Siddharth Srinivasan
Yuv Rout

CS 2316
Final Project

Purpose/Interest

- Interested in exploring the relationship between economic conditions and crime rates in Atlanta, with a specific focus on employment rates and labor force participation.
- Our purpose is to understand how fluctuations in the economy may influence criminal activity in the city.
- As students residing in Atlanta, we are concerned about the well-being and safety of our community, and we aim to uncover insights that could contribute to improving both economic conditions and public safety.
- By analyzing comprehensive datasets that include nearly every crime reported in Atlanta from 2009 to 2020, including statistics on employment and labor force participation, we can perform a thorough analysis of how these variables interact.

Expectations

Expected Findings

We expect to find a positive correlation between unemployment rates, decreased labor force participation, and increased crime rates in Atlanta.

Hypothesis

Hypothesis is that as people face financial difficulties due to unemployment or lack of participation in the labor force, they may be more inclined to engage in criminal activities.

Data Collection Process

CSV Dataset (Crime Data)

We searched for official crime statistics that would provide detailed information on criminal activity in Atlanta over an extended period. We found our CSV file on the Atlanta Police Department's open data portal. It includes nearly every reported crime in Atlanta from 2009 to 2020. The dataset provides crucial details such as report dates, occurrence dates, possible dates, crime types, and unique report numbers.

HTML Dataset (Labor Force Data)

We discovered the U.S. Bureau of Labor Statistics (BLS) website, which offers labor force participation rates over several years. The necessary data within the website contains the year, period, label, and value (participation rates). This dataset will help us understand the labor market trends that might influence crime rates.

API Dataset (Employment Data)

For unemployment rates specific to Atlanta, we sought datasets that provide localized economic data in a readable format. We found an API provided by Fulton County's open data portal. This dataset offers an extensive list of employment rates in Atlanta over time, which is essential for analyzing economic conditions in relation to crime rates.

Data Inconsistency - CSV

Inconsistency

Duplicate report numbers were discovered within the CSV file of the CrimeData dataset. Duplicate report numbers can indicate entry errors or repeated crime entries, which may skew analysis and reporting.

```python
import csv
import pandas as pd

def csv_data_parser():
    with open('2009_2020CrimeData.csv', 'r') as file:
        f = csv.reader(file)
        readerList = [line for line in f]

    readerList[0][0] = 'IDCol' #The IDCol Column heading has some invisible strings, making it hard to reference

    df = pd.DataFrame( readerList[1:], columns=readerList[0] )

    df = df.sort_values(by=['Report Number'])
    df = df.reset_index()


    dropIndexes = []
    for i in df.index[1:]:
        if df.loc[i, 'Report Number'] == df.loc[i - 1, 'Report Number']:
            dropIndexes.append(i)

    df = df.drop(dropIndexes, axis = 0)
    df = df.drop(['index'], axis=1)


    for i in df.index:
        if df.loc[i, 'Occur Date'] == 'NULL':
            df.loc[i, 'Occur Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Occur Time'] == 'NULL':
            df.loc[i, 'Occur Time'] = '12:00:00 AM'


        if df.loc[i, 'Possible Date'] == 'NULL':
            df.loc[i, 'Possible Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Possible Time'] == 'NULL':
            df.loc[i, 'Possible Time'] = '12:00:00 AM'

    df.to_csv('cleaned_atlantaCrime_csv_data')
    return df
```

Data Inconsistency - CSV

Inconsistency

Null values were found scattered across the CSV file in critical columns like Occur Date, Occur Time, Possible Date, and Possible Time. Null values represent missing information which can hinder accurate crime analysis, as these columns are crucial for understanding the timing of criminal events.

```python
import csv
import pandas as pd

def csv_data_parser():
    with open('2009_2020CrimeData.csv', 'r') as file:
        f = csv.reader(file)
        readerList = [line for line in f]

    readerList[0][0] = 'IDCol' #The IDCol Column heading has some invisible strings, making it hard to reference

    df = pd.DataFrame( readerList[1:], columns=readerList[0] )

    df = df.sort_values(by=['Report Number'])
    df = df.reset_index()


    dropIndexes = []
    for i in df.index[1:]:
        if df.loc[i, 'Report Number'] == df.loc[i - 1, 'Report Number']:
            dropIndexes.append(i)

    df = df.drop(dropIndexes, axis = 0)
    df = df.drop(['index'], axis=1)


    for i in df.index:
        if df.loc[i, 'Occur Date'] == 'NULL':
            df.loc[i, 'Occur Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Occur Time'] == 'NULL':
            df.loc[i, 'Occur Time'] = '12:00:00 AM'


        if df.loc[i, 'Possible Date'] == 'NULL':
            df.loc[i, 'Possible Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Possible Time'] == 'NULL':
            df.loc[i, 'Possible Time'] = '12:00:00 AM'

    df.to_csv('cleaned_atlantaCrime_csv_data')
    return df
```

Data Cleaning Process - CSV

Process

Step 1: Sort the dataset by the 'Report Number' column to prepare for iteration.

Step 2: Use a pandas function to iterate through the sorted list and check for consecutive entries with matching 'Report Number'.

Step 3: Remove all entries from the dataset where 'Report Number' duplicates are found to ensure each report is uniquely represented.

```
import csv
import pandas as pd

def csv_data_parser():
    with open('2009_2020CrimeData.csv', 'r') as file:
        f = csv.reader(file)
        readerList = [line for line in f]

    readerList[0][0] = 'IDCol' #The IDCol Column heading has some invisible strings, making it hard to reference

    df = pd.DataFrame( readerList[1:], columns=readerList[0] )

    df = df.sort_values(by=['Report Number'])
    df = df.reset_index()


    dropIndexes = []
    for i in df.index[1:]:
        if df.loc[i, 'Report Number'] == df.loc[i - 1, 'Report Number']:
            dropIndexes.append(i)

    df = df.drop(dropIndexes, axis = 0)
    df = df.drop(['index'], axis=1)


    for i in df.index:
        if df.loc[i, 'Occur Date'] == 'NULL':
            df.loc[i, 'Occur Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Occur Time'] == 'NULL':
            df.loc[i, 'Occur Time'] = '12:00:00 AM'


        if df.loc[i, 'Possible Date'] == 'NULL':
            df.loc[i, 'Possible Date'] = df.loc[i, 'Report Date'].split()[0]

        if df.loc[i, 'Possible Time'] == 'NULL':
            df.loc[i, 'Possible Time'] = '12:00:00 AM'

    df.to_csv('cleaned_atlantaCrime_csv_data')
    return df
```

Data Inconsistency - HTML

Inconsistency

The HTML dataset from the BLS Data Viewer website presented an inconsistency in date formats compared to other data sources. This discrepancy is significant as it prevents seamless integration and comparative analysis with other datasets.

```
def web_parser2():
  from selenium import webdriver
  from selenium.webdriver.chrome.service import Service
  from selenium.webdriver.common.by import By
  from webdriver_manager.chrome import ChromeDriverManager
  from selenium.webdriver.support.ui import Select

  import time

  from bs4 import BeautifulSoup
  import pandas as pd

  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

  driver.get('https://data.bls.gov/dataViewer/view/timeseries/LNS11300000')
  start_year_dropdown = driver.find_element(By.TAG_NAME, 'select')
  select = Select( start_year_dropdown )
  select.select_by_visible_text('2008')

  update_button = driver.find_element(By.XPATH, '//*[@id="dv-submit"]')
  update_button.click()

  time.sleep(2)
```

Data Cleaning Process - HTML

Process

Step 1: Identify the differing date format within the HTML dataset where dates are presented in a month-year format using words (e.g., 2008 Apr).

Step 2: Convert the word-based date format to a numeric format (year-month-day), making the assumption that data values are reported on the first of each month for consistency.

Step 3: Use concatenation to combine the year-month string with a day value ('01') to standardize dates across datasets.

```
def web_parser2():
  from selenium import webdriver
  from selenium.webdriver.chrome.service import Service
  from selenium.webdriver.common.by import By
  from webdriver_manager.chrome import ChromeDriverManager
  from selenium.webdriver.support.ui import Select

  import time

  from bs4 import BeautifulSoup
  import pandas as pd

  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

  driver.get('https://data.bls.gov/dataViewer/view/timeseries/LNS11300000')
  start_year_dropdown = driver.find_element(By.TAG_NAME, 'select')
  select = Select( start_year_dropdown )
  select.select_by_visible_text('2008')

  update_button = driver.find_element(By.XPATH, '//*[@id="dv-submit"]')
  update_button.click()

  time.sleep(2)
```

```
html = driver.page_source


soup = BeautifulSoup(html , 'html.parser')
data_value = soup.find('div', {'id' : 'bodytext'} ).find('div', {'id' : 'tablediv1'} ).find('tbody').find_all('td', {'class' : ""})
data_time = soup.find('div', {'id' : 'bodytext'} ).find('div', {'id' : 'tablediv1'} ).find('tbody').find_all('td', {'class' : "abbrlabel"})
data_period = soup.find('div', {'id' : 'bodytext'} ).find('div', {'id' : 'tablediv1'} ).find('tbody').find_all('td', {'class' : "period"})


data_dictionary={}

value_list = []
time_list = []
period_list = []

for i in data_value:
    value_list.append(i.text.strip() )

for j in data_time:
    time_list.append(j.text.strip() )

for k in data_period:
    period_list.append(k.text.strip() )

for item in range( len(value_list) ):
    data_dictionary[item] = [value_list[item], time_list[item], period_list[item]]

for key in data_dictionary:
    data_dictionary[key][1] = f'{data_dictionary[key][1].split()[0]}-{data_dictionary[key][2][1:]}-01'

df = pd.DataFrame(data_dictionary, index=['values', 'date', 'period']).T

df.to_csv('cleaned_webscrapped_data.csv')

return df
```

Data Cleaning Process - API

Process

Step 1: Use the Requests module to get a JSON response from the API.

Step 2: Parse through the data to ensure certain columns have the correct data type.

Step 3: Clean out data that isn't important to the dataset, and isn't in the correct timeframe.

```
import requests
import pandas as pd

def api_web_parser1():

    URL = 'https://sharefulton.fultoncountyga.gov/resource/5rm7-xueu.json?area_code='
    AREA_CODE = 'CT1304000000000'
    OFFSET = '&$offset=1099'

    response = requests.get(URL+AREA_CODE+OFFSET)

    df = pd.DataFrame(response.json() )

    df['value'] = df['value'].astype(float)
    df['year'] = df['year'].astype(int)
    df = df.sort_values(by='month2')

    droplist = []
    for i in df.index:
        if df.loc[i,'year'] < 2009:
            droplist.append(i)
        if df.loc[i,'series_id'] != 'LAUCT130400000000005':
            droplist.append(i)

    df = df.drop(droplist, axis=0)

    df.to_csv('cleaned_unemployment_api_data.csv')
    return df
```

Insight 1

```python
def insight1():

  import pandas as pd

  from phase2_data_cleaning.api_data_collect import api_web_parser

  dfApi = api_web_parser()
  dfApi = dfApi.reset_index()
  dfApi = dfApi.drop(['index', 'footnote_codes', 'id', 'year', 'period', 'area_code', 'area_text', 'series_id', 'month'], axis=1)
  dfApi.columns = ['values', 'series_title', 'dates']



  dfWeb = pd.read_csv('phase2_data_cleaning/cleaned_webscrapped_data.csv')
  dfWeb['series_title'] = pd.Series(index=dfWeb.index).fillna('(Seas) Labor Force Participation Rate')
  dfWeb['dates'] = dfWeb['date']
  dfWeb = dfWeb.drop(['Unnamed: 0', 'period', 'date'], axis=1)

  dfApi.index = dfApi['dates']

  dfWeb.index = dfWeb['dates']
  return (dfApi['values'].corr(dfWeb['values'], method='spearman'))



############ Function Call ############
insight1()
```

```
[58]:  -0.7233615194836591
```

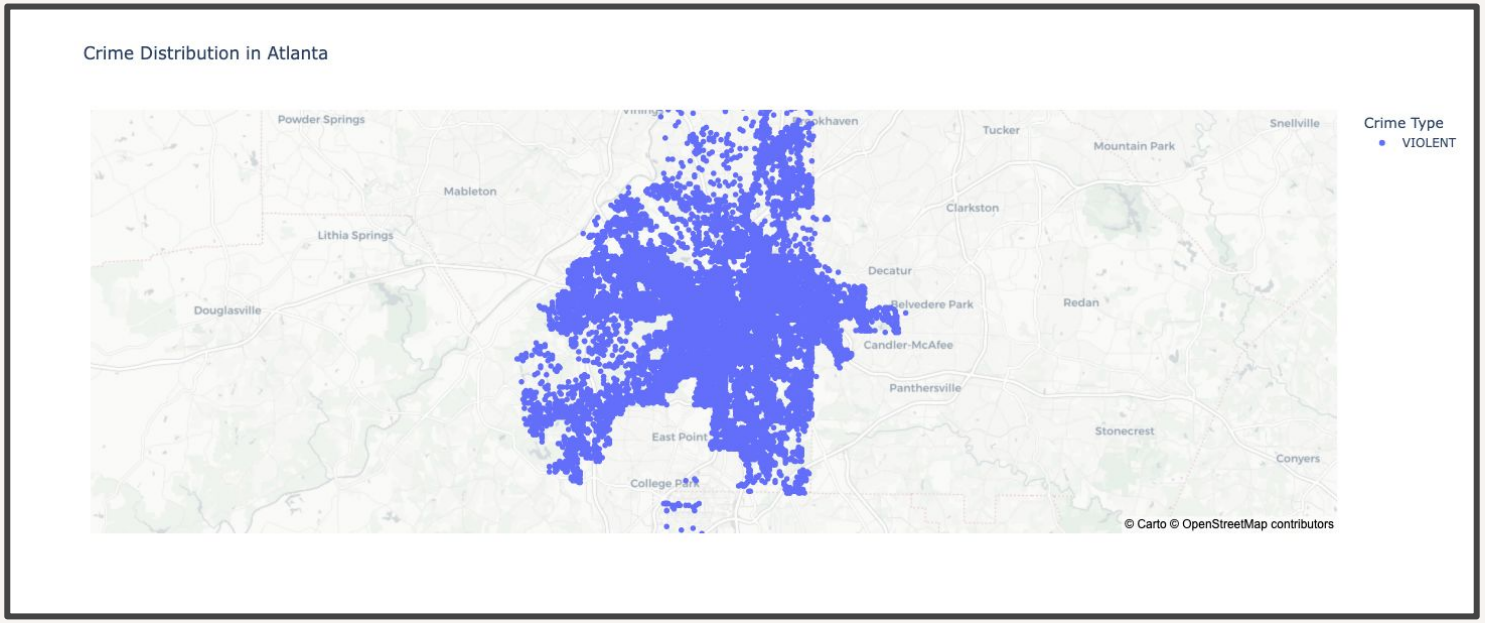
Visualization 1

Insight 2

```
import pandas as pd

def year_format(x):
    try:
        l = x.split('/')
        if len(l[1]) == 1:
            l[1] = '0' + l[1]
        if len(l[0]) == 1:
            l[0] = '0' + l[0]

        o = [l[2], l[0], l[1]]
        return ('-'.join(o) )
    except:
        a = x
def crime_type(x):
        if x in ['HOMICIDE', 'ROBBERY', 'AGG ASSAULT']:
            return 'VIOLENT'
        else:
            return 'NON-VIOLENT'

dfCrime = pd.read_csv('phase2_data_cleaning/cleaned_atlantaCrime_csv_data.csv')
dfCrime = dfCrime.drop(['Unnamed: 0'], axis=1)

dfCrime['Occur Date'] = pd.Series(dfCrime['Occur Date']).apply( lambda x : year_format(x) )
dfCrime['Crime Type'] = pd.Series(dfCrime['Crime Type']).apply( lambda x : crime_type(x) )

dfCrime['Latitude'] = dfCrime['Latitude'].round(3)
dfCrime['Longitude'] = dfCrime['Longitude'].round(3)

droplist = []
for i in dfCrime.index:
    if dfCrime.loc[i,'Occur Date'] is None:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) < 2009:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) > 2024:
        droplist.append(i)
    elif  dfCrime.loc[i,'Crime Type'] == 'NON-VIOLENT':
        droplist.append(i)

dfCrime = dfCrime.drop(droplist, axis=0)

crime_by_neighborhood = dfCrime.groupby("Neighborhood").size().reset_index(name="Crime Count").sort_values(by='Crime Count', ascending=False

return (crime_by_neighborhood)
```

	Neighborhood	Crime Count
79	Downtown	3383
147	Midtown	1564
221	West End	1530
163	Old Fourth Ward	1441
108	Grove Park	1220
...	...	...
118	Horseshoe Community	1
153	Mt. Paran Parkway	1
162	Old Fairburn Village	1
187	Regency Trace	1
87	Englewood Manor	1

Visualization 2



Insight 3

```
import pandas as pd
import numpy as np

def year_format(x):
    try:
        l = x.split('/')
        if len(l[1]) == 1:
            l[1] = '0' + l[1]
        if len(l[0]) == 1:
            l[0] = '0' + l[0]

        o = [l[2], l[0], l[1]]
        return ('-'.join(o) )
    except:
        a = x

dfCrime = pd.read_csv('phase2_data_cleaning/cleaned_atlantaCrime_csv_data.csv')
dfCrime = dfCrime.drop(['Unnamed: 0'], axis=1)

dfCrime['Occur Date'] = pd.Series(dfCrime['Occur Date']).apply( lambda x : year_format(x) )


droplist = []
for i in dfCrime.index:
    if dfCrime.loc[i,'Occur Date'] is None:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) < 2009:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) > 2024:
        droplist.append(i)

dfCrime = dfCrime.drop(droplist, axis=0)

stats1 = dfCrime.groupby(["Occur Date"])["Report Number"].count().reset_index()
stats1.columns = ["Occur Date", "Number of Crimes Committed"]
stats1['Occur Date'] = np.array(stats1['Occur Date'], dtype='datetime64[D]')
stats1['Year'] = pd.Series(stats1['Occur Date']).apply( lambda x : x.year )

final = stats1.groupby(['Year'])['Number of Crimes Committed'].mean().reset_index()
return ( final)
```

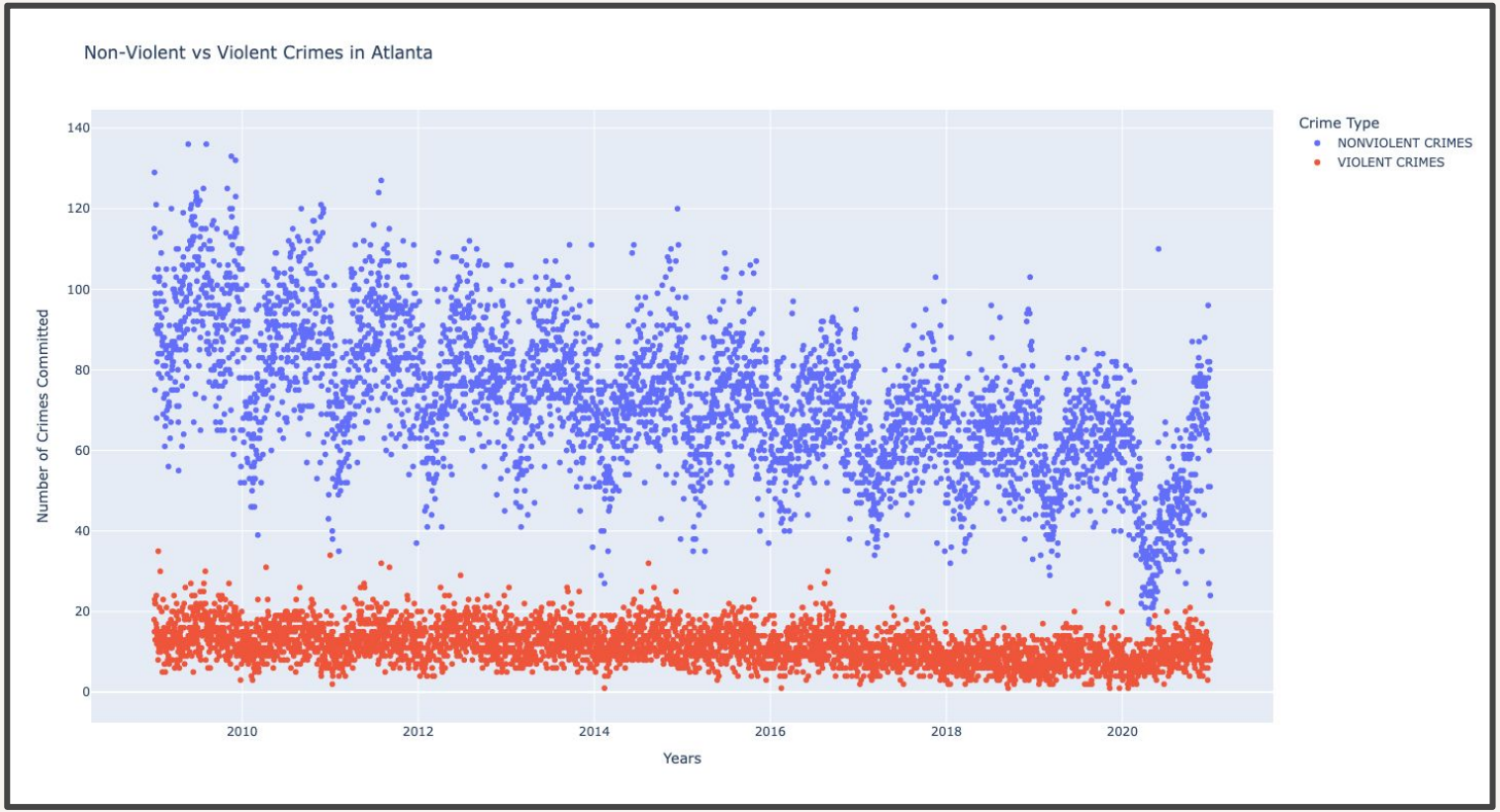
	Year	Number of Crimes Committed
0	2009	108.213699
1	2010	97.936986
2	2011	96.106849
3	2012	91.994536
4	2013	89.336986
5	2014	85.827397
6	2015	82.857534
7	2016	79.551913
8	2017	72.723288
9	2018	70.443836
10	2019	68.542466
11	2020	59.691257

Insight 4

```
import pandas as pd
import numpy as np


def year_format(x):
    try:
        l = x.split('/')
        if len(l[1]) == 1:
            l[1] = '0' + l[1]
        if len(l[0]) == 1:
            l[0] = '0' + l[0]

        o = [l[2], l[0], l[1]]
        return ('-'.join(o) )
    except:
        a = x

def crime_type(x):
        if x in ['HOMICIDE', 'ROBBERY', 'AGG ASSAULT']:
            return 'VIOLENT CRIMES'
        else:
            return 'NONVIOLENT CRIMES'

dfCrime = pd.read_csv('phase2_data_cleaning/cleaned_atlantaCrime_csv_data.csv')
dfCrime = dfCrime.drop(['Unnamed: 0'], axis=1)

dfCrime['Occur Date'] = pd.Series(dfCrime['Occur Date']).apply( lambda x : year_format(x) )
dfCrime['Crime Type'] = pd.Series(dfCrime['Crime Type']).apply( lambda x : crime_type(x) )

droplist = []
for i in dfCrime.index:
    if dfCrime.loc[i,'Occur Date'] is None:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) < 2009:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) > 2024:
        droplist.append(i)


dfCrime = dfCrime.drop(droplist, axis=0)


stats = dfCrime.groupby(["Occur Date", "Crime Type"])["Report Number"].count().reset_index()
stats.columns = ["Occur Date", "Crime Type", "Number of Crimes Committed"]
stats['Occur Date'] = np.array(stats['Occur Date'], dtype='datetime64[D]')

stats['Occur Date'] = np.array(stats['Occur Date'], dtype='datetime64[D]')
stats['Year'] = pd.Series(stats['Occur Date']).apply( lambda x : x.year )

final = stats.groupby(['Crime Type', 'Year'])['Number of Crimes Committed'].mean().reset_index()

return (final)
```

0	NONVIOLENT CRIMES	2009	93.583562
1	NONVIOLENT CRIMES	2010	84.706849
2	NONVIOLENT CRIMES	2011	82.649315
3	NONVIOLENT CRIMES	2012	78.707650
4	NONVIOLENT CRIMES	2013	76.438356
5	NONVIOLENT CRIMES	2014	72.997260
6	NONVIOLENT CRIMES	2015	70.854795
7	NONVIOLENT CRIMES	2016	68.068306
8	NONVIOLENT CRIMES	2017	63.093151
9	NONVIOLENT CRIMES	2018	62.353425
10	NONVIOLENT CRIMES	2019	59.939726
11	NONVIOLENT CRIMES	2020	50.475410
12	VIOLENT CRIMES	2009	14.630137
13	VIOLENT CRIMES	2010	13.230137
14	VIOLENT CRIMES	2011	13.457534
15	VIOLENT CRIMES	2012	13.286885
16	VIOLENT CRIMES	2013	12.898630
17	VIOLENT CRIMES	2014	12.830137
18	VIOLENT CRIMES	2015	12.002740
19	VIOLENT CRIMES	2016	11.483607
20	VIOLENT CRIMES	2017	9.630137
21	VIOLENT CRIMES	2018	8.090411
22	VIOLENT CRIMES	2019	8.602740
23	VIOLENT CRIMES	2020	9.241096

Visualization 3



Insight 5

```
import pandas as pd
import numpy as np

def year_format(x):
    try:
        l = x.split('/')
        if len(l[1]) == 1:
            l[1] = '0' + l[1]
        if len(l[0]) == 1:
            l[0] = '0' + l[0]

        o = [l[2], l[0], l[1]]
        return ('-'.join(o) )
    except:
        a = x

def crime_type(x):
        if x in ['HOMICIDE', 'ROBBERY', 'AGG ASSAULT']:
            return 'VIOLENT CRIMES'
        else:
            return 'NONVIOLENT CRIMES'

dfCrime = pd.read_csv('phase2_data_cleaning/cleaned_atlantaCrime_csv_data.csv')
dfCrime = dfCrime.drop(['Unnamed: 0'], axis=1)

dfCrime['Occur Date'] = pd.Series(dfCrime['Occur Date']).apply( lambda x : year_format(x) )
dfCrime['Crime Type'] = pd.Series(dfCrime['Crime Type']).apply( lambda x : crime_type(x) )

droplist = []
for i in dfCrime.index:
    if dfCrime.loc[i,'Occur Date'] is None:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) < 2009:
        droplist.append(i)
    elif int( dfCrime.loc[i,'Occur Date'].split('-')[0] ) > 2024:
        droplist.append(i)
    elif dfCrime.loc[i,'Crime Type'] == 'NONVIOLENT CRIMES':
        droplist.append(i)

dfCrime = dfCrime.drop(droplist, axis=0)
stats = dfCrime.groupby(["Occur Date", "Crime Type"])["Report Number"].count().reset_index()
stats.columns = ["Occur Date", "Crime Type", "Number of Crimes Committed"]
stats['Occur Date'] = np.array(stats['Occur Date'], dtype='datetime64[D]')

stats['Occur Date'] = np.array(stats['Occur Date'], dtype='datetime64[D]')
stats['Year'] = pd.Series(stats['Occur Date']).apply( lambda x : x.year )

crime_by_year = stats.groupby(['Year'])['Number of Crimes Committed'].mean().reset_index()

dfApi = pd.read_csv('phase2_data_cleaning/cleaned_unemployment_api_data.csv')
dfApi = dfApi.reset_index()
dfApi = dfApi.drop(['Unnamed: 0', 'index', 'footnote_codes', 'id', 'year', 'period', 'area_code', 'area_text', 'series_id', 'month'], axis
dfApi.columns = ['values', 'series_title', 'dates']
dfApi['dates'] = np.array(dfApi['dates'], dtype='datetime64[D]')
dfApi['Year'] = pd.Series(dfApi['dates']).apply( lambda x : x.year )

employ_by_year = dfApi.groupby(['Year'])['values'].mean().reset_index()
crime_by_year.index = crime_by_year['Year']
employ_by_year.index = employ_by_year['Year']
return (crime_by_year['Number of Crimes Committed'].corr(employ_by_year['values'], method='spearman'))
```

```
-0.9230769230769231
```

Overall Results & Conclusion

Challenges

We faced a significant challenge when trying to successfully collect data from the HTML dataset. The website required the use of Selenium to manipulate date inputs for data extraction. This added complexity as we had to use trial and error to effectively integrate Selenium into our script, ensuring accurate and efficient data collection for the specified dates.

Results

Higher crime density in more populated areas

Atlanta's employment rate significantly affects its crime rate

Inverse relationship between US Labor Force Participation and Atlanta employment

Learn

Was able to explore different plotly graphs such as choropleth map

Extend my Pandas knowledge (.merge(), groupby, Dataframe.corr)

Practical experience with all steps of data cleaning, data processing, and data analysis.