**UNIT 3: Little Data to Big Data**

**Project 3.2.4 Making Meaning from Data**

**Enduring Understandings**  
*EU-DAT-2 : Programs can be used to process data, which allows users to discover information and create new knowledge.*  
*EU-AAP-4: There exist problems that the computer cannot solve, and even when the computer can solve a problem, it may not be able to do so in a reasonable amount of time.*

**\*\*Instructions:** You may work in pairs to complete this Project.  Please change the text color of your responses to red text.  Please organize the endings to each page.

STEP 11: Test that the data cleansing worked by printing the Value column before and after your cleaning algorithm. Paste your solution in the space provided.

|  |
| --- |
| # Create dataframe from csv file  honey\_df = pandas.read\_csv("honey.csv")  # Remove commas  honey\_df['Value'] = honey\_df['Value'].str.replace(',', '')  # Convert strings in Value column to numbers  honey\_df['Value'] = pandas.to\_numeric(honey\_df['Value'], errors='coerce')  # Drop NaN values  honey\_df.dropna(subset=['Value'], inplace=True) |

Document with pseudocode the data-cleansing algorithm you used and include a brief description of the cleaning effect.

|  |
| --- |
| honey\_df 🡨 read\_csv(honey.csv)  Replace commas in Value column of honey\_df with blank text  Convert Value column of honey\_df to numbers  Drop NaN values from value column of honey\_df |

STEP 13: Iterate over your unique states and experiment with the following two algorithms that use pandas, with grouping and without. Each retrieves data from the CSV file and stores the data in two arrays. Observe the print results to see how data becomes grouped.

all\_honey = []

all\_states = []

# without grouping

for state in unique\_states:

honey\_data = df[df['State'] == state])['Value']

print (state, honey\_data.sum())

all\_honey.append(honey\_data.sum())

all\_states.append(state)

# with grouping

for state in unique\_states:

honey\_data = df[df['State'] == state].groupby('Year')['Value']

print (state, honey\_data.sum())

all\_honey.append(honey\_data.sum())

all\_states.append(state)

Describe how your data collection algorithm works using the groupby method.

|  |
| --- |
| The groupby method groups the data by year, allowing the sum of the values for all years for the state to be summed. |

STEP 15: Add comments to your code, summarizing how your data collection algorithm works.

|  |
| --- |
| import matplotlib  import pandas  # Create dataframe from csv file  honey\_df = pandas.read\_csv("honey.csv")  # Remove commas  honey\_df['Value'] = honey\_df['Value'].str.replace(',', '')  # Convert strings in Value column to numbers  honey\_df['Value'] = pandas.to\_numeric(honey\_df['Value'], errors='coerce')  # Drop NaN values  honey\_df.dropna(subset=['Value'], inplace=True)  unique\_states = honey\_df['State'].unique()  all\_honey = []  all\_states = []  # without grouping  for state in unique\_states:      # Get honey data for one state grouped by the year      honey\_data = honey\_df[honey\_df['State'] == state]['Value']      # Print the state and sum of honey values      print(state, honey\_data.sum())      # Add total honey value to the list of values      all\_honey.append(honey\_data.sum())      # Add state to the list of states      all\_states.append(state)  # with grouping  for state in unique\_states:      # Get honey data for one state grouped by the year      honey\_data = honey\_df[honey\_df['State'] == state].groupby('Year')['Value'].sum()      # Print the state and sum of honey values      print(state, honey\_data.sum())      # Add total honey value to the list of values      all\_honey.append(honey\_data.sum())      # Add state to the list of states      all\_states.append(state) |

STEP 19: Use three graphs to visualize the data. One way to do this is to show large honey producers, mid-level honey producers, and small producers, each on their own graph. Insert your code or a link to your code below.

|  |
| --- |
| import matplotlib.pyplot as plt  import pandas  # Create dataframe from csv file  honey\_df = pandas.read\_csv("honey.csv")  # Remove commas  honey\_df['Value'] = honey\_df['Value'].str.replace(',', '')  # Convert strings in Value column to numbers  honey\_df['Value'] = pandas.to\_numeric(honey\_df['Value'], errors='coerce')  # Drop NaN values  honey\_df.dropna(subset=['Value'], inplace=True)  unique\_states = honey\_df['State'].unique()  # Initialize dataframe to represent total honey production for each state  state\_honey\_df = pandas.DataFrame(columns=["State", "Total Honey Production"])  fig, axs = plt.subplots(3, sharex=True)  fig.suptitle('Honey production by production bracket')  brackets = [0, 1000000, 4500000]  all\_honey = {}  #  # # without grouping  # for state in unique\_states:  #     # Get honey data for one state grouped by the year  #     honey\_data = honey\_df[honey\_df['State'] == state]['Value']  #     # Print the state and sum of honey values  #     print(state, honey\_data.sum())  #     # Add total honey value to the list of values  #     all\_honey.append(honey\_data.sum())  #     # Add state to the list of states  #     all\_states.append(state)  # with grouping  for state in unique\_states:      # Get honey data for one state grouped by the year      honey\_data = honey\_df[honey\_df['State'] == state].groupby('Year')['Value'].sum()      honey\_years = honey\_df[honey\_df['State'] == state]['Year'].unique()      total\_production = honey\_data.sum()      # Print the state and sum of honey values      # print(state, total\_production)      # Add data to the states dataframe      state\_honey\_df = pandas.concat([state\_honey\_df, pandas.DataFrame([state, total\_production])])        bracket = -1      for i in range(len(brackets)):          if total\_production < brackets[i]: break          bracket += 1        print(state + str(bracket))      # Plot the honey data vs the years for the state      axs[bracket].plot(honey\_years, honey\_data, label=state)        all\_honey[state] = total\_production      # for i in range(len(brackets)):      #     if i == len(brackets): break      #     if brackets[i] >= total\_production: break      #     i += 1  all\_honey = {state : honey for state, honey in sorted(all\_honey.items(), key=lambda item: item[1])}  for k, v in all\_honey.items():      print(f"{k}: {v}")  plt.ylabel('Annual honey production')  plt.xlabel('Year')  for i in range(3):      axs[i].legend(loc="upper left", bbox\_to\_anchor=(1, 1))  plt.show() |

Chart

Description automatically generated

STEP 22: Use a new groupby statement to get yearly totals.  Add Comments and insert your code or a link to your code below.

|  |
| --- |
| def graph\_by\_years():      unique\_years = honey\_df["Year"].unique()      year\_honey\_df = pandas.DataFrame(columns=["Year", "Total"])      for year in unique\_years:          yearly\_sum = honey\_df[honey\_df["Year"] == year].groupby("Year")["Value"].sum().sum()          sum = pandas.DataFrame({"Year" : year, "Total" : yearly\_sum}, index=[0])          year\_honey\_df = pandas.concat([year\_honey\_df, sum])        plt.bar(year\_honey\_df["Year"], year\_honey\_df["Total"])      plt.xlabel("Year")      plt.ylabel("Annual Honey Production")      plt.show() |

Chart, bar chart

Description automatically generated

STEP 23:  Validate either an average or a total value using a process similar to the state-by-year validation. Show your work (image would suffice).  What is the value of manually validating the data?

|  |
| --- |
| def validate\_year(year: int):      values = honey\_df[honey\_df["Year"] == year]["Value"]      sum = 0      for value in values:          sum += value      print(f"{year} sum: {sum}") |

STEP 25:  Improve the look of your data visualizations by formatting the legend.

|  |
| --- |
| Chart  Description automatically generated |

STEP 26: With your data visualizations complete, you should now be ready to make some conclusions about the data. What does the data say about honey production? Do you think this correlates to problems with bee colonies in the U.S.? Note any unusual patterns or outliers in your data.

|  |
| --- |
| Honey production has been decreasing in many states since 2012, which shows that there may be problems with colony collapse disorder epidemics in some states. For example, this occurred in Louisiana, Georgia, California, Mississippi, and other states. |

STEP 30:  You will not need to submit your code for Step 29.  Make a conclusion about the operational facilities.  Then correlate the operational facilities to the honey production and revisit your conclusion.  Have operational facilities increased or decreased? How does the increase or decline relate to your conclusion about production rates?

|  |
| --- |
| The operational facilities’ hoeny production have been steadily decreasing, which is what my conclusion about production rates predicted. |