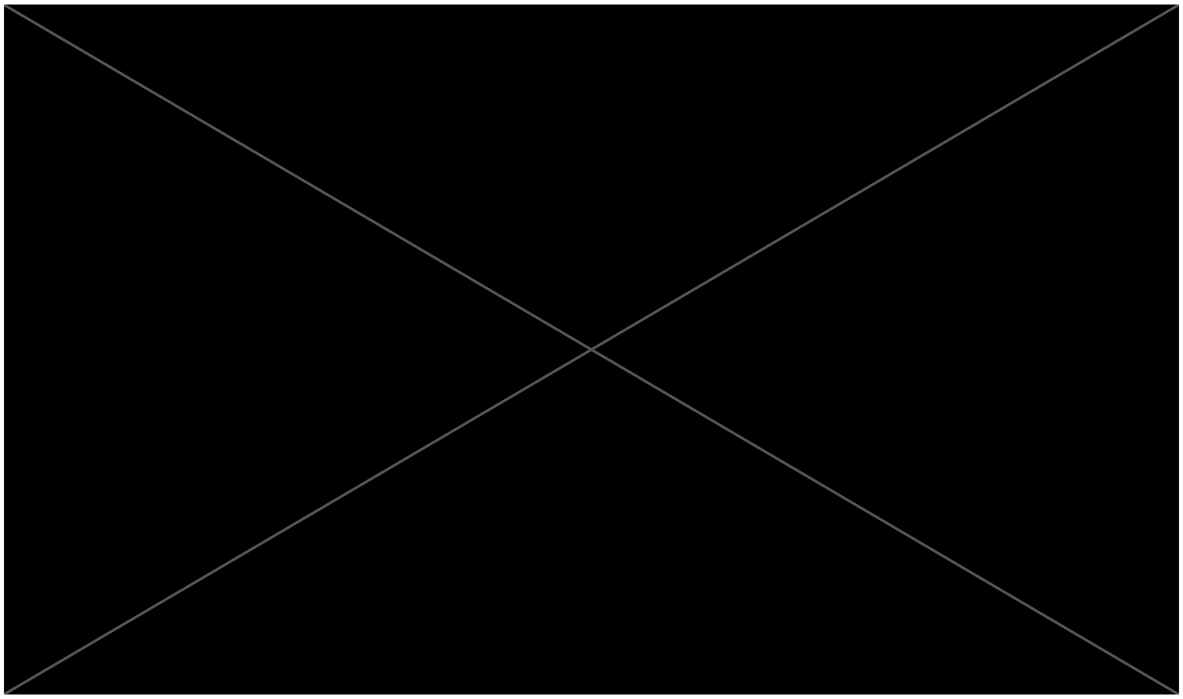
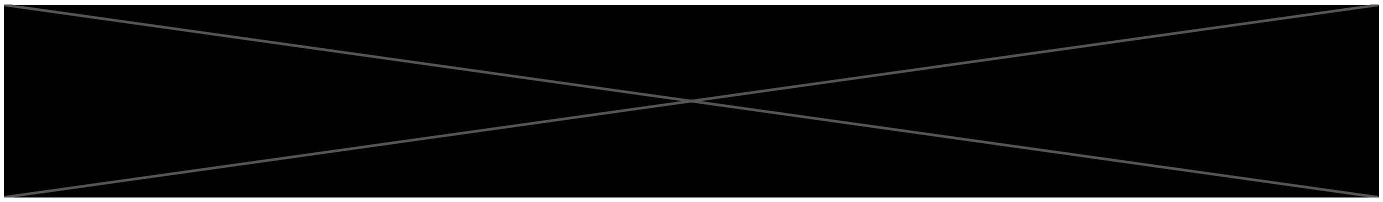


WESTYN HILLIARD



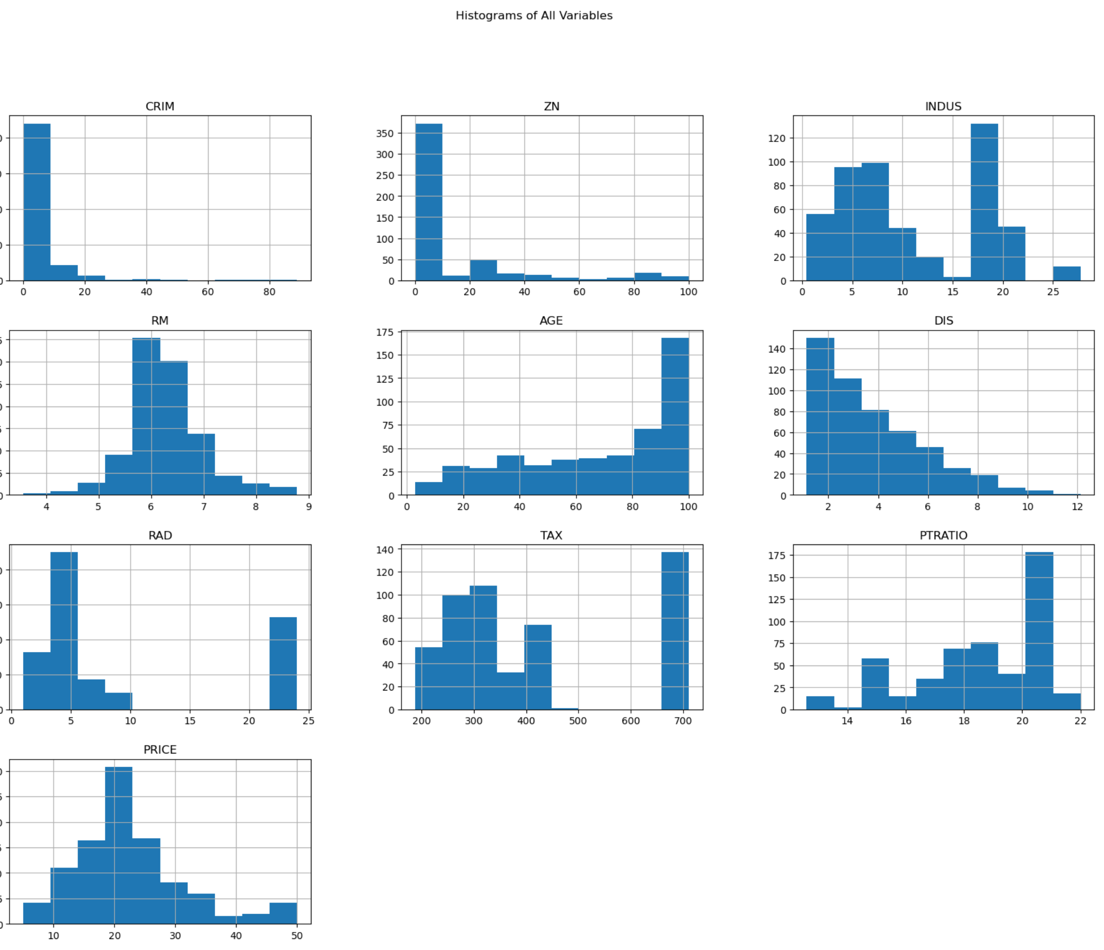
1.1 *1. The Data Wrangling Workshop: Activity 3.01, page 155 -*

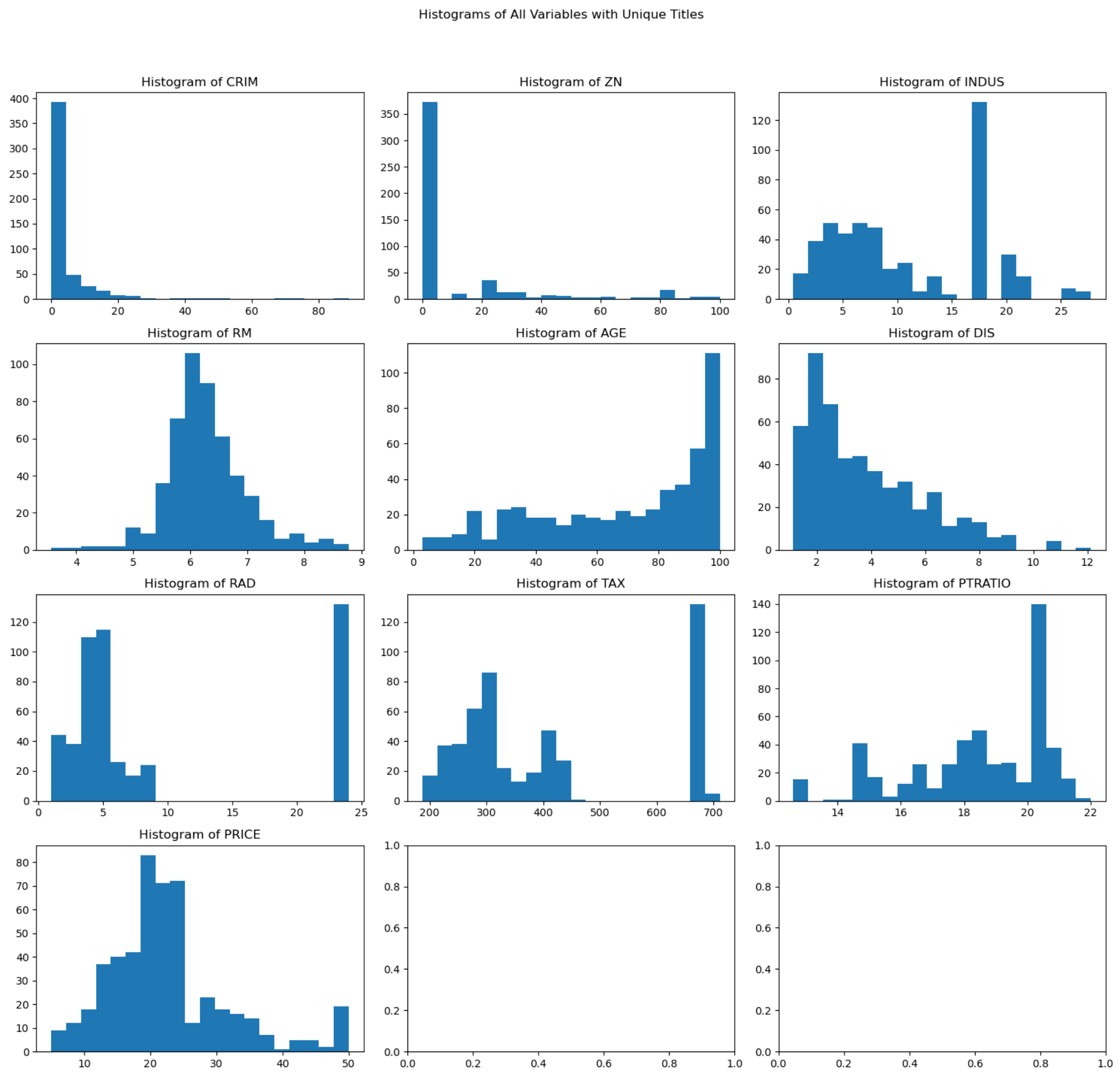
[50]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Step 2: Read in the Boston Housing dataset from the local directory
# The CSV file is named 'boston_housing.csv'
df = pd.read_csv('boston_housing.csv')

# Step 3: Check the first 10 records
print("First 10 records of the dataset:")
print(df.head(10))

# Find the total number of records
total_records = df.shape[0]
```

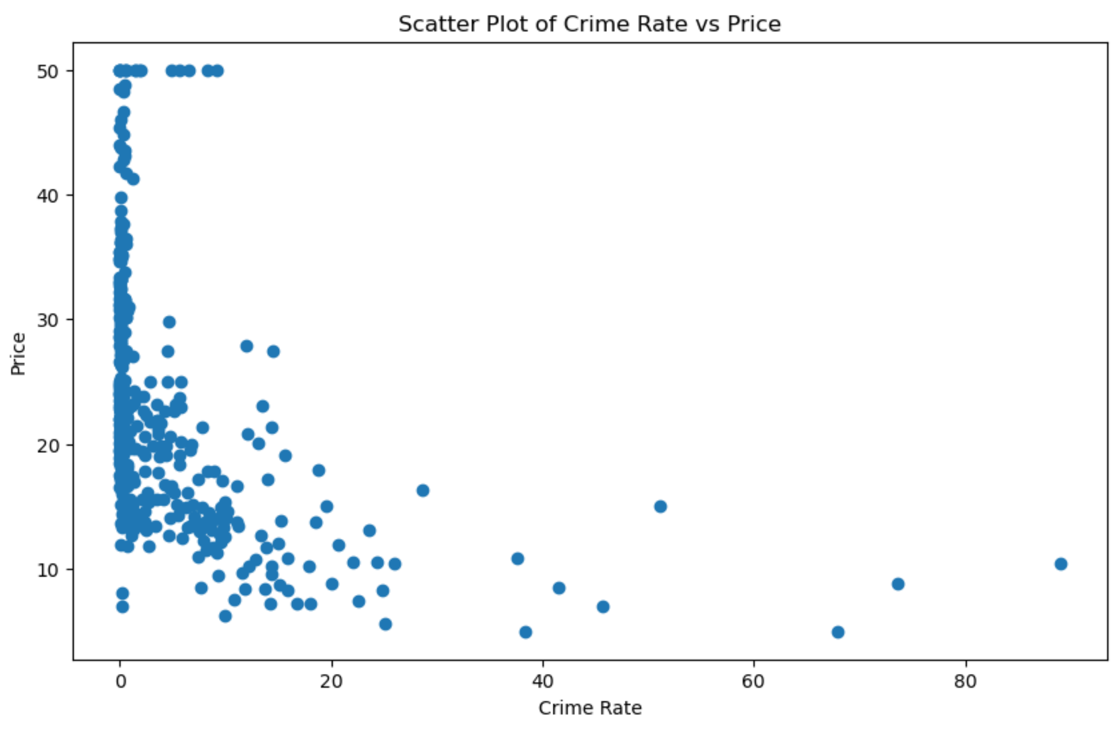
```python
print("\nTotal number of records:", total_records)

# Step 4: Create a smaller DataFrame excluding columns CHAS, NOX, B, and LSTAT
columns_to_exclude = ['CHAS', 'NOX', 'B', 'LSTAT']
smaller_df = df.drop(columns=columns_to_exclude)

# Step 5: Check the last seven records of the new DataFrame
print("\nLast 7 records of the new DataFrame:")
print(smaller_df.tail(7))

# Step 6: Plot the histograms of all the variables (columns) in the new␣
 ↪DataFrame
smaller_df.hist(figsize=(20, 15))
plt.suptitle('Histograms of All Variables')
plt.show()

# Step 7: Plot them all at once using a for loop with unique titles
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 15))
fig.suptitle('Histograms of All Variables with Unique Titles')
for i, column in enumerate(smaller_df.columns):
    ax = axes[i // 3, i % 3]
    ax.hist(smaller_df[column], bins=20)
    ax.set_title(f'Histogram of {column}')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Step 8: Create a scatter plot of crime rate versus price
plt.figure(figsize=(10, 6))
plt.scatter(df['CRIM'], df['PRICE'])
plt.xlabel('Crime Rate')
plt.ylabel('Price')
plt.title('Scatter Plot of Crime Rate vs Price')
plt.show()

# Step 9: Plot log10(crime) versus price
plt.figure(figsize=(10, 6))
plt.scatter(np.log10(df['CRIM']), df['PRICE'])
plt.xlabel('Log10(Crime Rate)')
plt.ylabel('Price')
plt.title('Scatter Plot of Log10(Crime Rate) vs Price')
plt.show()

# Step 10: Calculate useful statistics
mean_rooms_per_dwelling = df['RM'].mean()
median_age = df['AGE'].median()
mean_distance_to_employment_centers = df['DIS'].mean()
percentage_houses_low_price = (df['PRICE'] < 20).mean() * 100
```

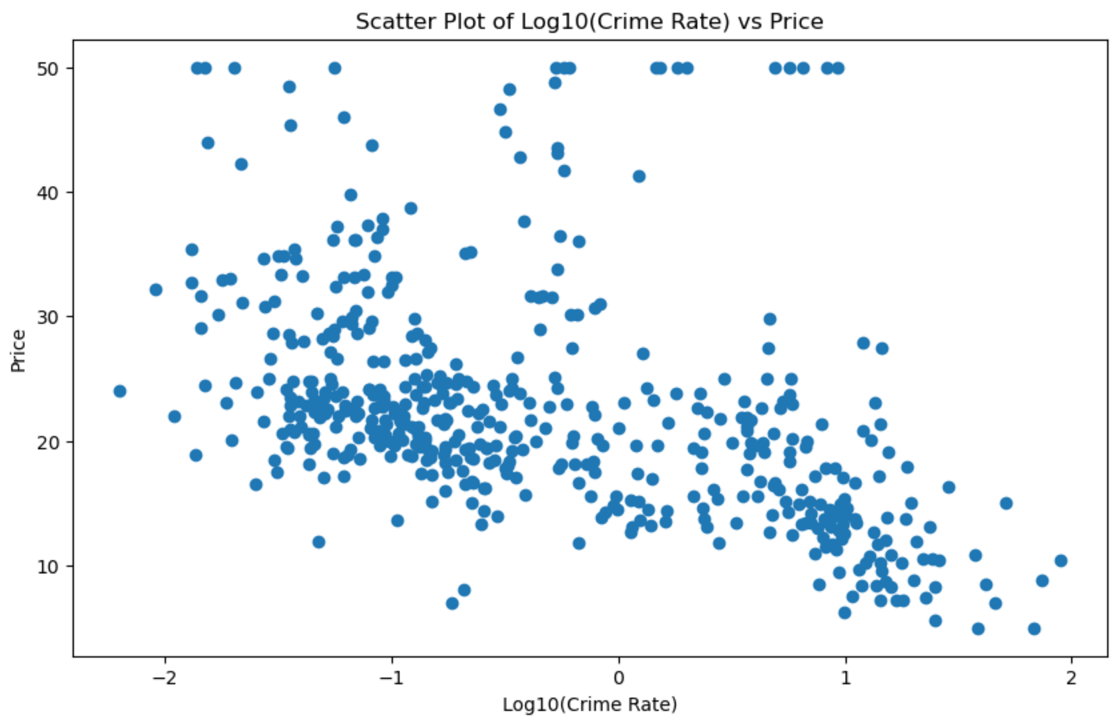
```
print(f"Mean rooms per dwelling: {mean_rooms_per_dwelling}")
print(f"Median age: {median_age}")
print(f"Mean distance to five Boston employment centers:␣
 ↪{mean_distance_to_employment_centers}")
print(f"Percentage of houses with a price below $20,000:␣
 ↪{percentage_houses_low_price:.2f}%")
```

```
First 10 records of the dataset:
      CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575   65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421   78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185   61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998   45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147   54.2  6.0622    3  222     18.7
5  0.02985   0.0   2.18     0  0.458  6.430   58.7  6.0622    3  222     18.7
6  0.08829  12.5   7.87     0  0.524  6.012   66.6  5.5605    5  311     15.2
7  0.14455  12.5   7.87     0  0.524  6.172   96.1  5.9505    5  311     15.2
8  0.21124  12.5   7.87     0  0.524  5.631  100.0  6.0821    5  311     15.2
9  0.17004  12.5   7.87     0  0.524  6.004   85.9  6.5921    5  311     15.2

        B  LSTAT  PRICE
0  396.90   4.98   24.0
1  396.90   9.14   21.6
2  392.83   4.03   34.7
3  394.63   2.94   33.4
4  396.90   5.33   36.2
5  394.12   5.21   28.7
6  395.60  12.43   22.9
7  396.90  19.15   27.1
8  386.63  29.93   16.5
9  386.71  17.10   18.9

Total number of records: 506

Last 7 records of the new DataFrame:
        CRIM   ZN  INDUS     RM   AGE     DIS  RAD  TAX  PTRATIO  PRICE
499  0.17783  0.0   9.69  5.569  73.5  2.3999    6  391     19.2   17.5
500  0.22438  0.0   9.69  6.027  79.7  2.4982    6  391     19.2   16.8
501  0.06263  0.0  11.93  6.593  69.1  2.4786    1  273     21.0   22.4
502  0.04527  0.0  11.93  6.120  76.7  2.2875    1  273     21.0   20.6
503  0.06076  0.0  11.93  6.976  91.0  2.1675    1  273     21.0   23.9
504  0.10959  0.0  11.93  6.794  89.3  2.3889    1  273     21.0   22.0
505  0.04741  0.0  11.93  6.030  80.8  2.5050    1  273     21.0   11.9
```


Histograms of All Variables
CRIM
ZN
INDUS
RM
AGE
DIS
RAD
TAX
PTRATIO
PRICE


Histograms of All Variables with Unique Titles
Histogram of CRIM
Histogram of ZN
Histogram of INDUS
Histogram of RM
Histogram of AGE
Histogram of DIS
Histogram of RAD
Histogram of TAX
Histogram of PTRATIO
Histogram of PRICE





```
Mean rooms per dwelling: 6.284634387351779
```

```
Median age: 77.5
Mean distance to five Boston employment centers: 3.795042687747036
Percentage of houses with a price below $20,000: 41.50%
```

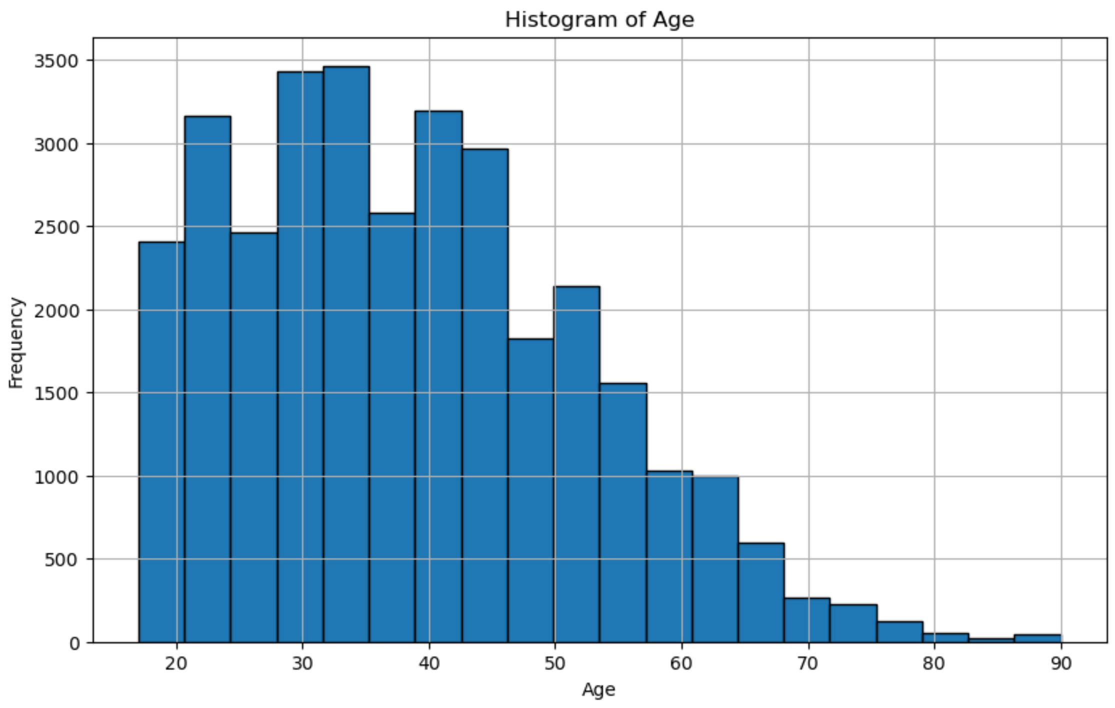
1.2 *2. The Data Wrangling Workshop: Activity 4.01, page 233* -

[51]:
```python
# Step 1: Load the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Step 2: Read the adult income dataset from the uploaded file
file_path = 'adult_income_data.csv'
df = pd.read_csv(file_path, header=None)

# Print the number of columns in the dataset
print(f"The dataset contains {df.shape[1]} columns.")

# Adjust column names to match the number of columns in the dataset
columns = ["age", "workclass", "fnlwgt", "education", "education_num",␣
 ↪"marital_status",
           "occupation", "relationship", "race", "sex", "capital_gain",␣
 ↪"capital_loss",
           "hours_per_week", "native_country"]
df.columns = columns

# Step 3: Display the first few rows to ensure the dataset is loaded correctly
df.head()

# Step 4: This step is completed by adding column names

# Step 5: Find the missing values
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)

# Step 6: Create a DataFrame with only age, education, and occupation by using␣
 ↪subsetting
df_subset = df[['age', 'education', 'occupation']]
df_subset.head()

# Step 7: Plot a histogram of age with a bin size of 20
plt.figure(figsize=(10, 6))
df['age'].hist(bins=20, edgecolor='black')
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
# Step 8: Create a function to strip the whitespace characters
def strip_whitespace(s):
    if isinstance(s, str):
        return s.strip()
    return s

# Step 9: Use the apply method to apply this function to all the columns with␣
 ↪string values
df_subset['education'] = df_subset['education'].apply(strip_whitespace)
df_subset['occupation'] = df_subset['occupation'].apply(strip_whitespace)

# Step 10: Find the number of people who are aged between 30 and 50
df_filtered = df_subset[(df_subset['age'] >= 30) & (df_subset['age'] <= 50)]
answer_1 = df_filtered.shape[0]
print(f"There are {answer_1} people of age between 30 and 50 in this dataset.")

# Step 11: Group the records based on age and education to find how the mean␣
 ↪age is distributed
age_education_group = df_filtered.groupby(['age', 'education']).size().
 ↪reset_index(name='counts')
print("Group by age and education:\n", age_education_group.head())

# Step 12: Group by occupation and show the summary statistics of age
occupation_stats = df_subset.groupby('occupation')['age'].describe()
print("Summary statistics of age by occupation:\n", occupation_stats)

oldest_occupation = occupation_stats['mean'].idxmax()
print("Profession with the oldest workers on average:", oldest_occupation)

occupation_75th_percentile = df_subset.groupby('occupation')['age'].quantile(0.
 ↪75).idxmax()
print("Profession with the largest share of the workforce above the 75th␣
 ↪percentile:", occupation_75th_percentile)

# Step 13: Use subset and groupBy to find the outliers
def find_outliers(group):
    Q1 = group.quantile(0.25)
    Q3 = group.quantile(0.75)
    IQR = Q3 - Q1
    outliers = group[(group < (Q1 - 1.5 * IQR)) | (group > (Q3 + 1.5 * IQR))]
    return outliers

outliers = df_subset.groupby('occupation')['age'].apply(find_outliers).dropna()
print("Outliers in age by occupation:\n", outliers)

# Step 14: Plot the outlier values on a bar chart
```

```
outlier_counts = outliers.groupby('occupation').size()

plt.figure(figsize=(12, 8))
outlier_counts.plot(kind='bar')
plt.title('Outliers in Age by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Number of Outliers')
plt.show()

# Step 15: Merge the two DataFrames using common keys to drop duplicate values
df_1 = df[['age', 'workclass', 'occupation']].sample(5, random_state=101)
df_2 = df[['education', 'occupation']].sample(5, random_state=101)

df_merged = pd.merge(df_1, df_2, on='occupation', how='inner').drop_duplicates()
print("Merged DataFrame:\n", df_merged)
```

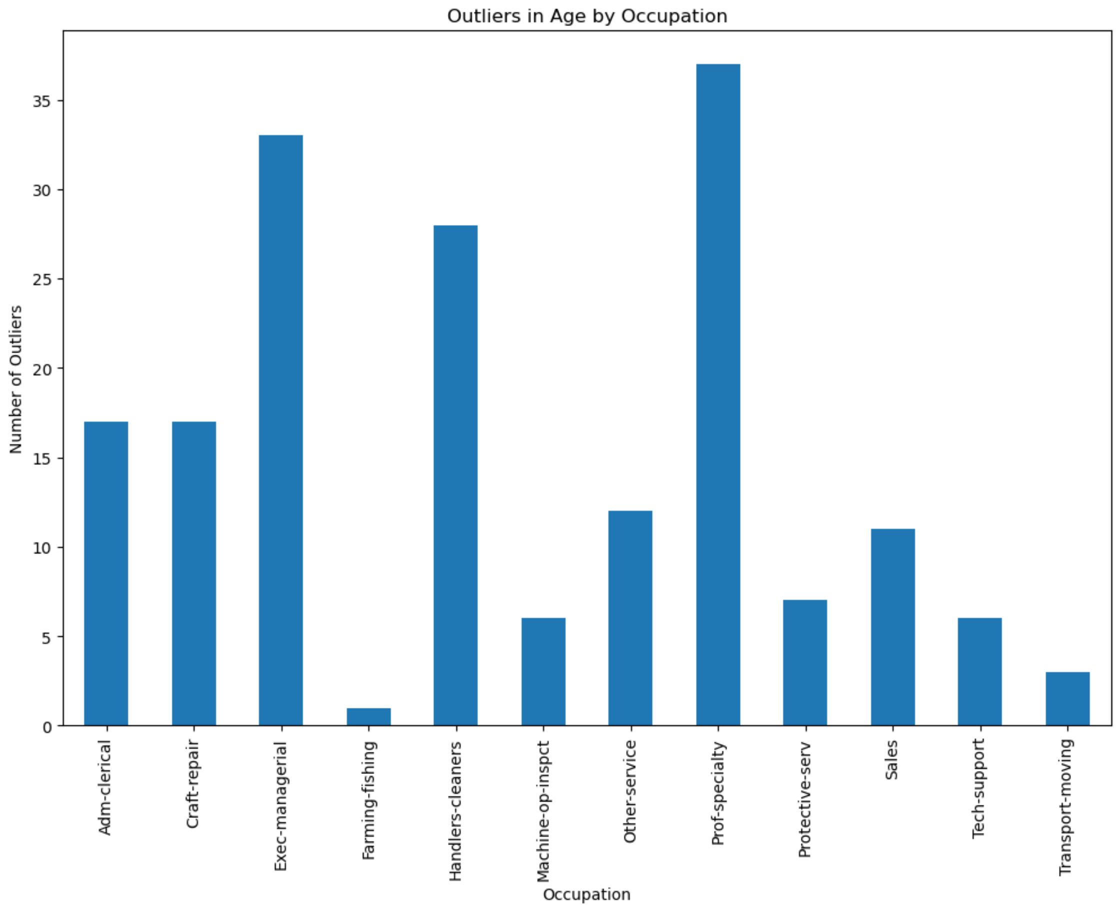
```
The dataset contains 14 columns.
Missing values in each column:
 age              0
workclass         0
fnlwgt            0
education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
dtype: int64
```



```
There are 16390 people of age between 30 and 50 in this dataset.
Group by age and education:
    age education  counts
0   30       10th      13
1   30       11th      28
2   30       12th       6
3   30    5th-6th       3
4   30    7th-8th      13
Summary statistics of age by occupation:
                    count       mean        std   min   25%   50%   75%   max
occupation
?                  1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Adm-clerical       3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Armed-Forces          9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
Craft-repair       4099.0  39.031471  11.606436  17.0  30.0  38.0  47.0  90.0
Exec-managerial    4066.0  42.169208  11.974548  17.0  33.0  41.0  50.0  90.0
Farming-fishing     994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
Handlers-cleaners  1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Machine-op-inspct  2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
Other-service      3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
Priv-house-serv     149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
Prof-specialty     4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Protective-serv     649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  90.0
Sales              3650.0  37.353973  14.186352  17.0  25.0  35.0  47.0  90.0
Tech-support        928.0  37.022629  11.316594  17.0  28.0  36.0  44.0  73.0
```

```
Transport-moving   1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
Profession with the oldest workers on average: Exec-managerial
Profession with the largest share of the workforce above the 75th percentile: ?
Outliers in age by occupation:
 occupation
Adm-clerical      2891     90
                  3537     81
                  4834     81
                  5272     90
                  6590     77
                            ..
Tech-support      24290    72
                  30022    70
Transport-moving  15356    90
                  26902    78
                  28948    81
Name: age, Length: 178, dtype: int64
```

```
/var/folders/67/hl77bzs97pggp3g9r_hnl9kw0000gn/T/ipykernel_2008/1718619150.py:47
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['education'] = df_subset['education'].apply(strip_whitespace)
/var/folders/67/hl77bzs97pggp3g9r_hnl9kw0000gn/T/ipykernel_2008/1718619150.py:48
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['occupation'] = df_subset['occupation'].apply(strip_whitespace)
```



```
Merged DataFrame:
   age workclass         occupation education
0   51   Private  Machine-op-inspct   HS-grad
1   19   Private              Sales      11th
2   40   Private    Exec-managerial   HS-grad
3   17   Private  Handlers-cleaners      10th
4   61   Private       Craft-repair   7th-8th
```

1.3 *3. The Data Wrangling Workshop: Activity 5.01, page 281* -

[52]:
```python
from bs4 import BeautifulSoup
import pandas as pd

# Step 1: Read the HTML file
file_path = 'List of countries by GDP (nominal) - Wikipedia.htm'
with open(file_path, "r", encoding="utf-8") as fd:
    soup = BeautifulSoup(fd, 'html.parser')

# Step 2: Find all tables
```

```python
all_tables = soup.find_all("table", {"class": "wikitable"})
print(f"Total number of tables are {len(all_tables)}")

# Function to parse the table
def parse_table(table):
    header = [th.get_text(strip=True) for th in table.find_all('tr')[0].
 ↪find_all('th')]
    rows = table.find_all('tr')[1:]  # Skip the header row
    data_rows = []
    for row in rows:
        cells = row.find_all(['td', 'th'])
        data_rows.append([cell.get_text(strip=True) for cell in cells])
    return header, data_rows

# Parse each table
dfs = []
for i, table in enumerate(all_tables):
    header, data_rows = parse_table(table)
    df = pd.DataFrame(data_rows, columns=header)
    dfs.append(df)
    print(f"Table {i} columns: {df.columns}")

# Assuming the tables correspond to IMF, World Bank, and UN respectively
df_imf = dfs[0]
df_world_bank = dfs[1]
df_un = dfs[2]

# Function to clean GDP values
def clean_gdp(gdp_series):
    return gdp_series.str.extract(r'(\d+,\d+,\d+|\d+,\d+|\d+)')[0].str.
 ↪replace(',', '').astype(float)

# Clean up the GDP data
df_imf['GDP(US$MM)'] = clean_gdp(df_imf['GDP(US$MM)'])
df_world_bank['GDP(US$MM)'] = clean_gdp(df_world_bank['GDP(US$MM)'])
df_un['GDP(US$MM)'] = clean_gdp(df_un['GDP(US$MM)'])

# Rename GDP columns for clarity
df_imf = df_imf.rename(columns={'GDP(US$MM)': 'GDP (IMF)'})
df_world_bank = df_world_bank.rename(columns={'GDP(US$MM)': 'GDP (World Bank)'})
df_un = df_un.rename(columns={'GDP(US$MM)': 'GDP (UN)'})

# Step 4: Save the DataFrames to CSV files
df_imf.to_csv('GDP_data_IMF.csv', index=False)
df_world_bank.to_csv('GDP_data_WorldBank.csv', index=False)
df_un.to_csv('GDP_data_UN.csv', index=False)
```

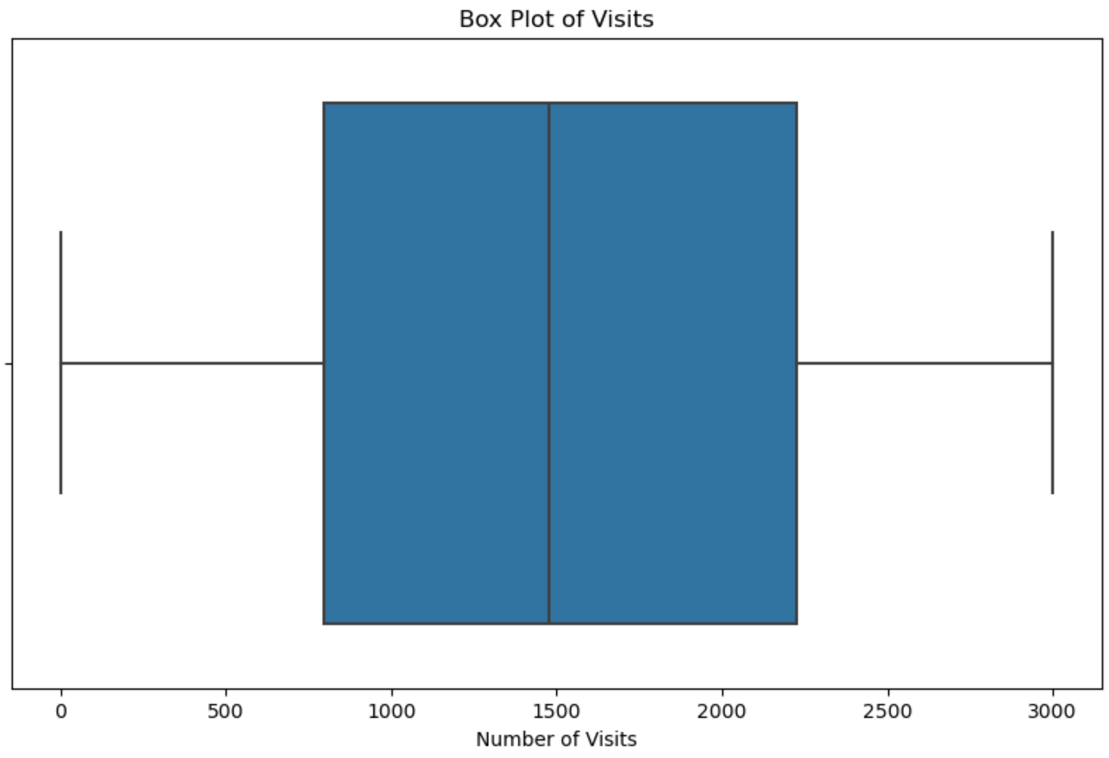
```
print("DataFrames have been saved as CSV files.")

# Load and inspect the cleaned CSV files
df_imf = pd.read_csv('GDP_data_IMF.csv')
df_world_bank = pd.read_csv('GDP_data_WorldBank.csv')
df_un = pd.read_csv('GDP_data_UN.csv')

print("IMF DataFrame:")
print(df_imf.head())

print("\nWorld Bank DataFrame:")
print(df_world_bank.head())

print("\nUnited Nations DataFrame:")
print(df_un.head())
```

```
Total number of tables are 3
Table 0 columns: Index(['Rank', 'Country', 'GDP(US$MM)'], dtype='object')
Table 1 columns: Index(['Rank', 'Country', 'GDP(US$MM)'], dtype='object')
Table 2 columns: Index(['Rank', 'Country', 'GDP(US$MM)'], dtype='object')
DataFrames have been saved as CSV files.
IMF DataFrame:
  Rank        Country   GDP (IMF)
0  NaN      World[19]  79865481.0
1    1  United States  19390600.0
2    2     China[n 1]  12014610.0
3    3          Japan   4872135.0
4    4        Germany   3684816.0

World Bank DataFrame:
   Rank             Country  GDP (World Bank)
0   NaN               World      7.007807e+18
1   1.0       United States      7.007194e+18
2   NaN  European Union[23]      7.007173e+18
3   2.0          China[n 4]      7.007122e+18
4   3.0               Japan      7.006487e+18

United Nations DataFrame:
   Rank        Country      GDP (UN)
0   NaN      World[24]  7.007756e+18
1   1.0  United States  7.007186e+18
2   2.0     China[n 4]  7.007112e+18
3   3.0          Japan  7.006494e+18
4   4.0        Germany  7.006348e+18
```

1.4 *4. The Data Wrangling Workshop: Activity 6.01, page 309 -*

[53]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Read the CSV file
file_path = 'visit_data.csv'
df = pd.read_csv(file_path)

# Step 2: Check for duplicates
initial_size = df.shape[0]
df = df.drop_duplicates()
duplicates_removed = initial_size - df.shape[0]
print(f"Number of duplicates removed: {duplicates_removed}")

# Step 3: Check for NaN values
print("Missing values in each column:")
print(df.isna().sum())

# Step 4: Handle missing values
df = df.dropna(subset=['visit'])

# Step 4: Get rid of the outliers
# Assuming 'visit' is the column of interest for outliers
Q1 = df['visit'].quantile(0.25)
Q3 = df['visit'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers
df_clean = df[(df['visit'] >= lower_bound) & (df['visit'] <= upper_bound)]

# Step 5: Report the size difference
final_size = df_clean.shape[0]
size_difference = initial_size - final_size
print(f"After getting rid of outliers, the new size of the data is:␣
 ↪{final_size}")
print(f"Size difference after removing outliers: {size_difference}")

# Step 6: Create a box plot to check for outliers
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['visit'])
plt.title("Box Plot of Visits")
plt.xlabel("Number of Visits")
plt.show()

# Step 7: Get rid of any additional outliers if necessary
# This step is already covered in step 4

# Final DataFrame
print("Cleaned DataFrame:")
print(df_clean.head())
```

```
Number of duplicates removed: 0
Missing values in each column:
id              0
first_name    296
last_name     296
email           0
gender        505
ip_address      0
visit          26
dtype: int64
After getting rid of outliers, the new size of the data is: 974
Size difference after removing outliers: 26
```



```
Cleaned DataFrame:
   id first_name last_name                         email gender  \
0   1      Sonny      Dahl             sdahl0@mysql.com   Male
1   2        NaN       NaN            dhoovart1@hud.gov    NaN
2   3        Gar     Armal      garmal2@technorati.com    NaN
3   4    Chiarra     Nulty       cnulty3@newyorker.com    NaN
4   5        NaN       NaN  sleaver4@elegantthemes.com    NaN

        ip_address   visit
0    135.36.96.183  1225.0
1  237.165.194.143   919.0
2   166.43.137.224   271.0
3   139.98.137.108  1002.0
4    46.117.117.27  2434.0
```

[54]:
```python
import pandas as pd

# Load the cleaned CSV file
file_path = 'visit_data.csv'
df_clean = pd.read_csv(file_path)

# Display the cleaned DataFrame
print("Cleaned DataFrame:")
print(df_clean.head())

# Save the cleaned DataFrame to a new CSV file
df_clean.to_csv('cleaned_visit_data.csv', index=False)
print("Cleaned DataFrame has been saved to 'cleaned_visit_data.csv'.")
```

```
Cleaned DataFrame:
   id first_name last_name                         email gender  \
0   1      Sonny      Dahl             sdahl0@mysql.com   Male
1   2        NaN       NaN            dhoovart1@hud.gov    NaN
2   3        Gar     Armal      garmal2@technorati.com    NaN
3   4    Chiarra     Nulty       cnulty3@newyorker.com    NaN
4   5        NaN       NaN  sleaver4@elegantthemes.com    NaN

        ip_address   visit
0    135.36.96.183  1225.0
1  237.165.194.143   919.0
2   166.43.137.224   271.0
3   139.98.137.108  1002.0
4    46.117.117.27  2434.0
Cleaned DataFrame has been saved to 'cleaned_visit_data.csv'.
```

1.5 *5. Create a series and practice basic arithmetic steps -*

[55]:
```python
import pandas as pd

# Create Series 1
series1 = pd.Series([7.3, -2.5, 3.4, 1.5], index=['a', 'c', 'd', 'e'])

# Create Series 2
series2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index=['a', 'c', 'e', 'f', 'g'])

# Add Series 1 and Series 2 together
result_add = series1 + series2
print("Addition of Series 1 and Series 2:")
print(result_add)

# Subtract Series 1 from Series 2
result_subtract = series2 - series1
print("\nSubtraction of Series 1 from Series 2:")
print(result_subtract)
```

```
Addition of Series 1 and Series 2:
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64

Subtraction of Series 1 from Series 2:
a   -9.4
c    6.1
d    NaN
e   -3.0
f    NaN
g    NaN
dtype: float64
```

1.6 *6. Insert data into a SQL Lite database -*

[56]:
```python
import sqlite3
from tabulate import tabulate

# Connect to SQLite database (or create it if it doesn't exist)
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

# Drop the table if it exists to remove any existing data
```

```
cursor.execute('''DROP TABLE IF EXISTS contacts''')

# Create table
cursor.execute('''CREATE TABLE IF NOT EXISTS contacts
                  (name TEXT, address TEXT, city TEXT, state TEXT, zip TEXT,␣
 ↪phone_number TEXT)''')

# Insert data
data = [
    ('Jimmy Hayes', '123 Elm St', 'Knoxville', 'TN', '37931', '555-1234'),
    ('Tara Hayes', '456 Oak St', 'Knoxville', 'TN', '37931', '555-5678'),
    ('Graecyn Hayes', '789 Pine St', 'Knoxville', 'TN', '37931', '555-8765'),
    ('Tryston Hayes', '101 Maple St', 'Knoxville', 'TN', '37931', '555-4321'),
    ('Annie Hayes', '202 Cedar St', 'Knoxville', 'TN', '37931', '555-8765'),
    ('Anderson Hayes', '303 Birch St', 'Knoxville', 'TN', '37931', '555-3456'),
    ('Ralph Hilliard', '404 Cherry St', 'Gainesville', 'GA', '30506',␣
 ↪'555-6543'),
    ('Karen Hilliard', '505 Ash St', 'Gainesville', 'GA', '30506', '555-2345'),
    ('Britney Javens', '606 Walnut St', 'Gainesville', 'GA', '30506',␣
 ↪'555-7654'),
    ('Westyn Hilliard', '707 Poplar St', 'Knoxville', 'TN', '37931', '555-4567')
]

cursor.executemany('INSERT INTO contacts VALUES (?, ?, ?, ?, ?, ?)', data)

# Commit the transaction
conn.commit()

# Query the database to get the results
cursor.execute('SELECT * FROM contacts')
rows = cursor.fetchall()

# Print the results in a nicely formatted table
headers = ["Name", "Address", "City", "State", "Zip", "Phone Number"]
print(tabulate(rows, headers, tablefmt="fancy_grid"))

# Close the connection
conn.close()
```

```
 Name              Address         City         State      Zip  Phone Number



 Jimmy Hayes       123 Elm St      Knoxville    TN       37931  555-1234
```

```
Tara Hayes        456 Oak St     Knoxville     TN        37931   555-5678

Graecyn Hayes     789 Pine St    Knoxville     TN        37931   555-8765

Tryston Hayes     101 Maple St   Knoxville     TN        37931   555-4321

Annie Hayes       202 Cedar St   Knoxville     TN        37931   555-8765

Anderson Hayes    303 Birch St   Knoxville     TN        37931   555-3456

Ralph Hilliard    404 Cherry St  Gainesville   GA        30506   555-6543

Karen Hilliard    505 Ash St     Gainesville   GA        30506   555-2345

Britney Javens    606 Walnut St  Gainesville   GA        30506   555-7654

Westyn Hilliard   707 Poplar St  Knoxville     TN        37931   555-4567
```

```
[ ]:
```