Assignment05_Candice Yao

May 15, 2023

0.1 Question 1

- Read data from `Crime.csv` then print the names of the columns which have missing values.
- Drop any row that has missing values.
- Which three subcategories have the top three event counts?

[3]:
```
#pip install seaborn
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[4]:
```
# Read data from Crime.csv
df = pd.read_csv('/home/jovyan/Assignment5/Crime.csv')
```

[5]:
```
# replace 'unknown' to nans so they all counts
df.replace('UNKNOWN', np.nan,inplace = True)
```

[6]:
```
# print the names of the columns which have missing values.
df_nan = df.loc[:,df.isna().sum(axis=0)>0]
print(df_nan.columns)
```

```
Index(['Occurred Date', 'Occurred Time', 'Reported Time', 'Crime Subcategory',
       'Precinct', 'Sector', 'Beat', 'Neighborhood'],
      dtype='object')
```

[7]:
```
# Drop any row that has missing values.
df = df.loc[df.isna().sum(axis=1)==0,:]
```

[8]:
```
# Which three subcategories have the top three event counts?
df['Crime Subcategory'].value_counts()[:3]
```

[8]:
```
CAR PROWL          144122
THEFT-ALL OTHER     52388
THEFT-SHOPLIFT      47227
Name: Crime Subcategory, dtype: int64
```

Car prowl, theft-all other, and theft-shoplift have the top three event counts.

0.2 Question 2

- Create two new columns (these two columns should be datetime objects):
  - `Occurred DateTime`= `Occurred Date` + `Occurred Time`
  - `Reported DateTime` = `Reported Date` + `Reported Time`
- Delete the following columns: `Occurred Date`, `Occurred Time`, `Reported Date`, `Reported Time`
- **Hints**:
  - The panadas function **pd.to_datetime** can help
  - The loaded Occurred/Reported Time appears to be a float like '1930.0'. This should be interpreted as '19:30:00' eventually

[13]:
```
# preprosses the format of the two time columns so they are all 6-digit with␣
↪one decimal number
df['Occurred Time'] = df['Occurred Time'].map(lambda x: (6 - len(str(x)))* '0'␣
↪+ str(x))
df['Reported Time'] = df['Reported Time'].map(lambda x: (6 - len(str(x)))* '0'␣
↪+ str(x))
```

[14]:
```
# create new columns
df['Occurred DateTime'] = df['Occurred Date'] + df['Occurred Time']
df['Reported DateTime'] = df['Reported Date'] + df['Reported Time']
```

[15]:
```
# oncert new column to Datetime
df['Occurred DateTime'] = pd.to_datetime(df['Occurred DateTime'],format='%m/%d/
↪%Y%H%M.%f', errors='coerce')
df['Reported DateTime'] = pd.to_datetime(df['Reported DateTime'],format='%m/%d/
↪%Y%H%M.%f', errors='coerce')
```

[16]:
```
# remove obsolete columns
df = df.drop(columns = ['Occurred Date','Occurred Time','Reported␣
↪Date','Reported Time'])
```

0.3 Question 3

- The crime subcategories are too fine-grained for our analysis. So regroup several subcategories together into 8 major categories: `[Thefts, Burglary, Robbery, Alcohol_Drug, Sex_Related, Homicide, Misc]`
- Construct a mapping dictionary from the subcategories to the corresponding major crime types, then use it by creating a new column called `crime_type`. mapping dictionary.:
  - For example, we would like to map `CAR PROWL`, `MOTOR VEHICLE THEFT`, … to `thefts`.
  - We would like to map `ROBBERY-STREET`, `ROBBERY-RESIDENTIAL`,… to `robbery`, etc.

hint: you can use df[col].map to map a column to values in a dictionary

[9]:
```
#Your Code Here
mappingDict={
    'MOTOR VEHICLE THEFT': 'Thefts',
    'THEFT-ALL OTHER': 'Thefts',
    'THEFT-BUILDING': 'Thefts',
    'THEFT-BICYCLE': 'Thefts',
    'THEFT-SHOPLIFT': 'Thefts',
    'BURGLARY-COMMERCIAL': 'Burglary',
    'BURGLARY-RESIDENTIAL-SECURE PARKING': 'Burglary',
    'BURGLARY-COMMERCIAL-SECURE PARKING': 'Burglary',
    'BURGLARY-RESIDENTIAL': 'Burglary',
    'ROBBERY-STREET': 'Robbery',
    'ROBBERY-RESIDENTIAL': 'Robbery',
    'ROBBERY-COMMERCIAL': 'Robbery',
    'LIQUOR LAW VIOLATION': 'Alcohol_Drug',
    'DUI': 'Alcohol_Drug',
    'RAPE': 'Sex_Related',
    'PROSTITUTION': 'Sex_Related',
    'SEX OFFENSE-OTHER': 'Sex_Related',
    'AGGRAVATED ASSAULT-DV': 'Sex_Related',
    'AGGRAVATED ASSAULT': 'Sex_Related',
    'PORNOGRAPHY': 'Sex_Related',
    'HOMICIDE': 'Homicide',
    'CAR PROWL': 'Misc',
    'WEAPON': 'Misc',
    'ARSON': 'Misc',
    'GAMBLE': 'Misc',
    'DISORDERLY CONDUCT': 'Misc',
    'FAMILY OFFENSE-NONVIOLENT': 'Misc',
    'LOITERING': 'Misc'
}
```

[20]:
```
df['crime_type'] = df['Crime Subcategory'].map(mappingDict)
```

[10]:
```
# Drop any row that has missing values in the new column:
df = df.loc[df.isna().sum(axis=1)==0,:]
```

0.4 Question 4

- Compare `visually` the average time gap between **Reported DateTime** vs **Occurred DateTime** of different crimes types.

Hints: - You can get the time in seconds from datetime column using `df[col_name].dt.total_seconds()`. - You can convert seconds to datetime usine `pd.to_timedelta(df["sec"], unit='s')`

[25]:
```
# add a column for displaying time gap
df['Time Gap'] = df['Reported DateTime'] - df['Occurred DateTime']
```

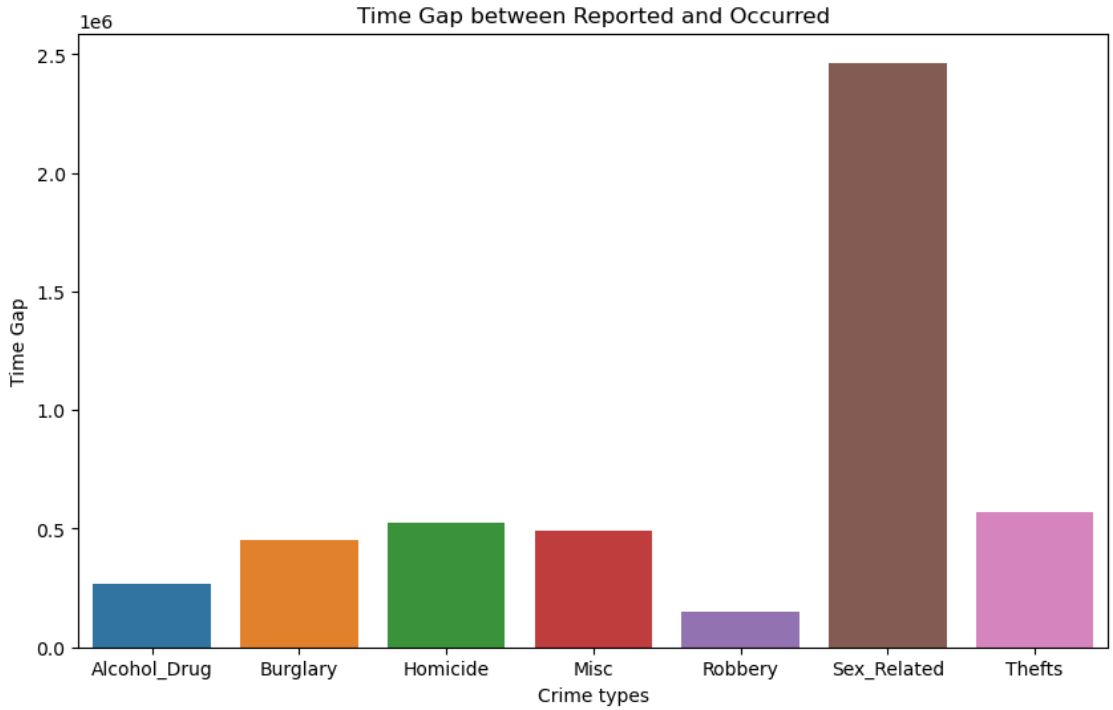
[26]:
```
# convert datetime to seconds
df['Time Gap'] = df['Time Gap'].dt.total_seconds()
```

[27]:
```
# display mean time gap by crime types
df_gap = df[['Time Gap','crime_type']].groupby('crime_type').mean()
```

[28]:
```
# display time gap in DateTime format
df_datetime_gap = pd.to_timedelta(df_gap['Time Gap'], unit = 's')
df_datetime_gap
```

[28]:
```
crime_type
Alcohol_Drug     3 days 02:46:54.251934651
Burglary         5 days 04:57:10.797294528
Homicide         6 days 01:51:18.154981550
Misc             5 days 16:17:34.055966795
Robbery          1 days 16:43:11.407206783
Sex_Related     28 days 12:40:00.219719857
Thefts           6 days 13:44:00.817351151
Name: Time Gap, dtype: timedelta64[ns]
```

[29]:
```
# plot time gap visually
fig = plt.figure(figsize=(10, 6))
sns.barplot(data=df_gap, x=df_gap.index.to_list(), y=df_gap['Time Gap'].
 ↪to_list(), ax=fig.gca())
plt.title('Time Gap between Reported and Occurred')
plt.xlabel('Crime types')
plt.ylabel('Time Gap')
plt.show()
```



0.5 Question 5

- Show `visually` the top 5 most dangerous neighorhood in the EAST Precinct (with number of crime events in each neighorhood)

[30]:
```
# create a dataframe with only crime taken place in the east precinct
df_east = df.loc[df['Precinct'] == 'EAST',:]
```

[31]:
```
# group the df and make it into a dictionary
top_five = df_east[['Neighborhood','crime_type']].groupby('Neighborhood').
 ↪size().sort_values(ascending= False)[:5].to_dict()
```

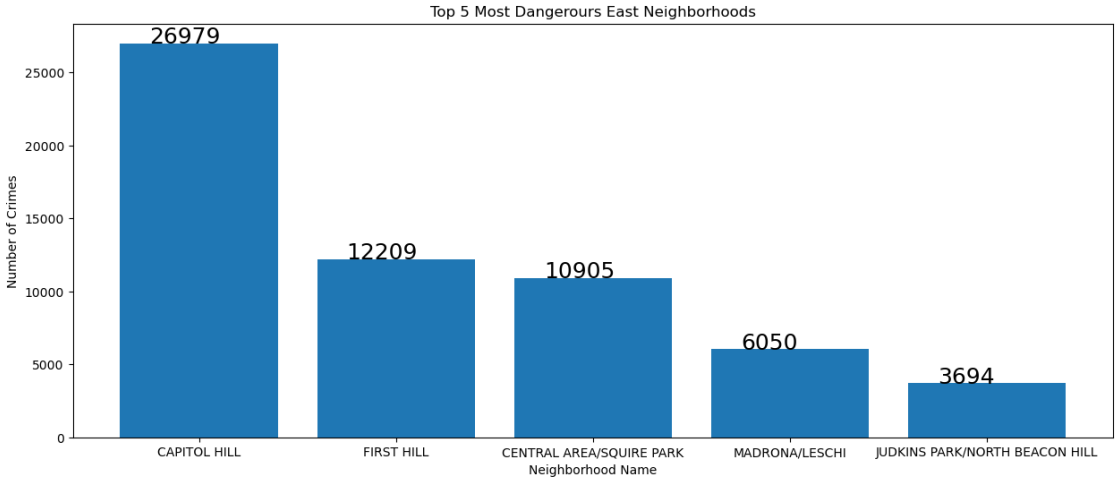
[32]:
```
# create a list of the top five most dangerous neighborhoods
top_five_neigh = list(top_five.keys())
```

[33]:
```
# create a dataframe for only the top five most dangerous neighborhoods
top_five_df = df_east.loc[df_east['Neighborhood'].isin(top_five_neigh),:]
```

[34]:
```
# get the crime numbers and make them a list
top_five_counts = top_five_df['Neighborhood'].value_counts().values.tolist()
```

[35]:
```
# plot the data
plt.figure(figsize = (15,6))
```

```
plt.bar(top_five_df['Neighborhood'].value_counts().index,␣
 ↪top_five_df['Neighborhood'].value_counts().values)
plt.title('Top 5 Most Dangerours East Neighborhoods')
plt.xlabel('Neighborhood Name')
plt.ylabel('Number of Crimes')
for i,j in enumerate(top_five_counts):
    plt.text(i-0.25, j, j,size=18)
plt.show()
```



0.6 Question 6

- Create column called **year** that shows the year of the crime. Which year has the lowest and highest crime event counts, respectively?
- Plot the number of `Thefts` crimes in the last ten years

[17]:
```
# create a year column and convert it into string
# get rid of the decimal at the end by slicing
df['year'] = df['Occurred DateTime'].dt.year.astype(int)
```

[18]:
```
df[['year','Crime Subcategory']].groupby('year').size().sort_values()
```

[18]:
```
year
1908    1
1964    1
1973    1
1975    1
1978    1
1980    1
1985    1
1986    1
```

```
1988        1
1989        1
1979        2
1981        2
1991        2
1994        2
1997        3
1996        3
1999        5
1993        5
1995        5
2002       15
1998       18
2003       25
2000       33
2004       36
2005       40
2001       49
2006       92
2007      601
2019     1554
2012    40764
2011    41014
2008    42263
2010    43032
2009    44729
2013    45258
2015    47487
2016    48949
2014    49095
2017    49947
2018    50675
dtype: int64
```

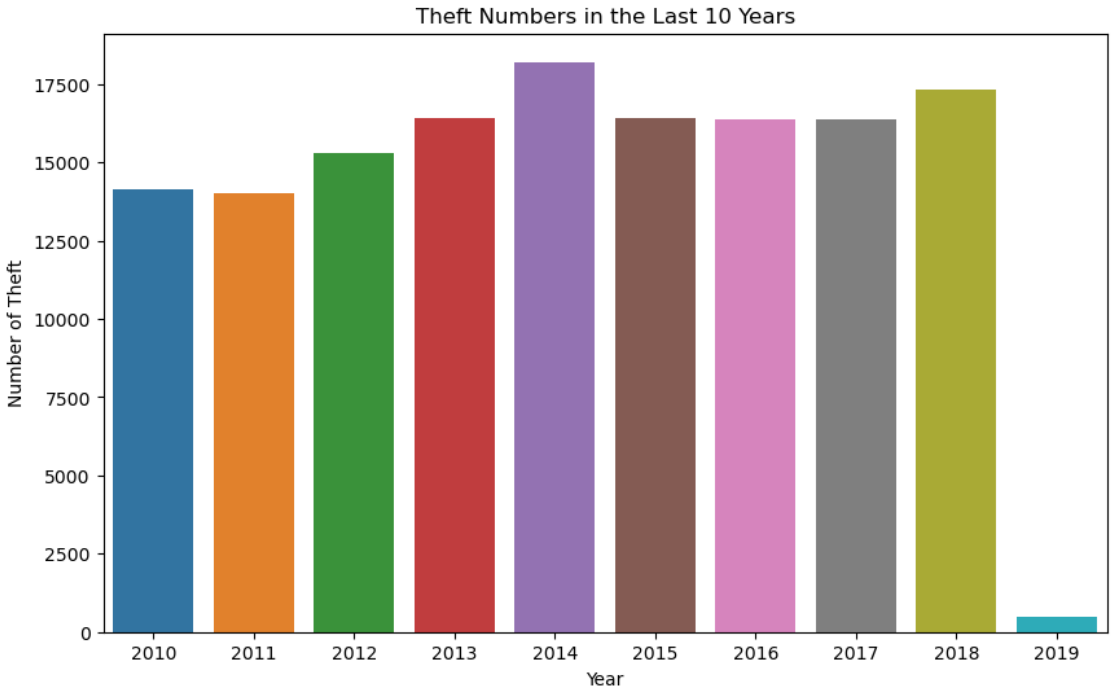
According to the result above, 1908,1964,1973,1975,1978,1980,1985,1986,1988,1989 all have lowest crimes at 1 and 2014 has the highest number of crimes has 47173.

[42]:
```
df_last_10_yrs = df.loc[(2010 <= df['year']) & (df['year'] <= 2019) &␣
↪(df['crime_type']=='Thefts'),:]
```

[43]:
```
# group the data frame for plotting
df_theft = df_last_10_yrs[['year','crime_type']].groupby('year').size()
```

[44]:
```
# visualization
fig = plt.figure(figsize=(10, 6))
sns.barplot(x=df_theft.index, y=df_theft.to_list(), ax=fig.gca())
plt.title('Theft Numbers in the Last 10 Years')
```

```
plt.xlabel('Year')
plt.ylabel('Number of Theft')
plt.show()
```



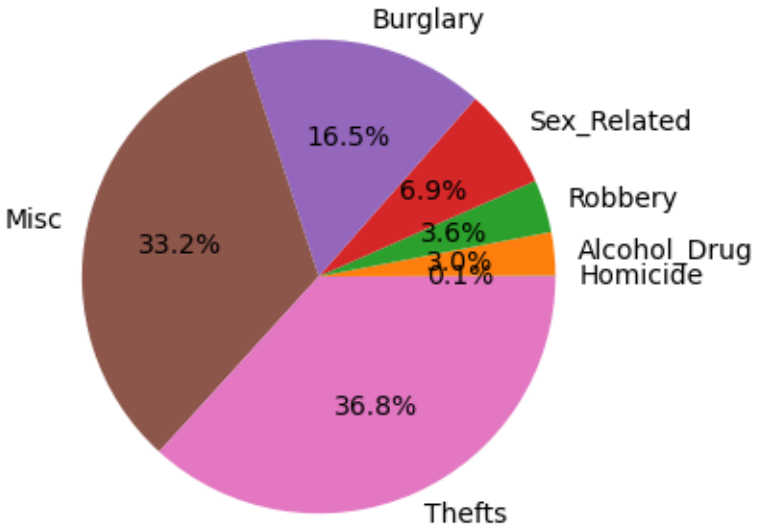
0.7 Question 7

- keep only crime events occurred in-between **2008-2018** (remove all other crimes).
- Show `visually` which crime type has the highest percentage of events

[19]:
```
# keep only crime events occurred in-between 2008-2018
df = df.loc[(2008 <= df['year']) & (df['year'] <= 2018),:]
```

[94]:
```
crimes = df.groupby('crime_type').size().sort_values().index.to_list()
crime_counts = list(df.groupby('crime_type').size().sort_values().values)
```

[95]:
```
# visualization
plt.figure(figsize=(8, 4))
plt.pie(crime_counts, labels=crimes, autopct='%1.1f%%')
plt.show()
# alternative
# sns.barplot(data=df_08_18, x=crimes ,y= crime_counts)
```



Theft has the largest amount of event counts at 36.8% according to the pie chart above.

0.8 Question 8

- Analyze the crime prevalence in terms of week days (create column called `day` that specifies the day of the crime).
- Which weekday (Sunday, Monday, ….Saturday) has highest/lowest daily Homicide crime count? (Visualize your findings)

Hint: use `.day_name()` with time object to get the day

[96]:
```
# get day and create a day column
df['day'] = df['Occurred DateTime'].dt.day_name()
```

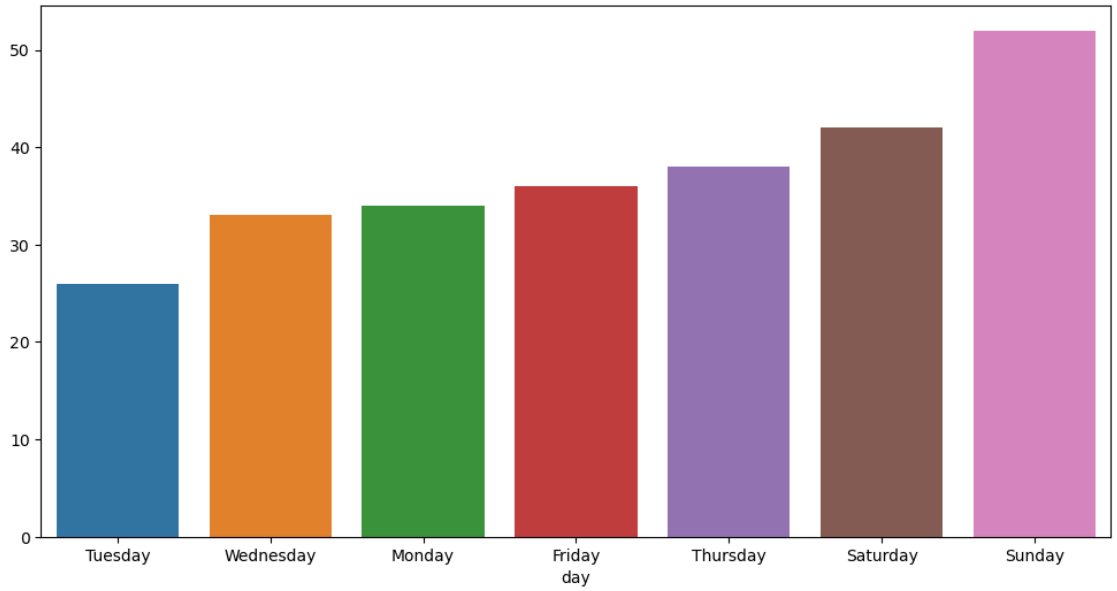
```
/tmp/ipykernel_94/650699073.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['day'] = df['Occurred DateTime'].dt.day_name()
```

[97]:
```
# create a dataframe with only homicides
df_homicide = df.loc[df['crime_type'] == 'Homicide']
```

[98]:
```
# display homicide counts by day
df_homicide.groupby('day').size().sort_values()
```

[98]:
```
day
Tuesday      26
Wednesday    33
Monday       34
Friday       36
Thursday     38
Saturday     42
Sunday       52
dtype: int64
```

[99]:
```
Days = df_homicide.groupby('day').size().sort_values().index
Count =  df_homicide.groupby('day').size().sort_values().values
# Visualization
plt.figure(figsize=(12, 6))
sns.barplot(data=df_homicide, x=Days ,y= Count)
plt.show()
```



Sunday has the highest amount of homicides while Tuesday has the lowest.

0.9 Question 9

- Compare `visually` the number of Alcohol_Drug crimes in each day. Which day has the highest number of Alcohol_Drug crimes?
- Show `visually` the evolve of Burglary crimes through the years.

[104]:
```
df_alc = df.loc[df['crime_type'] == 'Alcohol_Drug']
```

[69]:
```
df_alc.groupby('day').size().sort_values()
```

[69]:
```
day
Monday       1287
Tuesday      1316
Wednesday    1515
Thursday     1931
Friday       2310
Sunday       2502
Saturday     3095
dtype: int64
```

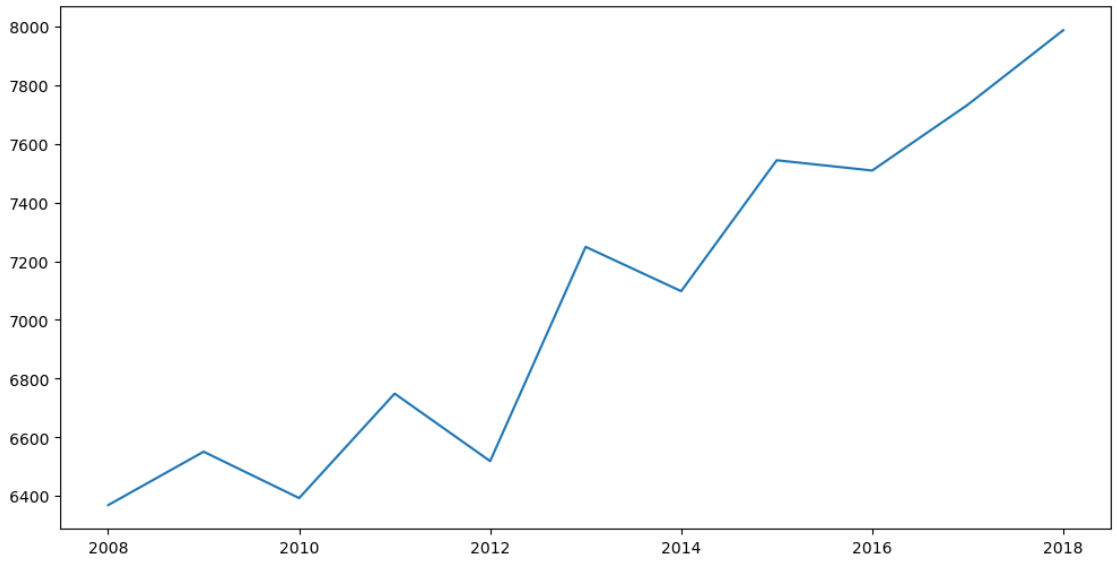
Saturday has the highest number of Alcohol Drug crimes.

[ ]:
```
# visualization
Days = df_alc.groupby('day').size().sort_values().index
Count =  df_alc.groupby('day').size().sort_values().values
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x=Days ,y= Count)
plt.show()
```

[105]:
```
# create a dataframe for burgalry crime
df_bur = df.loc[df['crime_type'] == 'Burglary']
```

[121]:
```
# visualization
plt.figure(figsize=(12,6))
plt.title('Evolution of Burgalry Crime Through the Years')
sns.lineplot(x = df_bur.year.value_counts().index ,y = df_bur.year.
 ↪value_counts().values)
plt.show()
```

[121]:
```
<AxesSubplot: >
```



0.10 Extra Credit Questions

- Consider the 24 hours window into 6 frames '12am-4am', '4am-8am', '8am-12pm', '12pm-4pm', '4pm-8pm', '8pm-12am'. Create a column called `Occured TimeFrame` that contains the crime occurred time frame.
- Compare `visually` time-frames in term of the number of crimes.
- Print the name of the most dangerous neighborhood in the WEST Precinct, Then Show `visually` the percentages of crimes during different TimeFrames in that neighborhood

[37]:
```
df['Occured TimeFrame'] = df['Occurred DateTime'].dt.hour
hours = df['Occured TimeFrame'].to_list()
```

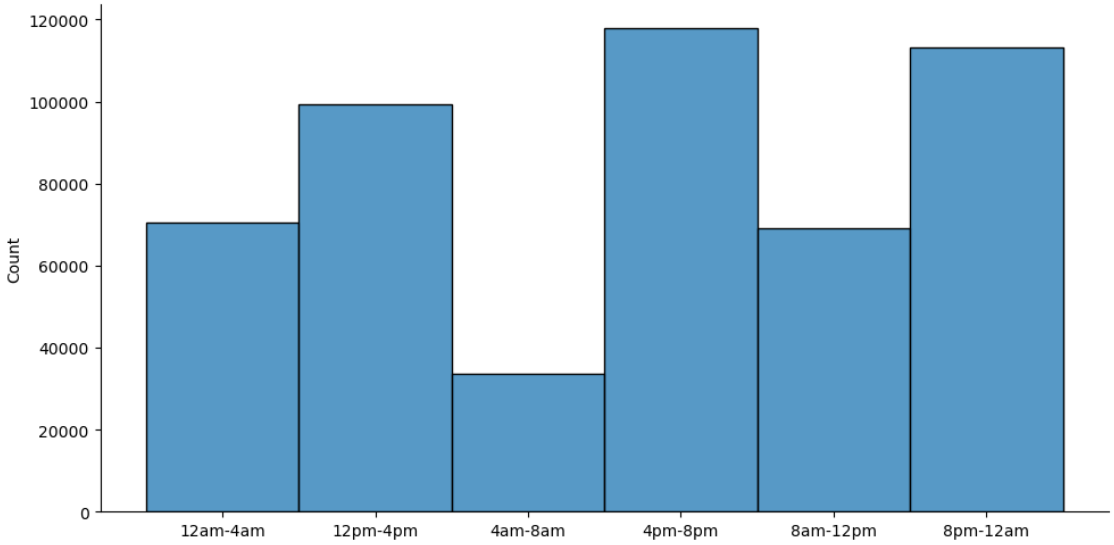
[38]:
```
# Define the time frames and their corresponding ranges
time_frames = ['12am-4am', '4am-8am', '8am-12pm', '12pm-4pm', '4pm-8pm',␣
 ↪'8pm-12am']
time_ranges = [(0, 4), (4, 8), (8, 12), (12, 16), (16, 20), (20, 24)]
```

[39]:
```
# loop through all the hours and assign time frame it
for i, hour in enumerate(hours):
    for time_range, time_frame in zip(time_ranges, time_frames):
        if hour in range(time_range[0], time_range[1]):
            hours[i] = time_frame
            break
```

[41]:
```
# assign the list back to the df so the time frame is in the column
df['Occured TimeFrame'] = hours
```

[25]:
```
sorted_hours = sorted(df['hours'])
sns.displot(data=df, x=sorted_hours, aspect=2)
```

[25]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7c28223b50>
```



[26]:
```
df_west = df.loc[df['Precinct'] == 'WEST',:]
```

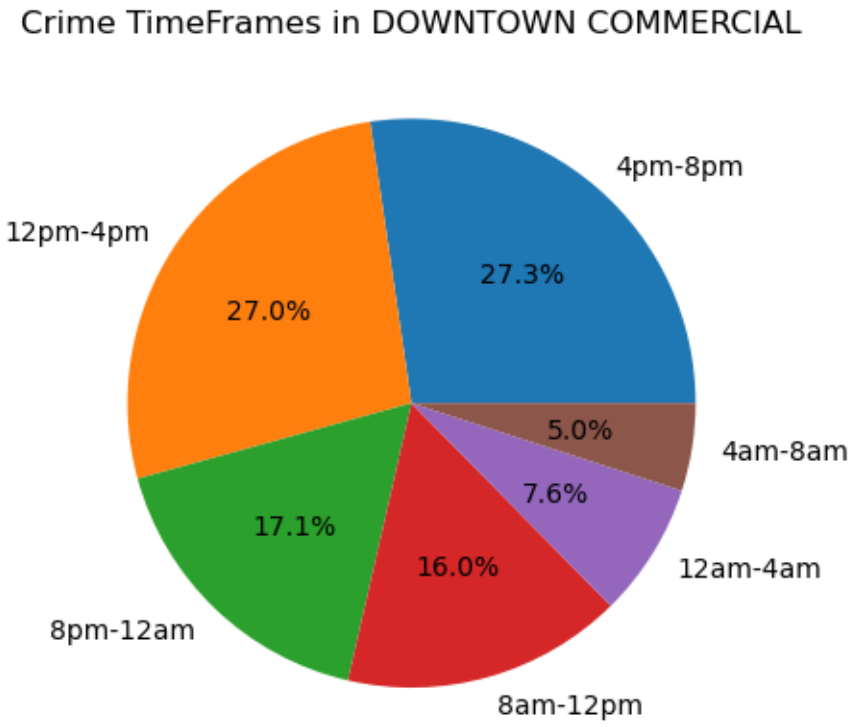
[33]:
```
#Print the name of the most dangerous neighborhood in the WEST Precinct
df_west.groupby('Neighborhood').size().sort_values().index[-1]
```

[33]:
```
'DOWNTOWN COMMERCIAL'
```

[34]:
```
df_DC = df_west.loc[df_west['Neighborhood'] == 'DOWNTOWN COMMERCIAL']
```

[35]:
```
# compute the counts of crimes for each time frame
counts = df_DC['hours'].value_counts()
```

[36]:
```
# plot a pie chart of the percentages
plt.pie(counts, labels=counts.index, autopct='%1.1f%%')
plt.title('Crime TimeFrames in DOWNTOWN COMMERCIAL')
plt.show()
```


Crime TimeFrames in DOWNTOWN COMMERCIAL
4pm-8pm
27.3%
12pm-4pm
27.0%
5.0%
4am-8am
7.6%
17.1%
16.0%
12am-4am
8pm-12am
8am-12pm

[ ]: