ANALYSIS OF ACCIDENTS IN USA

Datasets

Car Accidents in USA

Format: CSV

NYC Motor Vehicle Collisions - Crashes

Format: JSON

URL: https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.json?accessType=DOWNLOAD

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import urllib.request
import json
```

Data Loading

Structured Data (CSV)

```
from google.colab import drive

drive.mount('/content/drive')
```

```
df_raw_veh = pd.read_csv('/content/drive/MyDrive/final project scripting/Datasets/veh_20.csv',sep=",",encoding = 'ISO-8859-1')
df_raw_veh.head()
```

```
vehicle_df = df_raw_veh[['CASENUM','STRATUMNAME', 'REGIONNAME','URBANICITYNAME','MONTHNAME','HOURNAME','HARM_EVNAME','MAN_COLLNA
```

```
vehicle_df.set_axis(['Case No.', 'Strat Injuries', 'Region','Rural/Urban','Month','Hour','Harmed','Collision Type','Hit & Run','
vehicle_df.head()
```

```
df_raw_per = pd.read_csv('/content/drive/MyDrive/final project scripting/Datasets/pers_20.csv',sep=",",encoding = 'ISO-8859-1')
```

```
person_df = df_raw_per[['CASENUM','PER_NO','REGIONNAME','PSU_VAR','URBANICITYNAME','MONTHNAME','HARM_EVNAME','MAN_COLLNAME','BOD
person_df.columns.to_list()
```

```
person_df.set_axis(['Case No.','Person No.','Region','PSU','Rural/Urban','Month','Harmed','Collision Type','Body Type','Car Mode
```

```
person_df.head()
```

Semi Structures data (JSON)

```
nyc_accidents_url = 'https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.json?accessType=DOWNLOAD'
```

```
response = urllib.request.urlopen(nyc_accidents_url)
response.code

# As the response code is 200, We can code further
```

```
json_string = response.read().decode('utf-8')
```

```
eq_parsed_json = json.loads(json_string)

nyc_accident_data = eq_parsed_json['data']
nyc_accident_data[1]
```

```
column_names = [col['name'] for col in eq_parsed_json['meta']['view']['columns']]
#Extracted the column names from the 'eq_parsed_json' JSON data

nyc_df = pd.DataFrame(nyc_accident_data, columns=column_names)
#Created a DataFrame 'nyc_df' using the 'nyc_accident_data' and specified column names

print(nyc_df.columns)

nyc_df.head(5)
```

Data Cleaning

```
null_count = person_df.isna().sum()
print(null_count)
```

```
person_df['Body Type'].fillna('Unspecified', inplace=True)
person_df['Car Model Year'].fillna('Unspecified', inplace=True)
person_df['Impact'].fillna('Unspecified', inplace=True)
person_df['Vehicle Type'].fillna('Unspecified', inplace=True)
```

```
null_count = person_df.isna().sum()
print(null_count)
```

```
null_count = vehicle_df.isna().sum()
print(null_count)
```

```
print(person_df.dtypes)
```

```
print(vehicle_df.dtypes)
```

Data cleaning for Semi-structured data

```
columns_to_keep = ['COLLISION_ID','CRASH DATE', 'CRASH TIME', 'NUMBER OF PERSONS INJURED',
       'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIANS INJURED',
       'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED',
       'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED',
       'NUMBER OF MOTORIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 1',
       'CONTRIBUTING FACTOR VEHICLE 2',
       'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2']

nyc_df = nyc_df[columns_to_keep]

nyc_df.head(10)
```

```
# As we just want to focus on data for 2020, we are filtering and just keeping the data for year 2022 in nyc_df

# Converting 'CRASH DATE' to datetime format
nyc_df['CRASH DATE'] = pd.to_datetime(nyc_df['CRASH DATE'])

# Creating a new column for the year
nyc_df['YEAR'] = nyc_df['CRASH DATE'].dt.year

# Filtering the dataset to keep only the rows for the year 2020
nyc_df = nyc_df[nyc_df['YEAR'] == 2020]

nyc_df
```

```
#checking for null values

nyc_null_count = nyc_df.isna().sum()
print(nyc_null_count)
```

```
# replacing missing values with 'Unspecified' value
nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].fillna('Unspecified', inplace=True)
nyc_df['CONTRIBUTING FACTOR VEHICLE 2'].fillna('Unspecified', inplace=True)
nyc_df['VEHICLE TYPE CODE 1'].fillna('Unspecified', inplace=True)
nyc_df['VEHICLE TYPE CODE 2'].fillna('Unspecified', inplace=True)

#checking for missing values again
nyc_null_count = nyc_df.isna().sum()
print(null_count)
```

Analysis

```
person_df.head()
```

```
person_df.describe(include='all')
```

```
city = person_df['Rural/Urban'].value_counts().nlargest(5)
print(city)
city.plot.pie()
```

```
Region=person_df['Region'].value_counts()

Region.plot.pie()
```

```
group=person_df.groupby('Vehicle Type')['Case No.'].count().sort_values(ascending=False)

T10= group.head(10)
plt.xticks(rotation=90)
plt.scatter(T10.index,T10.values, color='blue', marker='o', label='Scatter Plot')
```

```
age=person_df['Age']

def categorize_age(age):
    if age <= 10:
        return '0–10 years'
    elif 11 <= age <= 25:
        return '11–25 years'
    elif 26 <= age <= 40:
        return '26–40 years'
    elif 41 <= age <= 60:
        return '41–60 years'
    elif 61 <= age <= 75:
        return '61–75 years'
    else:
        return '75+ years'

person_df['age_group']=person_df['Age'].apply(categorize_age)

group_age=person_df.groupby('age_group')['Case No.'].count()

print(group_age)

colors = ['red', 'green', 'blue', 'orange','black','purple']

# Create a bar chart
plt.bar(group_age.index, group_age.values, color=colors)

plt.xticks(rotation=45)
# Set labels and title
plt.xlabel('Age Group')
plt.ylabel('Number of Accidents')

# Display the plot
plt.show()


plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Case No.', data=vehicle_df, estimator='count', ci=None, color='orange')
plt.title('Number of Accidents Over Time')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()


top_5_conditions = vehicle_df['Road Condition'].value_counts().head(5).index

filtered_df = vehicle_df[vehicle_df['Road Condition'].isin(top_5_conditions)]

plt.figure(figsize=(12, 6))
sns.countplot(x='Road Condition', hue='Injury', data=filtered_df, palette='viridis')
plt.title('Count of Injuries by Road Condition (Top 5)')
plt.xlabel('Road Condition')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()


plt.figure(figsize=(16, 8))
collision_injury_matrix = pd.crosstab(vehicle_df['Collision Type'], vehicle_df['Injury'])
sns.heatmap(collision_injury_matrix, annot=True, cmap='viridis', fmt='d', cbar_kws={'label': 'Count'})
plt.title('Count of Injuries by Collision Type and Injury Severity')
plt.xlabel('Injury Severity')
plt.ylabel('Collision Type')
plt.show()
```

Question:

What are the primary contributing factors to vehicle collisions in New York City?

```
def plot_top_contributing_factors(column_name, title):
    plt.figure(figsize=(10, 6))

    # Filtering out 'Unspecified' values
    filtered_df = nyc_df[nyc_df[column_name] != 'Unspecified']

    # Getting the top 5 contributing factors
    top_factors = filtered_df[column_name].value_counts().nlargest(5)

    # Bar plot with a custom color palette
    sns.barplot(y=top_factors.index, x=top_factors.values, palette='viridis')

    # Adding data labels to the bars
    for i, v in enumerate(top_factors.values):
        plt.text(v + 5, i, str(v), color='black', va='center', fontweight='bold')

    plt.title(title)
    plt.xlabel('Count')
    plt.ylabel('Contributing Factor')
    plt.show()
```

Question:

What is the distribution of collision severity (in terms of the number of persons injured) across the top 5 contributing factors in vehicle collisions?

```
# Determining the top 5 contributing factors
top_factors = nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].value_counts().nlargest(5).index

# Filtering the DataFrame for only the top 5 contributing factors
filtered_df = nyc_df[nyc_df['CONTRIBUTING FACTOR VEHICLE 1'].isin(top_factors)]

# Setting the figure size to increase the width
plt.figure(figsize=(18, 10))  # Adjust the width (10) and height (6) as needed

# Creating a stacked histogram
sns.histplot(data=filtered_df, x='CONTRIBUTING FACTOR VEHICLE 1', hue='NUMBER OF PERSONS INJURED', multiple='stack')
plt.title('Distribution of Collision Severity by Top 5 Contributing Factors')
plt.show()
```

```
# Merging person_df and vehicle_df on 'Case No.'
combined_df = pd.merge(person_df, vehicle_df, on='Case No.')
combined_df.columns
```

Question:

How does the distribution of the top N vehicle types arriving after crash at hospitals vary, and which experience the highest volume of cases for these specific vehicle types?

```
top_n_vehicle_types = 5

# Calculating the top N most frequent vehicle types
top_vehicle_types = combined_df['Vehicle type'].value_counts().nlargest(top_n_vehicle_types).index

# Filtering the DataFrame for the top N vehicle types
filtered_df = combined_df[combined_df['Vehicle type'].isin(top_vehicle_types)]

# Creating a table of counts for each hospital and vehicle type
```