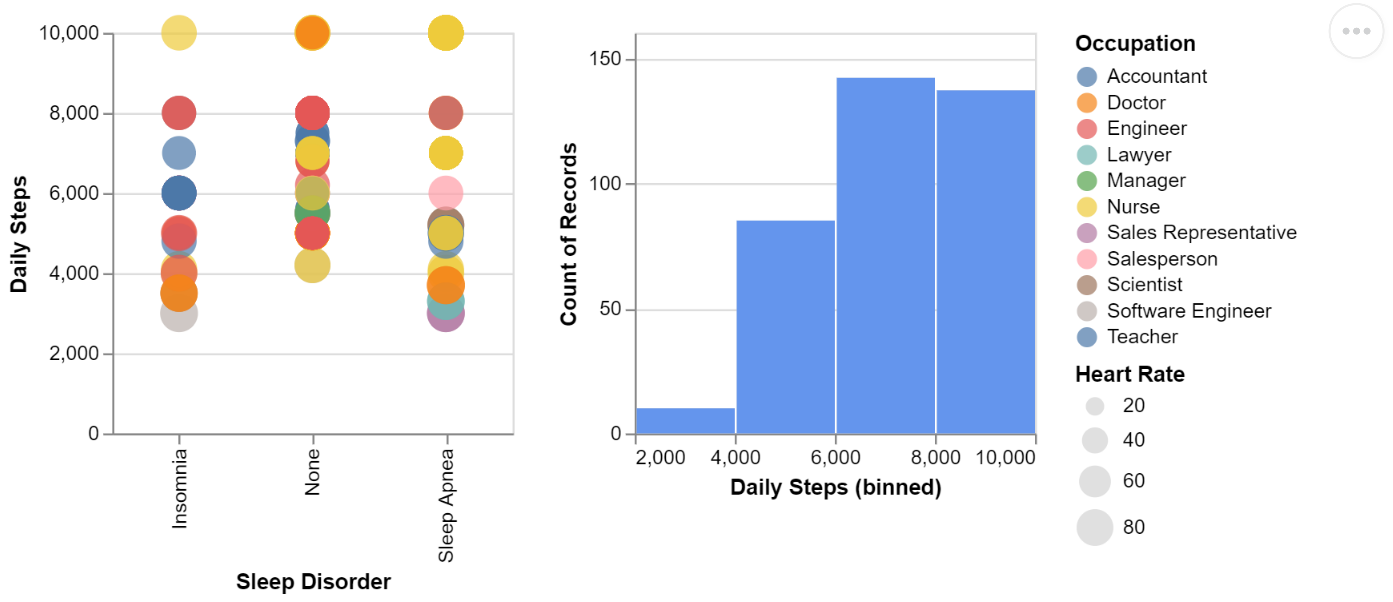
```
# Linked views
# Creating a selection:
selection = alt.selection(type="multi", fields=["Occupation"])

# Create a container for our two different views
base =  alt.Chart(data).properties(width=250, height=250)

# Create our scatterplot
```

```
scatterplot = base.mark_circle().encode(
    x = 'Sleep Disorder',
    y = 'Daily Steps',
    size = "Heart Rate",
    color = alt.condition(selection, "Occupation", alt.value('lightgray'))
).add_selection(selection)

# Create a histogram
hist = base.mark_bar().encode(
    x = alt.X("Daily Steps", bin=alt.Bin(maxbins=5)),
    y = "count()"
).transform_filter(selection)

# Connect our charts using the pipe operation
scatterplot | hist
```



```
# This selection is going to be an interval selection
selection = alt.selection(type="interval", encodings=["x", "y"])
```

```
# Create our scatterplot
scatterplot = alt.Chart(data).mark_circle().encode(
    x = 'Sleep Disorder',
    y = 'Daily Steps',
    size = "Heart Rate",
    color = alt.condition(selection, "Occupation", alt.value('lightgray'))
).properties(
    width = 200,
    height = 200
).add_selection(selection)

# Define our background chart
base = alt.Chart().mark_bar(color="cornflowerblue").encode(
    x = alt.X("Daily Steps", bin=alt.Bin(maxbins=5)),
    y = "count()"
).properties (
    width=200,
    height = 200
)

# Grey background to show the selection range in the scatterplot
background = base.encode(color=alt.value('lightgray')).add_selection(selection)

# Blue highlights to show the transformed (brushed) data
highlight = base.transform_filter(selection)

# Layer the two charts
layers = alt.layer(background, highlight, data = data)

scatterplot | layers
```


Daily Steps
10,000
8,000
6,000
4,000
2,000
0
Insomnia
None
Sleep Apnea
Sleep Disorder
Count of Records
150
100
50
0
2,000
4,000
6,000
8,000
10,000
Daily Steps (binned)
Occupation
Accountant
Doctor
Engineer
Lawyer
Manager
Nurse
Sales Representative
Salesperson
Scientist
Software Engineer
Teacher
Heart Rate
20
40
60
80