```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

1 -> import libraries

2 -> load dataset

3 -> columns , value count

4 -> categorised the data into cat , num , bool , int

5 -> Cause of Accident

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df_USA=pd.read_csv('/kaggle/input/us-accidents/US_Accidents_Dec21_updated.csv')
```

```
df_USA.head()
```

```
df_USA.columns
```

```
df_USA.dtypes.value_counts()
```

Shape of original data

```
df_USA.shape   # USA DATASET
```

num_col=df.select_dtypes('number') cat_col=df.select_dtypes('object') bool_col=df.select_dtypes('bool') float_col=df.select_dtypes('float64') float_col=df.select_dtypes('int64')

missing_data = df.isna().sum(axis=0).sort_values(ascending=True) missing_data = missing_data.to_frame() missing_data.columns = ['missing_count'] missing_data = missing_data.loc[missing_data['missing_count']>0]

missing_data

```
# COLUMN NUMBER has highest no of null value , we cam drop that col further
```

```
df_USA.describe()
```

```
df_USA.State.unique
```

```
df1=df_USA[df_USA['State']=='CA']
```

```
df1['IDD'] = df1['ID'].astype('str').str.extractall('(\d+)').unstack().fillna('').sum(axis=1).astype(int)
```

```
df1
```

```
#df1['ID'].astype('int64')
#df1['ID'] = df1['ID'].astype(int)
```

```
df1.head()
```

```
df1.shape
```

```
df1.columns
```

```
df1.duplicated().sum()
```

```
d1f=df1.dropna(subset=['Precipitation(in)'])
```

```
df1.shape
```

```
df1=df1.dropna(subset=['Temperature(F)','Wind_Chill(F)','Humidity(%)','Pressure(in)','Visibility(mi)','Wind_Direction', 'Wind_Speed(mph)',
                       'Weather_Condition'])
```

```
df1.shape
```

```
df1.isna().sum()/len(df1)*100
```

```
df1=df1.dropna(subset=['City','Sunrise_Sunset',
       'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight'])
```

```
df1.isna().sum()/len(df1)*100
```

```
df1['Weather_Condition'].value_counts()
```

```
df1.Side.unique()
```

```
df_cat=df1.select_dtypes('object')
df_num=df1.select_dtypes(np.number)
df_cat=df_cat.drop('ID',axis=1)
```

Double-click (or enter) to edit

```
df_cat=df1.select_dtypes('object')
col_name=[]
length=[]

for i in df_cat.columns:
    col_name.append(i)
    length.append(len(df_cat[i].unique()))
df_2=pd.DataFrame(zip(col_name,length),columns=['feature','count_of_unique_values'])
df_2
```

num_col=df.select_dtypes('number') cat_col=df.select_dtypes('object') bool_col=df.select_dtypes('bool') float_col=df.select_dtypes('float64') int_col=df.select_dtypes('int64')

```
df1.drop(['Description','Zipcode','Weather_Timestamp'],axis=1,inplace=True)
```

```
del df1['Airport_Code']
```

```
df_num.columns
```

```
len(df_num.columns)
```

```
df_cat.columns
```

```
#bool_col.columns
```

```
#int_col.columns
```

```
#cat_col.head()
```

```
Start coding or generate with AI.
```

```
len(df['City'].unique())
```

```
#cat_col
```

```
#num_col_1=df.select_dtypes('number')
```

Numeric Data

```
df_num=df1.select_dtypes(np.number)
col_name=[]
length=[]

for i in df_num.columns:
    col_name.append(i)
    length.append(len(df_num[i].unique()))
df_2=pd.DataFrame(zip(col_name,length),columns=['feature','count_of_unique_values'])
df_2
```

```
plt.figure(figsize=(15 ,9))
sns.heatmap(df_num.corr() , annot=True)
```

```
cities = df1['City'].unique()
len(cities)
```

```
accidents_by_cities = df1['City'].value_counts()
accidents_by_cities
```

```
#top 10 cities by number of accident
accidents_by_cities[:10]
```

```
fig, ax = plt.subplots(figsize=(8,5))
accidents_by_cities[:10].plot(kind='bar')
ax.set(title = 'Top 10 cities By Number of Accidents',
       xlabel = 'Cities',
       ylabel = 'Accidents Count')
plt.show()
```

```
accidents_severity = df1.groupby('Severity').count()['ID']
accidents_severity
```

```
fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(aspect="equal"))
label = [1,2,3,4]
plt.pie(accidents_severity, labels=label,
        autopct='%1.1f%%', pctdistance=0.85)
circle = plt.Circle( (0,0), 0.5, color='white')
p=plt.gcf()
p.gca().add_artist(circle)
ax.set_title("Accident by Severity",fontdict={'fontsize': 16})
plt.tight_layout()
plt.show()
```

```
df1['Start_Time'].dtypes
```

```
df1['End_Time'].dtypes
```

```
df1 = df1.astype({'Start_Time': 'datetime64[ns]', 'End_Time': 'datetime64[ns]'})
df1['Start_Time'].dtypes


df1['Start_Time'][2408]


df1['End_Time'][2408]


df1['start_date'] = [d.date() for d in df1['Start_Time']]
df1['start_time'] = [d.time() for d in df1['Start_Time']]


df1['end_date'] = [d.date() for d in df1['End_Time']]
df1['end_time'] = [d.time() for d in df1['End_Time']]


df1['end_time']


fig, ax = plt.subplots(figsize=(8,5))
sns.histplot(df1['Start_Time'].dt.hour, bins = 24)

plt.xlabel("Start Time")
plt.ylabel("Number of Occurence")
plt.title('Accidents Count By Time of Day')

plt.show()


fig, ax = plt.subplots(figsize=(8,5))
sns.histplot(df1['Start_Time'].dt.hour, bins = 24)

plt.xlabel("End_Time")
plt.ylabel("Number of Occurence")
plt.title('Accidents Count By Time of Day')

plt.show()


del df1['Start_Time']
del df1['End_Time']


#df.head()


%matplotlib inline
import os


fig, ax = plt.subplots(figsize=(8,5))
weather_conditions.sort_values(ascending=False)[:20].plot(kind='bar')
ax.set(title = 'Weather Conditions at Time of Accident Occurence',
       xlabel = 'Weather',
       ylabel = 'Accidents Count')
plt.show()
```

most accidents happened when the weather was 'fair'. Perhaps weather (bad weather) was not a big contributing factor to accidents.

```
#df_num.head()


# shape of original data -> 2845342 rows


df.shape  # shape of data after removing null values


df_num.shape


# Accidents by order of severity (1 being lowest, and 4 being highest)

df1.groupby('Severity').count()['IDD']
```

```
# scatter plot


df_num.plot(kind='scatter', y='Start_Lat', x='Severity')


sns.jointplot(x=df_num.Start_Lat.values , y=df_num.Start_Lng.values,height=10)
plt.ylabel('Start lattitude', fontsize=12)
plt.xlabel('Start lattitude', fontsize=12)
plt.show()


sns.jointplot(x=df_num.End_Lat.values , y=df_num.End_Lng.values,height=10)
plt.ylabel('end lattitude', fontsize=12)
plt.xlabel('end longitude', fontsize=12)
plt.show()
```

Start coding or generate with AI.

Start coding or generate with AI.