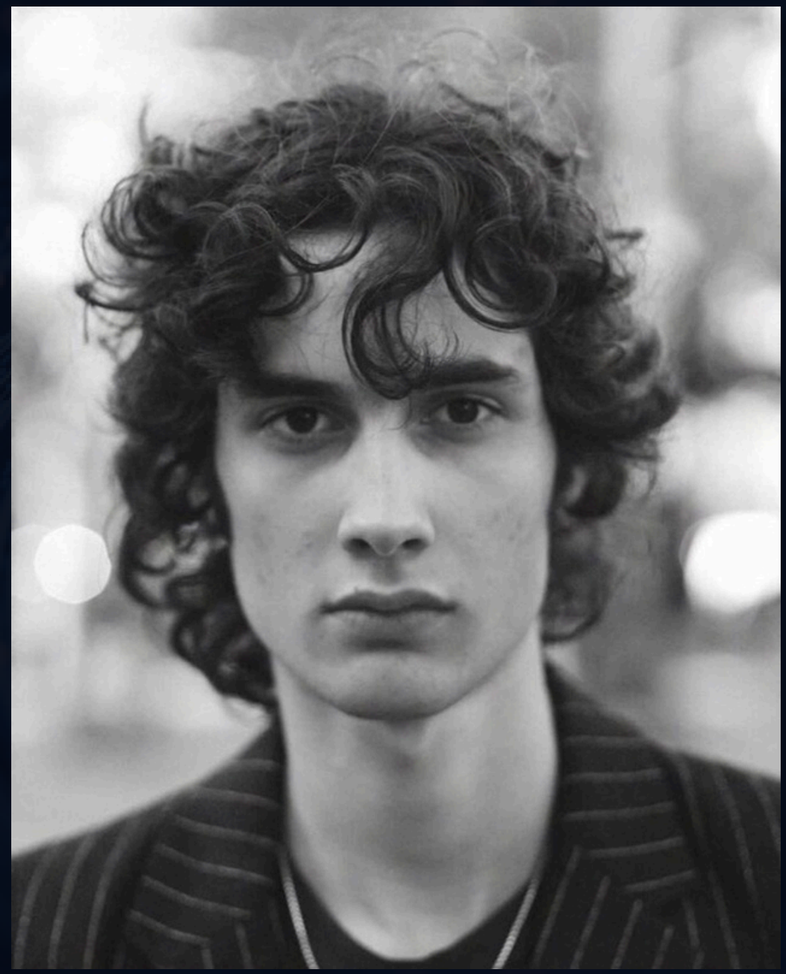
A DATA ANALYSIS OF FBI WANTED INDIVIDUALS USING HYPOTHESIS TESTING AND TEXT ANALYSIS

THIS STUDY INVOLVED RETRIEVING DATA FROM THE FBI WANTED API, CLEANING AND ANALYZING THE DATASET, PERFORMING TEXT AND N-GRAM ANALYSIS, AND APPLYING STATISTICAL TESTING TO EVALUATE AGE-BASED TRENDS IN CRIMES.

IRFAN FAISAL - 23224532
LABIB MASHFIQ RAHMAN - 23229564
HOSSAIN MUHTASIM TAHMID - 23229550
ABRER RAHAT HOSSAIN - 23229502



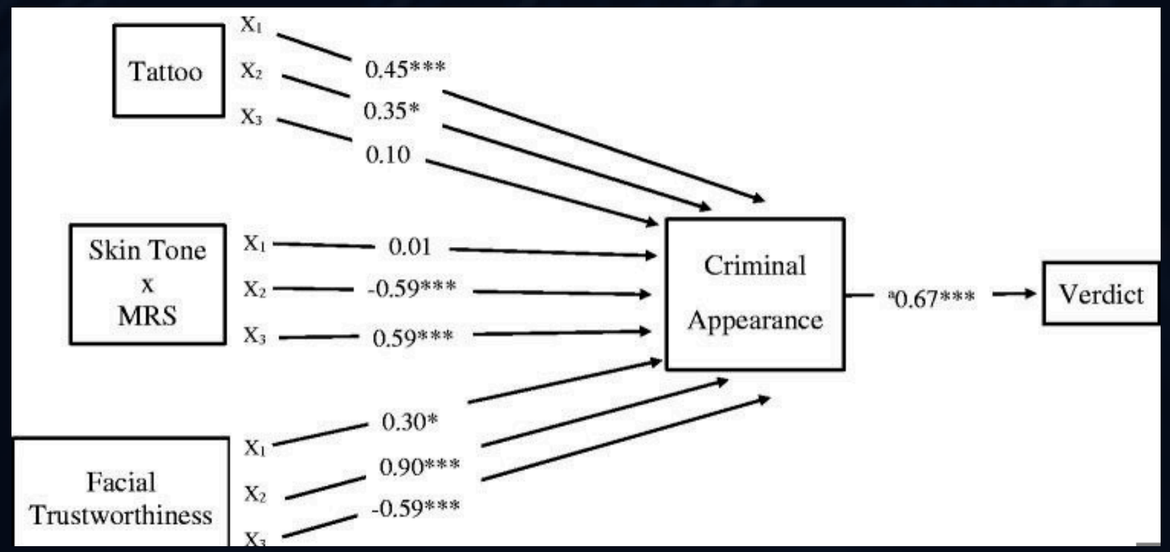
Richard Ramirez

Known as the “Night Stalker,” he was an American serial killer who terrorized California between 1984 and 1985. Ramirez was captured in August 1985, convicted of 13 murders and numerous other crimes, and sentenced to death. He died in 2013 on death row from cancer at age 53.

Known for the pentagon tattoo on his forearm

You are not alone

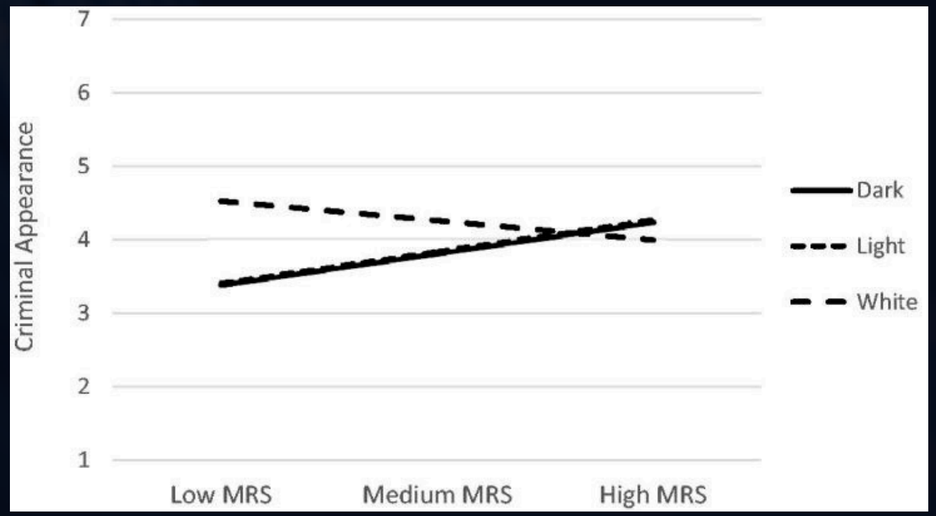
This study by Mariah Sorby and Andre Kehn (University of North Dakota, 2020) found that physical traits like tattoos, darker skin, and untrustworthy faces influenced mock jurors' guilty verdicts by making defendants appear more "criminal." The effect was stronger among jurors with higher racial prejudice, showing that personal biases amplify how appearance affects guilt decisions.



What Made People Look More Criminal:

1. **TATTOOS ✔✔✔ (p < .001)**
   - Non-aggressive tattoo vs. none: b = 0.44 (significant)
   - Aggressive tattoo vs. none: b = 0.35 (significant)
   - Effect: Having ANY tattoo increased criminal appearance ratings

2. **SKIN TONE ✔✔✔ (p < .001)**
   - White vs. Dark skin: b = 1.84 (VERY significant)
   - Light vs. Dark skin: b = 0.07 (not significant)
   - Effect: Darker skin strongly associated with criminal appearance



HYPOTHESIS 1

Research Question: Are individuals wanted for certain types of crimes (e.g., violent crimes vs. white-collar crimes) more likely to have tattoos than individuals wanted for other types of crimes within this dataset?

Null Hypothesis (H0): There is no statistically significant difference in the proportion of individuals with tattoos between those wanted for violent crimes and those wanted for white-collar crimes in this dataset.

Alternative Hypothesis (H1): There is a statistically significant difference in the proportion of individuals with tattoos between those wanted for violent crimes and those wanted for white-collar crimes in this dataset.

DATA ACQUISITION FROM FBI API

HOW WE COLLECTED THE DATA

Used the FBI's public API:
https://api.fbi.gov/wanted/v1/list

We used Python's requests library to send a GET request to the FBI's public Wanted Persons API.
The goal was to retrieve 500 records in a single request by setting the pageSize to 500.
The code also included error handling using raise_for_status() to ensure the request succeeded before parsing the JSON response.

```python
import requests

api_url = "https://api.fbi.gov/wanted/v1/list"
params = {'pageSize': 500} # Set pageSize parameter to 500

try:
    response = requests.get(api_url, params=params)
    response.raise_for_status()  # Raise an exception for bad status codes (4xx or 5xx)
    data = response.json()
    print(f"Number of items retrieved: {len(data.get('items', []))}")
    # print(data) # Uncomment to see the full data

except requests.exceptions.RequestException as e:
    print(f"Error fetching data: {e}")
```

```
Number of items retrieved: 50
```

CHALLENGES AND SOLUTIONS

CHALLENGES IN DATA COLLECTION

API limitation: max 50 records per request

High percentage of **missing values** in several columns

Solution: Paginated requests, dropped sparse columns, and cleaned data

```python
import requests
import pandas as pd

api_url = "https://api.fbi.gov/wanted/v1/list"
all_items = []
pageSize = 50  # Keep the page size to 50 as before
current_page = 1

print("Attempting to fetch all entries using pagination...")

while True: # Loop indefinitely until explicitly broken
    params = {'pageSize': pageSize, 'page': current_page}
    try:
        response = requests.get(api_url, params=params)
        response.raise_for_status()  # Raise an exception for bad status codes (4xx or 5xx)
        data = response.json()
        items = data.get('items', [])

        if not items:
            print("No more items to retrieve.")
            break

        all_items.extend(items)
        print(f"Fetched {len(items)} items from page {current_page}. Total items collected: {len(all_items)}")

        current_page += 1

    except requests.exceptions.RequestException as e:
        print(f"Error fetching data: {e}")
        break

if all_items:
    df = pd.DataFrame(all_items) #in a pandas dataframe
    display(df.head())
    print(f"DataFrame created with {len(df)} entries.")
else:
    print("No data retrieved to create DataFrame.")
```

Retrieve 1,000 entries from the FBI Wanted Persons database.

- Our initial attempt was to request 1,000 entries at once using pageSize=1000, but the API still returned only 50 records.
- This revealed a built-in limit of 50 results per request, regardless of the parameter used.

We implemented pagination, requesting data page-by-page, collecting 50 records per page.

- By looping through 20+ pages, we successfully gathered 1,000 total records.

```
if all_items:
    df = pd.DataFrame(all_items) #in a pandas dataframe
    display(df.head())
    print(f"DataFrame created with {len(df)} entries.")
else:
    print("No data retrieved to create DataFrame.")
```

```
Attempting to fetch all entries using pagination...
Fetched 50 items from page 1. Total items collected: 50
Fetched 50 items from page 2. Total items collected: 100
Fetched 50 items from page 3. Total items collected: 150
Fetched 50 items from page 4. Total items collected: 200
Fetched 50 items from page 5. Total items collected: 250
Fetched 50 items from page 6. Total items collected: 300
Fetched 50 items from page 7. Total items collected: 350
Fetched 50 items from page 8. Total items collected: 400
Fetched 50 items from page 9. Total items collected: 450
Fetched 50 items from page 10. Total items collected: 500
Fetched 50 items from page 11. Total items collected: 550
Fetched 50 items from page 12. Total items collected: 600
Fetched 50 items from page 13. Total items collected: 650
Fetched 50 items from page 14. Total items collected: 700
Fetched 50 items from page 15. Total items collected: 750
Fetched 50 items from page 16. Total items collected: 800
Fetched 50 items from page 17. Total items collected: 850
Fetched 50 items from page 18. Total items collected: 900
Fetched 50 items from page 19. Total items collected: 950
Fetched 50 items from page 20. Total items collected: 1000
Fetched 50 items from page 21. Total items collected: 1050
Fetched 10 items from page 22. Total items collected: 1060
No more items to retrieve.
```

DATA CLEANING PROCESS
CLEANING THE DATASET

- **Dropped columns** with more than **80%** missing values

```
# Dropping columns with more than 80% missing values
missing_percentage = df.isnull().sum() / len(df) * 100
columns_to_drop = missing_percentage[missing_percentage > 80].index
df = df.drop(columns=columns_to_drop)
```

- **Removed irrelevant fields:** files, reward, path, weight, coordinates

- **Focused on core fields:** description, title, hair, eyes, sex, subjects, age_min, age_max

EXPLORATORY DATA ANALYSIS(EDA)

Inititially fetched data

```
Data columns (total 54 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   possible_states         68 non-null     object
 1   warning_message         165 non-null    object
 2   field_offices           409 non-null    object
 3   details                 325 non-null    object
 4   locations               0 non-null      object
 5   age_range               150 non-null    object
 6   path                    600 non-null    object
 7   occupations             112 non-null    object
 8   eyes_raw                429 non-null    object
 9   scars_and_marks         177 non-null    object
 10  weight                  380 non-null    object
 11  poster_classification   600 non-null    object
 12  possible_countries      51 non-null     object
 13  eyes                    422 non-null    object
 14  files                   600 non-null    object
 15  modified                600 non-null    object
 16  age_min                 146 non-null    float64
 17  caution                 277 non-null    object
 18  description             600 non-null    object
 19  person_classification   600 non-null    object
 20  hair                    450 non-null    object
 21  reward_max              600 non-null    int64
 22  title                   600 non-null    object
 23  coordinates             600 non-null    object
 24  place_of_birth          262 non-null    object
 25  languages               88 non-null     object
 26  race_raw                431 non-null    object
 27  reward_min              600 non-null    int64
 28  complexion              26 non-null     object
 29  aliases                 250 non-null    object
 30  url                     600 non-null    object
 31  ncic                    137 non-null    object
 32  height_min              383 non-null    float64
 33  race                    431 non-null    object
 34  publication             600 non-null    object
 35  weight_max              373 non-null    float64
 36  subjects                600 non-null    object
 37  images                  600 non-null    object
 38  suspects                3 non-null      object
 39  remarks                 337 non-null    object
 40  reward_text             219 non-null    object
 41  nationality             251 non-null    object
 42  legat_names             3 non-null      object
 43  dates_of_birth_used     325 non-null    object
 44  status                  600 non-null    object
 45  build                   46 non-null     object
 46  weight_min              373 non-null    float64
 47  hair_raw                464 non-null    object
 48  uid                     600 non-null    object
 49  sex                     507 non-null    object
 50  height_max              383 non-null    float64
 51  additional_information  10 non-null     object
 52  age_max                 146 non-null    float64
 53  pathId                  600 non-null    object
```

Percentage values

```
#shows how much data is missing in each column
missing_percentage = df.isnull().sum() / len(df) * 100
print(missing_percentage)
```

```
possible_states           88.666667
warning_message           72.500000
field_offices             31.833333
details                   45.833333
locations                100.000000
age_range                 75.000000
path                       0.000000
occupations               81.333333
eyes_raw                  28.500000
scars_and_marks           70.500000
weight                    36.666667
poster_classification      0.000000
possible_countries        91.500000
eyes                      29.666667
files                      0.000000
modified                   0.000000
age_min                   75.666667
caution                   53.833333
description                0.000000
person_classification      0.000000
hair                      25.000000
reward_max                 0.000000
title                      0.000000
coordinates                0.000000
place_of_birth            56.333333
languages                 85.333333
race_raw                  28.166667
reward_min                 0.000000
complexion                95.666667
aliases                   58.333333
url                        0.000000
ncic                      77.166667
height_min                36.166667
race                      28.166667
publication                0.000000
weight_max                37.833333
subjects                   0.000000
images                     0.000000
suspects                  99.500000
remarks                   43.833333
reward_text               63.500000
nationality               58.166667
legat_names               99.500000
dates_of_birth_used       45.833333
status                     0.000000
build                     92.333333
weight_min                37.833333
hair_raw                  22.666667
uid                        0.000000
sex                       15.500000
height_max                36.166667
additional_information    98.333333
age_max                   75.666667
pathId                     0.000000
dtype: float64
```

EXPLORATORY DATA ANALYSIS(EDA)

Dropping columns with more than 80% missing values

```
DataFrame info after dropping columns with > 80% missing values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 44 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   warning_message       165 non-null    object
 1   field_offices         409 non-null    object
 2   details               325 non-null    object
 3   age_range             150 non-null    object
 4   path                  600 non-null    object
 5   eyes_raw              429 non-null    object
 6   scars_and_marks       177 non-null    object
 7   weight                380 non-null    object
 8   poster_classification 600 non-null    object
 9   eyes                  422 non-null    object
 10  files                 600 non-null    object
 11  modified              600 non-null    object
 12  age_min               146 non-null    float64
 13  caution               277 non-null    object
 14  description           600 non-null    object
 15  person_classification 600 non-null    object
 16  hair                  450 non-null    object
 17  reward_max            600 non-null    int64
 18  title                 600 non-null    object
 19  coordinates           600 non-null    object
 20  place_of_birth        262 non-null    object
 21  race_raw              431 non-null    object
 22  reward_min            600 non-null    int64
 23  aliases               250 non-null    object
 24  url                   600 non-null    object
 25  ncic                  137 non-null    object
 26  height_min            383 non-null    float64
 27  race                  431 non-null    object
 28  publication           600 non-null    object
 29  weight_max            373 non-null    float64
 30  subjects              600 non-null    object
 31  images                600 non-null    object
 32  remarks               337 non-null    object
 33  reward_text           219 non-null    object
 34  nationality           251 non-null    object
 35  dates_of_birth_used   325 non-null    object
 36  status                600 non-null    object
 37  weight_min            373 non-null    float64
 38  hair_raw              464 non-null    object
 39  uid                   600 non-null    object
 40  sex                   507 non-null    object
 41  height_max            383 non-null    float64
 42  age_max               146 non-null    float64
 43  pathId                600 non-null    object
dtypes: float64(6), int64(2), object(36)
memory usage: 206.4+ KB
None
```

Data Cleaning

```
#dropping the columns that are not important to our analysis

columns_to_drop=['files','reward_max','reward_min','path','coordi
# Keep 'age_min' and 'age_max' for age-based analysis
df = df.drop(columns=columns_to_drop, errors='ignore') # Use erro

display(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 32 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   warning_message       165 non-null    object
 1   field_offices         409 non-null    object
 2   details               325 non-null    object
 3   eyes_raw              429 non-null    object
 4   scars_and_marks       177 non-null    object
 5   weight                380 non-null    object
 6   poster_classification 600 non-null    object
 7   eyes                  422 non-null    object
 8   modified              600 non-null    object
 9   age_min               146 non-null    float64
 10  caution               277 non-null    object
 11  description           600 non-null    object
 12  person_classification 600 non-null    object
 13  hair                  450 non-null    object
 14  title                 600 non-null    object
 15  place_of_birth        262 non-null    object
 16  race_raw              431 non-null    object
 17  ncic                  137 non-null    object
 18  height_min            383 non-null    float64
 19  race                  431 non-null    object
 20  publication           600 non-null    object
 21  subjects              600 non-null    object
 22  images                600 non-null    object
 23  remarks               337 non-null    object
 24  reward_text           219 non-null    object
 25  nationality           251 non-null    object
 26  dates_of_birth_used   325 non-null    object
 27  status                600 non-null    object
 28  hair_raw              464 non-null    object
 29  sex                   507 non-null    object
 30  height_max            383 non-null    float64
 31  age_max               146 non-null    float64
dtypes: float64(4), object(28)
memory usage: 150.1+ KB
None
```

Dropping the columns that are not important to our analysis

STEP: N GRAM ANALYSIS

By n gram analysis, it finds the most frequent two-word phrases (bigrams) and three-word phrases (trigrams) to see what kinds of combinations of words appear most often.

```python
def get_top_n_ngrams(corpus, n=None, ngram_range=(1, 1)):
    """
    Calculates the frequency of n-grams in a corpus and returns the top n.
    """
    vec = CountVectorizer(stop_words='english', ngram_range=ngram_range).fit(corpus.dropna())
    bag_of_words = vec.transform(corpus.dropna())
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
```

Top 20 Bigrams in Descriptions:

	Bigram	Frequency
0	conspiracy commit	281
1	wire fraud	172
2	identity theft	115
3	united states	104
4	unlawful flight	95
5	flight avoid	95

Top 20 Trigrams in Descriptions:

	Trigram	Frequency
0	unlawful flight avoid	95
1	flight avoid prosecution	92
2	aggravated identity theft	90
3	commit wire fraud	85
4	conspiracy commit wire	77
5	conspiracy commit computer	56

STEP: ENTRIES WITH TATTOO

Analyze the 'scars_and_marks' column to identify entries where tattoos are mentioned.

```
# Analyze the 'scars_and_marks' column to identify entries with tattoos
# Create a new boolean column 'has_tattoo'
# Use .loc to avoid SettingWithCopyWarning
df_selected.loc[:, 'has_tattoo'] = df_selected['scars_and_marks'].apply(
    lambda x: 'tattoo' in str(x).lower() if pd.notnull(x) else False
)

# Print the number of entries identified as having tattoos
tattoo_count = df_selected['has_tattoo'].sum()
print(f"Number of entries identified as having tattoos: {tattoo_count}")

# Display the head of the df_selected DataFrame with the new has_tattoo column
display(df_selected.head())
```

scars_and_marks	has_tattoo
None	False
Matthews has tattoos on his left and right for...	True
Doghmi has a large tattoo on her right leg of ...	True
None	False
Rodriguez Singh has tattoos on her back, left ...	True

Number of entries identified as having tattoos: 120

STEP: CRIME ANALYSIS

The n-gram analysis, along with word frequency analysis and the 'subjects' column, helped identify specific keywords and phrases related to violent and white-collar crimes.

These terms were then used to categorize each entry in the dataset by checking for their presence in the 'description', 'title', and 'subjects' columns.

```
violent_crime_keywords = [
    'murder', 'homicide', 'assault', 'battery', 'robbery', 'kidnapping',
    'violent crime', 'fugitive', 'ten most wanted', 'criminal enterprise investigations',
    'additional violent crimes', 'violent crime - murders'
]

white_collar_crime_keywords = [
    'fraud', 'wire fraud', 'mail fraud', 'computer fraud', 'identity theft',
    'money laundering', 'conspiracy commit wire', 'commit wire fraud',
    'aggravated identity theft', 'white-collar crime', 'cyber\'s most wanted'
]
```

proportion_violent_crime_tattoo = violent_crime_with_tattoos / total_violent_crimes if total_violent_crimes > 0 else 0

proportion_white_collar_crime_tattoo = white_collar_crime_with_tattoos / total_white_collar_crimes if total_white_collar_crimes > 0 else 0

Proportion of individuals with tattoos in Violent Crimes: 0.1393
Proportion of individuals with tattoos in White-Collar Crimes: 0.0335

STEP: HYPOTHESIS TESTING

For comparing proportions of a categorical variable between two or more groups, the chi-squared test of independence is the correct statistical method.

```python
from scipy.stats import chi2_contingency
import numpy as np
contingency_table = np.array([
    [total_violent_crimes - violent_crime_with_tattoos, violent_crime_with_tattoos],
    [total_white_collar_crimes - white_collar_crime_with_tattoos, white_collar_crime_with_tattoos]
])

print("Contingency Table:")
display(contingency_table)

# Perform the chi-squared test
chi2_statistic, p_value, degrees_of_freedom, expected_frequencies = chi2_contingency(contingency_table)
```

WHY CHI-SQUARRED ANALYSIS?

We chose the chi-squared test because we are analyzing the relationship between two categorical variables: crime type (violent vs. white-collar) and the presence of tattoos (yes or no).

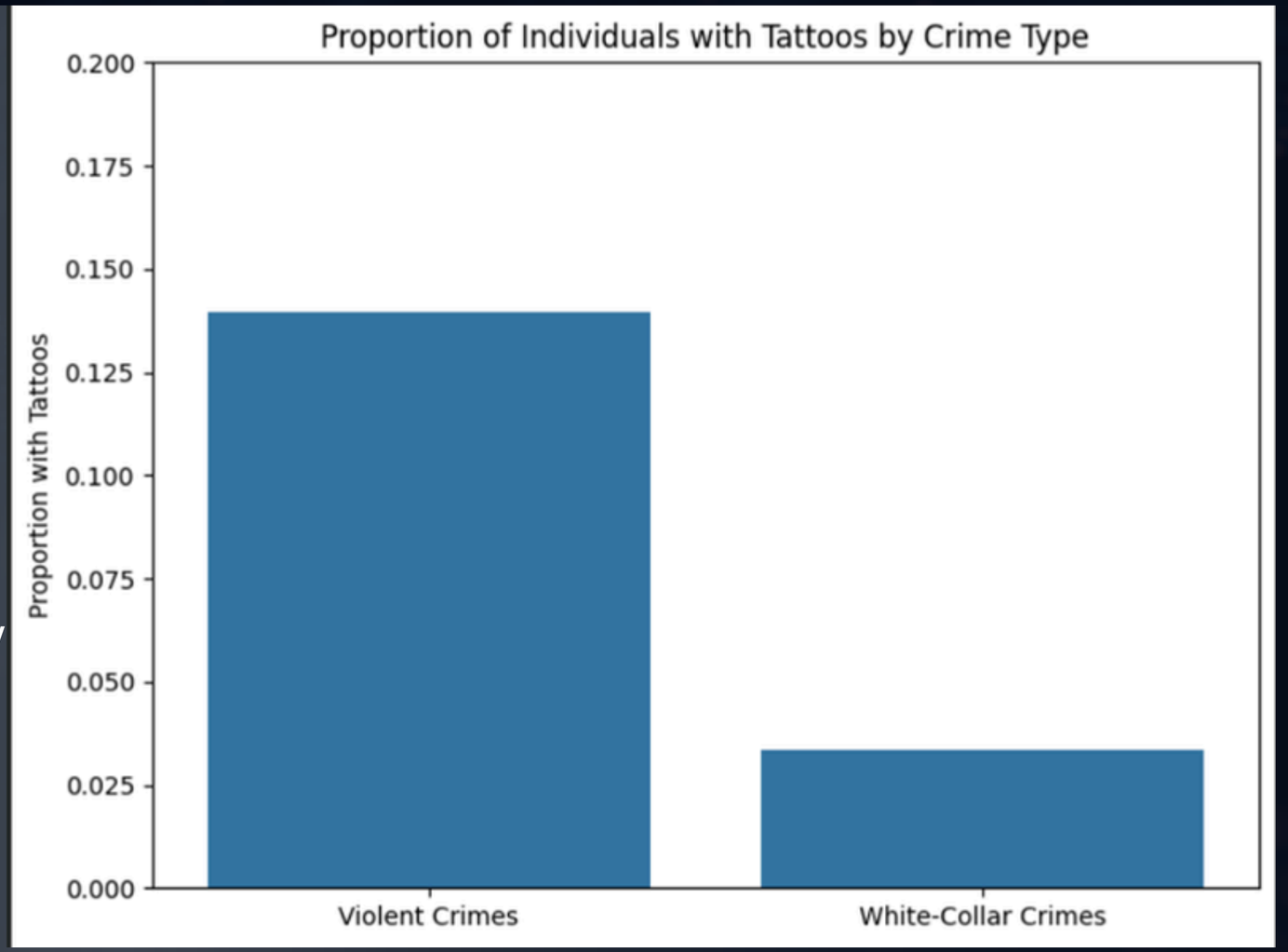
The chi-squared test of independence is the appropriate statistical test to determine if there is a statistically significant association between these two categorical variables. It compares the observed frequencies in each category to the frequencies that would be expected if there were no association between the variables.

STEP: HYPOTHESIS TESTING

Contingency Table:
array([[420, 68],
[231, 8]])
Chi-squared Statistic: 18.0951
P-value: 0.0000
Degrees of Freedom: 1
Expected Frequencies:
array([[436.98486933, 51.01513067],
[214.01513067, 24.98486933]])

The p-value (0.0000) is less than the significance level (0.05), so we reject the null hypothesis.

Conclusion: Based on the analysis, there is statistically significant evidence to support the hypothesis that individuals wanted for violent crimes are more likely to have tattoos than individuals wanted for white-collar crimes in this dataset.



CYBER ATTACK

On September 29, 2025, Asahi announced that its Japanese systems were experiencing a major disruption due to a cyber-attack.

The attack halted production at many of Asahi's domestic factories (six beer plants plus other soft-drink/food plants).

The attacker group is reported to be Qilin — a ransomware-as-a-service gang that claims to have stolen ~9,300 files (≈ 27 GB) from Asahi.

Does young adults are more prone to cyber crimes??

HYPOTHESIS 2

Research Question: Are cyber crimes more frequently associated with young adults (age 18-25) compared to older individuals?

Null Hypothesis (H0): There is no difference in the frequency of cyber crimes between young adults (age 18-25) and older individuals (age 40-65).

Alternative Hypothesis (H1): **Cyber crimes are more frequently associated with young adults (age 18-25) compared to older individuals (age 40-65).**

STEP: KEYWORD ANALYSIS

- **Method:**
  N-gram & keyword analysis
- **Keywords Used:**
  'computer fraud', 'identity theft', 'wire fraud' 'hacking', 'data breach', 'Cyber's Most Wanted'
- **Result:**
  107 entries successfully identified as cyber crimes
- **Purpose:** Created clean dataset for age-based hypothesis testing

```
cyber_crime_keywords = ['cyber', 'computer fraud', 'hacking', 'malware', 'phishing', 'ransomware',
                        'data breach', 'identity theft', 'wire fraud', 'conspiracy commit computer',
                        'commit computer fraud', 'aggravated identity theft', 'cyber¥'s most wanted']
```

```
cyber_crime_entries = df_selected[
    df_selected.apply(lambda row:
        any(keyword in str(row['description']).lower() for keyword in cyber_crime_keywords) or
        any(keyword in str(row['title']).lower() for keyword in cyber_crime_keywords) or
        (isinstance(row['subjects'], list) and 'Cyber¥'s Most Wanted' in row['subjects']),
        axis=1
)]

print(f"Number of entries identified as cyber crimes: {len(cyber_crime_entries)}")
display(cyber_crime_entries.head())
```

```
Number of entries identified as cyber crimes: 107
```

	description	title	hair	eyes	field_offices	modified	publication	subjects	sex
8	Wire Fraud Conspiracy; Wire Fraud; Money Laund...	FRAUDULENT REMOTE IT WORKERS FROM DPRK	None	None	[atlanta]	2025-06-26T21:19:41+00:00	2025-06-24T09:45:00	[Cyber's Most Wanted]	None
42	Conspiracy to Commit Wire Fraud and Bank Fraud...	JON CHANG HYOK	black	brown	[losangeles]	2025-05-21T18:12:58+00:00	2021-02-05T16:04:00	[Cyber's Most Wanted]	Male
43	Conspiracy to Commit Wire Fraud and Bank Fraud...	KIM IL	black	brown	[losangeles]	2025-05-21T18:10:20+00:00	2021-02-05T16:17:00	[Cyber's Most Wanted]	Male
44	Conspiracy to Commit Wire Fraud and Bank Fraud...	PARK JIN HYOK	black	brown	[losangeles]	2025-05-21T18:09:01+00:00	2018-08-30T09:36:00	[Cyber's Most Wanted]	Male
55	Conspiracy to Commit Computer Hacking; Conspir...	RIM JONG HYOK	None	None	[stlouis]	2025-05-06T19:09:30+00:00	2024-07-08T09:28:00	[Cyber's Most Wanted]	Male

STEP: ENTRIES BY AGE AND CYBER CRIME

- **Created Two Key Columns:**
  age_group: 'young adult', 'older adult', or 'other'
  is_cyber_crime: True or False

- **Method:**
  Applied filters to label each entry
  Enabled clear comparison between groups

- **Result:**
  Structured dataset ready for statistical testing
  Can now calculate proportions for each age group

```
# Create 'age_group' column
df_selected['age_group'] = 'other' # Default to 'other'
df_selected.loc[df_selected.index.isin(young_adult_entries.index), 'age_group'] = 'young adult'
df_selected.loc[df_selected.index.isin(older_adult_entries.index), 'age_group'] = 'older adult'

# Create 'is_cyber_crime' column
df_selected['is_cyber_crime'] = False # Default to False
df_selected.loc[df_selected.index.isin(cyber_crime_entries.index), 'is_cyber_crime'] = True

# Display the head of the DataFrame with the new columns
display(df_selected.head())
```

description	title	hair	eyes	field_offices	modified	publication	subjects	sex	age_group	is_cyber_crime
Las Vegas, Nevada\r\nJune 11, 2025	DEFACEMENT OF FEDERAL PROPERTY	None	None	[lasvegas]	2025-07-08T19:27:28+00:00	2025-06-25T09:42:00	[Seeking Information]	None	other	False
Conspiracy to Possess with Intent to Distribut...	TERRY MATTHEWS	brown	brown	[louisville]	2025-07-02T14:33:15+00:00	2025-07-02T08:03:00	[Criminal Enterprise Investigations]	Male	other	False
Auburn, Maine\r\nJune 1, 2021	CELESTE DIANA DOGHMI - AUBURN, MAINE	brown	brown	None	2025-07-02T12:35:52+00:00	2025-07-02T07:26:00	[ViCAP Missing Persons]	Female	other	False
Conspiracy to Commit Hostage Taking; Hostage T...	VITEL'HOMME INNOCENT	black	brown	[miami]	2025-07-01T15:31:48+00:00	2022-11-03T10:49:00	[Additional Violent Crimes]	Male	other	False

STEP: CRIME ANALYSIS

- **Proportion of Cyber Crimes by Age Group:**
  Young Adults (18-25): 1.56%
  Older Adults (40-65): 0.00%

- **Calculation Method:**
  Proportion = (Cyber Crimes in Age Group) / (Total Entries in Age Group)

- **Initial Observation:**
  A small difference exists in the sample data
  Statistical testing is needed to determine if this difference is significant

```
# 1. Calculate the total number of entries in the 'young adult' age group
total_young_adults = len(df_selected[df_selected['age_group'] == 'young adult'])

# 2. Calculate the number of cyber crime entries within the 'young adult' age group
young_adult_cyber_crimes = len(df_selected[(df_selected['age_group'] == 'young adult') & (df_selected['is_cyber_crime'] == True)])

# 3. Calculate the proportion of cyber crime entries in the 'young adult' group
proportion_young_adult_cyber_crime = young_adult_cyber_crimes / total_young_adults if total_young_adults > 0 else 0

# 4. Calculate the total number of entries in the 'older adult' age group
total_older_adults = len(df_selected[df_selected['age_group'] == 'older adult'])

# 5. Calculate the number of cyber crime entries within the 'older adult' age group
older_adult_cyber_crimes = len(df_selected[(df_selected['age_group'] == 'older adult') & (df_selected['is_cyber_crime'] == True)])

# 6. Calculate the proportion of cyber crime entries in the 'older adult' group
proportion_older_adult_cyber_crime = older_adult_cyber_crimes / total_older_adults if total_older_adults > 0 else 0

# 7. Print the calculated proportions
print(f"Proportion of cyber crime entries in young adults (18-25): {proportion_young_adult_cyber_crime:.4f}")
print(f"Proportion of cyber crime entries in older adults (40-65): {proportion_older_adult_cyber_crime:.4f}")
```

```
Proportion of cyber crime entries in young adults (age 18-25): 0.0156
Proportion of cyber crime entries in older adults (age 40-65): 0.0000
```

STEP: HYPOTHESIS TESTING

- **Statistical Test:** Chi-Squared Test of Independence
- **Results:**
  p-value = 0.9817
  Significance level (α) = 0.05
- **Interpretation:**
  p-value > α
  We **FAIL TO REJECT** the null hypothesis
- **Conclusion:**
  No statistically significant difference found in cyber crime proportions between young adults (18-25) and older adults (40-65)
  The data does not support the hypothesis that cyber crimes are more frequent among young adults

```python
from scipy.stats import chi2_contingency
import numpy as np

# Create a contingency table
# Rows: Young Adults, Older Adults
# Columns: Not Cyber Crime, Cyber Crime
contingency_table = np.array([
    [total_young_adults - young_adult_cyber_crimes, young_adult_cyber_crimes],
    [total_older_adults - older_adult_cyber_crimes, older_adult_cyber_crimes]
])

print("Contingency Table:")
display(contingency_table)

# Perform the chi-squared test
chi2_statistic, p_value, degrees_of_freedom, expected_frequencies = chi2_contingency(contingency_table)
```

```
Hypothesis: Are cyber crimes more frequently associated with young adults (age 18-25) compared to older individuals (age 40-65)?

Proportion of cyber crime entries in young adults (age 18-25): 0.0156
Proportion of cyber crime entries in older adults (age 40-65): 0.0000

Statistical Test Summary (Chi-squared test):
Test performed: Chi-squared test of independence
Calculated p-value: 0.9817
Significance level (alpha): 0.05
Result: We fail to reject the null hypothesis.
```

LIMITATIONS

API Page Size Limitation: Initially, we tried to fetch all 1000 records in a single API call, but the FBI API had a limitation on the page size, returning only 50 entries. We had to implement a pagination strategy using a loop to retrieve the desired number of records.

Missing Data: A significant challenge was the large amount of missing data across many columns in the dataset, particularly the age information (age_min, age_max). This limited the number of entries we could use for the age-based hypothesis testing and highlighted the importance of careful data cleaning and acknowledging data limitations.

REFERENCES

Sorby, M., “Juror perceptions of the stereotypical violent crime defendant,” Victims & Offenders, vol. 15, no. 6, 2020, pp. 746-767, DOI:10.1080/15564886.2020.1783017.

Pascual, M. G. (2025, October 30). Anatomy of a cyberattack with a hangover: How Japan was left without beer. EL PAÍS. https://english.elpais.com/technology/2025-10-30/anatomy-of-a-cyberattack-with-a-hangover-how-japan-was-left-without-beer.html

Vicens, A. J. (2025, October 8). Japan’s Asahi hack that halted beer production claimed by Qilin ransomware group. Reuters. https://www.reuters.com/world/asia-pacific/cybercriminals-claim-hack-japans-asahi-group-2025-10-07/?utm_source=chatgpt.com

Yoshida, K. (2025, October 3). Asahi blames ransomware for crippling Japanese beer plants. The Japan Times. Retrieved from https://www.japantimes.co.jp/business/2025/10/03/companies/seven-eleven-super-dry-asahi/?utm_source=chatgpt.com

THANK YOU


DEPARTMENT OF JUSTICE
FEDERAL BUREAU OF INVESTIGATION
FIDELITY
BRAVERY
INTEGRITY