Create a single Jupyter/IPython notebook (see the Artefacts section below for all the requirements), where you perform what follows.

1. Download atleast five different datasets that are part of the NHANES study; see https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?Cycle=2021-2023 and https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?Cycle=2017-2020. Merge them into a single data frame.

```
import pandas as pd

demographics_data = pd.read_sas('DEMOGRAPHICS.XPT')
dietary_data = pd.read_sas('DIET.XPT')
examination_data = pd.read_sas('EXAMINATION.XPT')
laboratory_data = pd.read_sas('LABORATORY.XPT')
questionnaire_data = pd.read_sas('QUESTIONNAIRE.XPT')

# Merge the datasets on the 'SEQN' column
merged_dataframes = demographics_data.merge(dietary_data, on='SEQN',
how='outer')
merged_dataframes = merged_dataframes.merge(examination_data,
on='SEQN', how='outer')
merged_dataframes = merged_dataframes.merge(laboratory_data,
on='SEQN', how='outer')
merged_dataframes = merged_dataframes.merge(questionnaire_data,
on='SEQN', how='outer')

merged_dataframes.head()

       SEQN  SDDSRVYR  RIDSTATR  RIAGENDR  RIDAGEYR  RIDAGEMN
RIDRETH1  \
0  109263.0      66.0       2.0       1.0       2.0       NaN
5.0
1  109264.0      66.0       2.0       2.0      13.0       NaN
1.0
2  109265.0      66.0       2.0       1.0       2.0       NaN
3.0
3  109266.0      66.0       2.0       2.0      29.0       NaN
5.0
4  109267.0      66.0       1.0       2.0      21.0       NaN
2.0

   RIDRETH3  RIDEXMON  DMDBORN4  ...  BPQ020  BPQ030  BPD035  BPQ040A
\
0       6.0       2.0       1.0  ...     NaN     NaN     NaN      NaN

1       1.0       2.0       1.0  ...     NaN     NaN     NaN      NaN

2       3.0       2.0       1.0  ...     NaN     NaN     NaN      NaN

3       6.0       2.0       2.0  ...     2.0     NaN     NaN      NaN
```

```
4        2.0         NaN         2.0  ...      2.0      NaN      NaN       NaN


   BPQ050A  BPQ080  BPQ060  BPQ070  BPQ090D  BPQ100D
0      NaN     NaN     NaN     NaN      NaN      NaN
1      NaN     NaN     NaN     NaN      NaN      NaN
2      NaN     NaN     NaN     NaN      NaN      NaN
3      NaN     1.0     NaN     1.0      2.0      NaN
4      NaN     2.0     1.0     2.0      2.0      NaN

[5 rows x 110 columns]
```

1. Using the bokeh package, which you will have to learn yourself (this is part of this HD - level task), create at least five nontrivial interactive data visualisations and/or tables.

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool, Slider,
TextInput
from bokeh.layouts import row, column
from bokeh.io import push_notebook

#Enable Bokeh to output to the notebook
output_notebook()

"(function(root) {\n  function now() {\n    return new Date();\n  }\n\
n  const force = true;\n\n  if (typeof root._bokeh_onload_callbacks
=== \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded() {\n    const el =
document.getElementById(\"ef11e810-0dab-435f-9687-41bd5950ac93\");\n
if (el != null) {\n      el.textContent = \"BokehJS is loading...\";\n
}\n    if (root.Bokeh !== undefined) {\n      if (el != null) {\n
el.textContent = \"BokehJS \" + root.Bokeh.version + \" successfully
loaded.\";\n      }\n    } else if (Date.now() < root._bokeh_timeout)
{\n      setTimeout(display_loaded, 100)\n    }\n  }\n\n  function
run_callbacks() {\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.3.4.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.3.4.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.3.4.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.3.4.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.3.4.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n          for (let i = 0; i <
inline_js.length; i++) {\n      inline_js[i].call(root, root.Bokeh);\n
}\nif (force === true) {\n        display_loaded();\n      }} else if
(Date.now() < root._bokeh_timeout) {\n      setTimeout(run_inline_js,
100);\n    } else if (!root._bokeh_failed_load) {\n
console.log(\"Bokeh: BokehJS failed to load within specified
timeout.\");\n      root._bokeh_failed_load = true;\n    } else if
```

```
(force !== true) {\n      const cell = $
(document.getElementById(\"ef11e810-0dab-435f-9687-
41bd5950ac93\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

from bokeh.plotting import figure, output_file, show, ColumnDataSource
from bokeh.models import HoverTool, CategoricalColorMapper

# Prepare the data for the scatter plot
scatter_data = merged_dataframes[['SEQN', 'RIAGENDR', 'RIDAGEYR',
'BMXWT']].dropna()
scatter_data['RIAGENDR'] = scatter_data['RIAGENDR'].map({1: 'Male', 2:
'Female'})

# Create a ColumnDataSource
source = ColumnDataSource(scatter_data)

# Create the scatter plot
scatter_plot = figure(title="Age vs Weight by Gender",
                      x_axis_label="Age (years)",
                      y_axis_label="Weight (kg)",
                      width=800,
                      height=400,
                      tools="pan,wheel_zoom,box_zoom,reset,save")

# Add scatter glyphs
color_mapper = CategoricalColorMapper(factors=['Male', 'Female'],
palette=['green', 'purple'])
scatter_plot.scatter('RIDAGEYR', 'BMXWT', source=source,
                     color={'field': 'RIAGENDR', 'transform':
color_mapper},
                     legend_field='RIAGENDR',
                     size=10,  # Increased marker size
                     alpha=0.8,  # Increased marker transparency
                     marker='triangle')  # Changed marker shape to
triangle

# Add hover tool
hover = HoverTool(tooltips=[("Age", "@RIDAGEYR"), ("Weight",
"@BMXWT"), ("Gender", "@RIAGENDR")])
scatter_plot.add_tools(hover)

show(scatter_plot)

""
```

```
# Prepare the data for the line plot
line_data = merged_dataframes[['SEQN', 'RIDAGEYR', 'BMXBMI']].dropna()

# Aggregate data to calculate the average BMI for each age
avg_bmi_data = line_data.groupby('RIDAGEYR').agg({'BMXBMI':
'mean'}).reset_index()

# Create a ColumnDataSource
line_source = ColumnDataSource(avg_bmi_data)

# Create the line plot
line_plot = figure(title="Average BMI by Age",
                   x_axis_label="Age (years)",
                   y_axis_label="Average BMI",
                   width=800,
                   height=400,
                   tools="pan,wheel_zoom,box_zoom,reset,save")

# Add line glyphs
line_plot.line('RIDAGEYR', 'BMXBMI', source=line_source, line_width=2,
color='orange')

# Add hover tool
hover_line = HoverTool(tooltips=[("Age", "@RIDAGEYR"), ("Average BMI",
"@BMXBMI")])
line_plot.add_tools(hover_line)

show(line_plot)

""

from bokeh.transform import transform
from bokeh.models import LinearColorMapper, ColorBar
from bokeh.palettes import plasma

# Prepare the data for the heatmap
heatmap_data = merged_dataframes[['DS1IKCAL', 'DS1IPROT', 'DS1ICARB',
'DS1ITFAT']].dropna()

# Calculate the correlation matrix
correlation_matrix = heatmap_data.corr().values

# Create a DataFrame for the correlation matrix
correlation_df = pd.DataFrame(correlation_matrix,
                              index=['Calories', 'Protein',
'Carbohydrates', 'Total Fat'],
                              columns=['Calories', 'Protein',
'Carbohydrates', 'Total Fat'])

# Convert the DataFrame to a format suitable for heatmap
correlation_df = correlation_df.stack().reset_index()
```

```
correlation_df.columns = ['Feature1', 'Feature2', 'Correlation']

# Create a ColumnDataSource
heatmap_source = ColumnDataSource(correlation_df)

# Create the heatmap
heatmap = figure(title="Dietary Intake Correlations",
                 x_axis_label="Features",
                 y_axis_label="Features",
                 x_range=list(correlation_df['Feature1'].unique()),
                 y_range=list(correlation_df['Feature2'].unique()),
                 width=800,
                 height=400,
                 tools="pan,wheel_zoom,box_zoom,reset,save")

# Add rect glyphs
palette = plasma(256)  # Get a list of 256 colors from the plasma
palette
mapper = LinearColorMapper(palette=palette, low=-1, high=1)
heatmap.rect(x='Feature1', y='Feature2', width=1, height=1,
source=heatmap_source,
             fill_color=transform('Correlation', mapper),
line_color=None)

# Add color bar
color_bar = ColorBar(color_mapper=mapper, location=(0, 0))
heatmap.add_layout(color_bar, 'right')

# Add hover tool
hover_heatmap = HoverTool(tooltips=[("Feature 1", "@Feature1"),
("Feature 2", "@Feature2"), ("Correlation", "@Correlation")])
heatmap.add_tools(hover_heatmap)

show(heatmap)

""

# Prepare the data for the bar plot
bar_data = merged_dataframes[['SEQN', 'RIAGENDR', 'LBXTC']].dropna()
bar_data['RIAGENDR'] = bar_data['RIAGENDR'].map({1: 'Male', 2:
'Female'})

# Aggregate data to calculate the average cholesterol level for each
gender
avg_cholesterol_data = bar_data.groupby('RIAGENDR').agg({'LBXTC':
'mean'}).reset_index()

# Create a ColumnDataSource
bar_source = ColumnDataSource(avg_cholesterol_data)

# Create the bar plot
```

```
bar_plot = figure(title="Average Cholesterol Levels by Gender",
                  x_axis_label="Gender",
                  y_axis_label="Average Cholesterol Level",

x_range=list(avg_cholesterol_data['RIAGENDR'].unique()),
                  width=1000,  # Increased width
                  height=600,  # Increased height
                  tools="pan,wheel_zoom,box_zoom,reset,save")

# Add bar glyphs
bar_plot.vbar(x='RIAGENDR', top='LBXTC', source=bar_source, width=0.7,
color='orange')  # Changed color to orange

# Add hover tool
hover_bar = HoverTool(tooltips=[("Gender", "@RIAGENDR"), ("Average
Cholesterol", "@LBXTC")])
bar_plot.add_tools(hover_bar)

show(bar_plot)

""

from bokeh.models import DataTable, TableColumn
from bokeh.layouts import column
from bokeh.io import output_file, show
from bokeh.plotting import ColumnDataSource

# Prepare the data for the table
table_data = merged_dataframes[['SEQN', 'RIAGENDR', 'RIDAGEYR',
'BMXWT', 'BMXBMI', 'LBXTC',
                               'DS1IKCAL', 'DS1IPROT', 'DS1ICARB',
'DS1ITFAT']].dropna()
table_data['RIAGENDR'] = table_data['RIAGENDR'].map({1: 'Male', 2:
'Female'})

# Create a ColumnDataSource
table_source = ColumnDataSource(table_data)

# Define columns
columns = [
   TableColumn(field="SEQN", title="SEQN"),
   TableColumn(field="RIAGENDR", title="Gender"),
   TableColumn(field="RIDAGEYR", title="Age"),
   TableColumn(field="BMXWT", title="Weight (kg)"),
   TableColumn(field="BMXBMI", title="BMI"),
   TableColumn(field="LBXTC", title="Cholesterol"),
   TableColumn(field="DS1IKCAL", title="Calories"),
   TableColumn(field="DS1IPROT", title="Protein"),
   TableColumn(field="DS1ICARB", title="Carbohydrates"),
   TableColumn(field="DS1ITFAT", title="Total Fat"),
]

# Create the DataTable
data_table = DataTable(source=table_source, columns=columns,
width=800, height=400)

show(column(data_table))

""
```

1. Draw insightful and interesting conclusions. Do not forget to reflect on the potential data privacy and ethics issues that arise during the data analysis process.

Upon reviewing the interactive visualizations I created using the merged NHANES dataset, several key findings stood out to me.

The scatter plot of age vs weight colored by gender provides a clear picture of how body weight differs between males and females across the lifespan. I can see that females tend to weigh less on average compared to males, with weight peaking for both genders around age 60 before declining in older age, likely due to loss of muscle mass.

I also generated a line plot showing how average BMI changes with age. The steady increase in BMI from young adulthood up to age 60 is noteworthy, with the mean BMI falling into the overweight category for most of the adult years. This suggests that being overweight is quite common in this sample.

The heatmap I created revealed strong positive correlations between calories, protein, carbs and fat intake. So individuals consuming high calorie diets tend to eat greater amounts across all the macronutrient categories, rather than eating proportionately more of one macronutrient.

Lastly, the bar chart comparing average total cholesterol really highlighted the disparity between males and females, with males having markedly higher cholesterol. This is concerning given that high cholesterol increases risk of cardiovascular disease.

While these insights are intriguing, I want to acknowledge the importance of protecting participant privacy, especially with sensitive health data. Responsibly using this data in aggregate form, as I've done here, allows for meaningful conclusions to be drawn without compromising confidentiality.

Overall, I believe these findings, which I arrived at through data wrangling and visualization using Python and Bokeh, provide a compelling snapshot of key sex and age differences in weight, diet composition, and cholesterol levels in a sample of American adults. Further exploration of the interplay between these factors could yield insights useful for tailoring dietary and lifestyle recommendations to promote optimal health outcomes.