us-accident-eda

todo- Talk about eda Talk About Dataset (source-What it contains and how it can be useful)

- Source-kaggle
- Information about accidents
- Is useful to prevent accidents
- No Data for New York
- List item

  .

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="us-accident-eda")
```

```
[jovian] Detected Colab notebook...
[jovian] Please enter your API key ( from https://jovian.ai/ ):
API KEY: ··········
[jovian] Error: The current API key is invalid or expired.
[jovian] Please enter your API key ( from https://jovian.ai/ ):
API KEY: ··········
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/simar4447/us-accident-eda
'https://jovian.ai/simar4447/us-accident-eda'
```

Data Download

```
pip install opendatasets --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opendatasets in /usr/local/lib/python3.7/dist-packages (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from opendatasets) (4.64.0)
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (from opendatasets) (1.5.12)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from opendatasets) (7.1.2)
```

```
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (2022.6.15)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (2.23.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (1.24.3)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (6.1.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle->opendatasets) (1.15.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle->opendatasets) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle->opendatasets) (2.10)
```

```
import opendatasets as od
dataset_url = 'https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents?resource=down
od.download(dataset_url)
```

```
Skipping, found downloaded files in "./us-accidents" (use force=True to force download)
```

```
data_filename='./us-accidents/US_Accidents_Dec21_updated.csv'
```

```

```

DATA PREPARATION AND CLEANING

- Load the file using pandas
- Look at some information regarding data and coloumns
- Fix any missing or incorrect values

```
import pandas as pd
```

```
df=pd.read_csv(data_filename)
```

```
df
```

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)

	ID	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)
0	A-1	3	2016-02-08 00:37:08	2016-02-08 06:37:08	40.108910	-83.092860	40.112060	-83.031870	3.230
1	A-2	2	2016-02-08 05:56:20	2016-02-08 11:56:20	39.865420	-84.062800	39.865010	-84.048730	0.747
2	A-3	2	2016-02-08 06:15:39	2016-02-08 12:15:39	39.102660	-84.524680	39.102090	-84.523960	0.055
3	A-4	2	2016-02-08 06:51:45	2016-02-08 12:51:45	41.062130	-81.537840	41.062170	-81.535470	0.123
4	A-5	3	2016-02-08 07:53:43	2016-02-08 13:53:43	39.172393	-84.492792	39.170476	-84.501798	0.500
...	...	...	...	...	...	...	...	...	...
2845337	A-2845338	2	2019-08-23 18:03:25	2019-08-23 18:32:01	34.002480	-117.379360	33.998880	-117.370940	0.543
2845338	A-2845339	2	2019-08-23 19:11:30	2019-08-23 19:38:23	32.766960	-117.148060	32.765550	-117.153630	0.338
2845339	A-2845340	2	2019-08-23 19:00:21	2019-08-23 19:28:49	33.775450	-117.847790	33.777400	-117.857270	0.561
2845340	A-2845341	2	2019-08-23 19:00:21	2019-08-23 19:29:42	33.992460	-118.403020	33.983110	-118.395650	0.772
2845341	A-2845342	2	2019-08-23 18:52:06	2019-08-23 19:21:31	34.133930	-117.230920	34.137360	-117.239340	0.537

2845342 rows × 47 columns

```
df.columns
```

```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 47 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   ID                     object 
 1   Severity               int64  
 2   Start_Time             object 
 3   End_Time               object 
 4   Start_Lat              float64
 5   Start_Lng              float64
 6   End_Lat                float64
 7   End_Lng                float64
 8   Distance(mi)           float64
 9   Description            object 
 10  Number                 float64
 11  Street                 object 
 12  Side                   object 
 13  City                   object 
 14  County                 object 
 15  State                  object 
 16  Zipcode                object 
 17  Country                object 
 18  Timezone               object 
 19  Airport_Code           object 
 20  Weather_Timestamp      object 
 21  Temperature(F)         float64
 22  Wind_Chill(F)          float64
 23  Humidity(%)            float64
 24  Pressure(in)           float64
 25  Visibility(mi)         float64
 26  Wind_Direction         object 
 27  Wind_Speed(mph)        float64
 28  Precipitation(in)      float64
 29  Weather_Condition      object 
 30  Amenity                bool   
 31  Bump                   bool   
 32  Crossing               bool   
 33  Give_Way               bool   
 34  Junction               bool   
 35  No_Exit                bool   
 36  Railway                bool   
 37  Roundabout             bool
```

```
 38  Station                bool
 39  Stop                   bool
 40  Traffic_Calming        bool
 41  Traffic_Signal         bool
 42  Turning_Loop           bool
 43  Sunrise_Sunset         object
 44  Civil_Twilight         object
 45  Nautical_Twilight      object
 46  Astronomical_Twilight  object
dtypes: bool(13), float64(13), int64(1), object(20)
memory usage: 773.4+ MB
```

```
df.describe()
```

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Number	T
count	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	2.845342e+06	1.101431e+06	2
mean	2.137572e+00	3.624520e+01	-9.711463e+01	3.624532e+01	-9.711439e+01	7.026779e-01	8.089408e+03	(
std	4.787216e-01	5.363797e+00	1.831782e+01	5.363873e+00	1.831763e+01	1.560361e+00	1.836009e+04	1
min	1.000000e+00	2.456603e+01	-1.245481e+02	2.456601e+01	-1.245457e+02	0.000000e+00	0.000000e+00	-8
25%	2.000000e+00	3.344517e+01	-1.180331e+02	3.344628e+01	-1.180333e+02	5.200000e-02	1.270000e+03	5
50%	2.000000e+00	3.609861e+01	-9.241808e+01	3.609799e+01	-9.241772e+01	2.440000e-01	4.007000e+03	(
75%	2.000000e+00	4.016024e+01	-8.037243e+01	4.016105e+01	-8.037338e+01	7.640000e-01	9.567000e+03	1
max	4.000000e+00	4.900058e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	1.551860e+02	9.999997e+06	1

```
df.isna().sum()
```

```
ID                        0
Severity                  0
Start_Time                0
End_Time                  0
Start_Lat                 0
Start_Lng                 0
End_Lat                   0
End_Lng                   0
Distance(mi)              0
Description               0
Number              1743911
Street                    2
Side                      0
City                    137
County                    0
State                     0
Zipcode                1319
Country                   0
Timezone               3659
Airport_Code           9549
```

```
Weather_Timestamp          50736
Temperature(F)             69274
Wind_Chill(F)             469643
Humidity(%)                73092
Pressure(in)               59200
Visibility(mi)             70546
Wind_Direction             73775
Wind_Speed(mph)           157944
Precipitation(in)         549458
Weather_Condition          70636
Amenity                       0
Bump                          0
Crossing                      0
Give_Way                      0
Junction                      0
No_Exit                       0
Railway                       0
Roundabout                    0
Station                       0
Stop                          0
Traffic_Calming               0
Traffic_Signal                0
Turning_Loop                  0
Sunrise_Sunset             2867
Civil_Twilight             2867
Nautical_Twilight          2867
Astronomical_Twilight      2867
dtype: int64
```

Columns which we will analyze

- City
- Start Time
- Start LAT,LNG
- Weather condition

```
df.columns
```

```
Index(['ID', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Number', 'Street',
       'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone',
       'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
       'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction',
       'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
```

```
       'Astronomical_Twilight'],
      dtype='object')
```

CITY

```
df.City
```

```
0               Dublin
1               Dayton
2           Cincinnati
3                Akron
4           Cincinnati
              ...
2845337      Riverside
2845338      San Diego
2845339         Orange
2845340    Culver City
2845341       Highland
Name: City, Length: 2845342, dtype: object
```

```
cities=df.City.unique()
cities[:99]
```

```
array(['Dublin', 'Dayton', 'Cincinnati', 'Akron', 'Williamsburg',
       'Cleveland', 'Lima', 'Westerville', 'Jamestown', 'Freeport',
       'Columbus', 'Toledo', 'Roanoke', 'Ft Mitchell', 'Edinburgh',
       'Fairborn', 'Shelbyville', 'Greensburg', 'Saint Paul',
       'Parkersburg', 'Indianapolis', 'Dundee', 'Jeffersonville',
       'Pittsburgh', 'Lewis Center', 'Dunkirk', 'Redkey', 'Milton',
       'Willshire', 'Straughn', 'Cambridge Springs', 'Fremont',
       'Louisville', 'South Charleston', 'Edinboro', 'Buckhannon',
       'Lockbourne', 'Painesville', 'Washington', 'Dunbar', 'Angola',
       'Edon', 'Medina', 'De Mossville', 'New Albany', 'Charleston',
       'Fort Wayne', 'Burnsville', 'Bedford', 'Clarksville', 'Lakewood',
       'Richfield', 'Sewickley', 'Independence', 'Westlake', 'Erlanger',
       'Grove City', 'Monroe', 'West Middlesex', 'Gaston', 'Economy',
       'Fairmount', 'Hagerstown', 'Walton', 'Crittenden', 'Coraopolis',
       'Holland', 'Greenfield', 'Anderson', 'Englewood', 'Knightstown',
       'Bentleyville', 'Memphis', 'Henryville', 'Kendallville', 'Avilla',
       'Ohio City', 'Van Wert', 'Rocky River', 'Sturgis', 'West Chester',
       'Orient', 'Madison', 'Deputy', 'Keystone', 'Mercer', 'Bryant',
       'Pennville', 'Kimbolton', 'Thornville', 'Wexford', 'Fishers',
       'Noblesville', 'Macedonia', 'Youngstown', 'Fairdale', 'Sutton',
       'Mount Sterling', 'Northwood'], dtype=object)
```

```
Cities_by_accident=df.City.value_counts()
Cities_by_accident
```

```
Miami                      106966
Los Angeles                 68956
```

```
Orlando                          54691
Dallas                           41979
Houston                          39448
                                   ...
Ridgedale                            1
Sekiu                                1
Wooldridge                           1
Bullock                              1
American Fork-Pleasant Grove         1
Name: City, Length: 11681, dtype: int64
```

```
Cities_by_accident[:9]
```

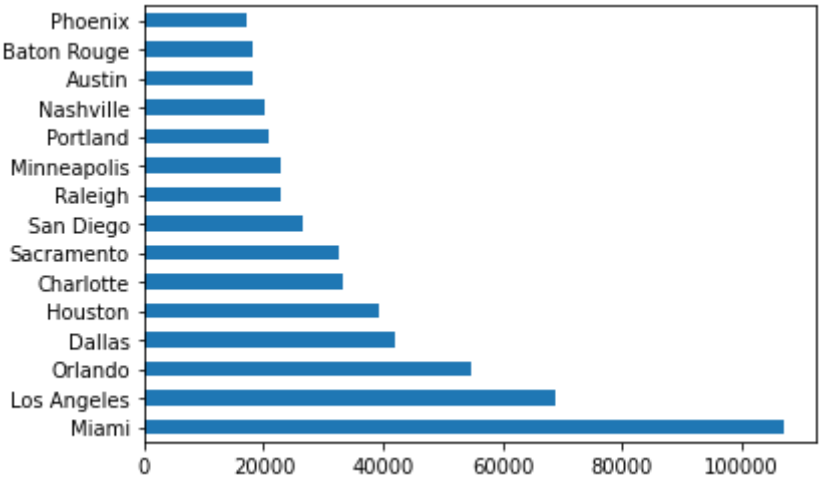
```
Miami          106966
Los Angeles     68956
Orlando         54691
Dallas          41979
Houston         39448
Charlotte       33152
Sacramento      32559
San Diego       26627
Raleigh         22840
Name: City, dtype: int64
```

```
'NY' in df.State
```

```
False
```

```
Cities_by_accident[:15].plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf972e9990>
```



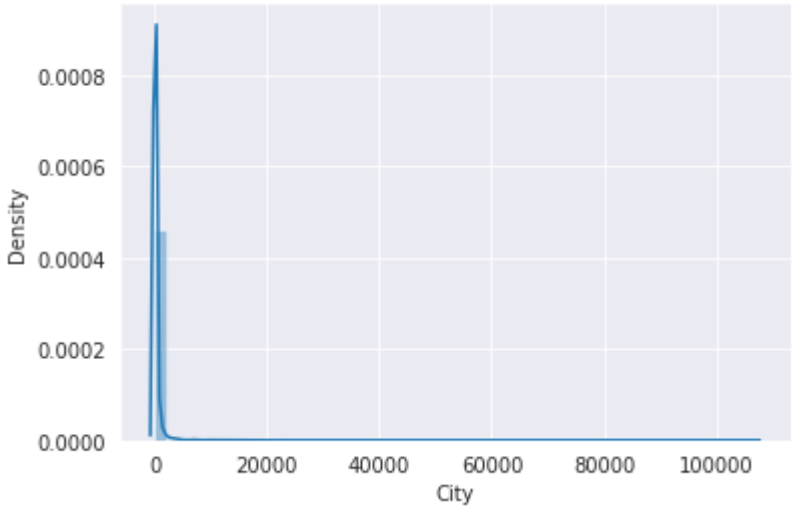

```
import seaborn as sns
```

```
sns.set_style("darkgrid")
```

```
sns.distplot(Cities_by_accident)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf8824d5d0>
```




```
high_accident_cities=Cities_by_accident[Cities_by_accident>=1000]
low_accident_cities=Cities_by_accident[Cities_by_accident<1000]
```

```
len(high_accident_cities)/len(Cities_by_accident)
```

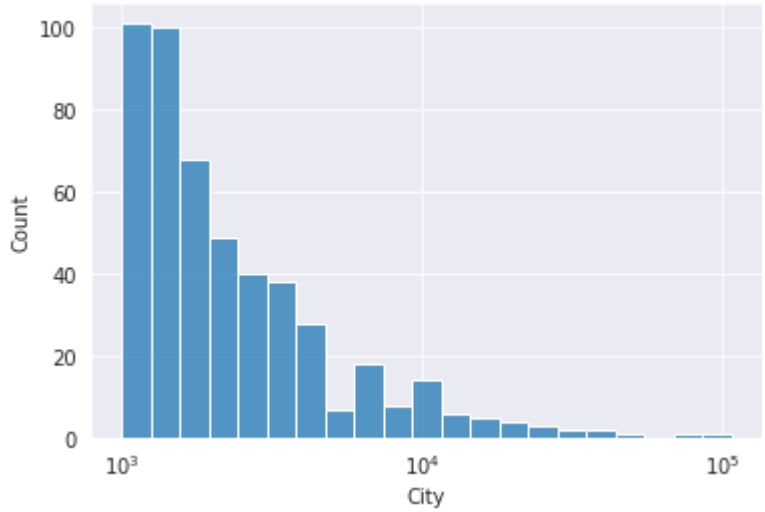
```
0.04246211796935194
```

```
len(low_accident_cities)/len(Cities_by_accident)
```

```
0.957537882030648
```

```
sns.histplot(high_accident_cities,log_scale=True)
```

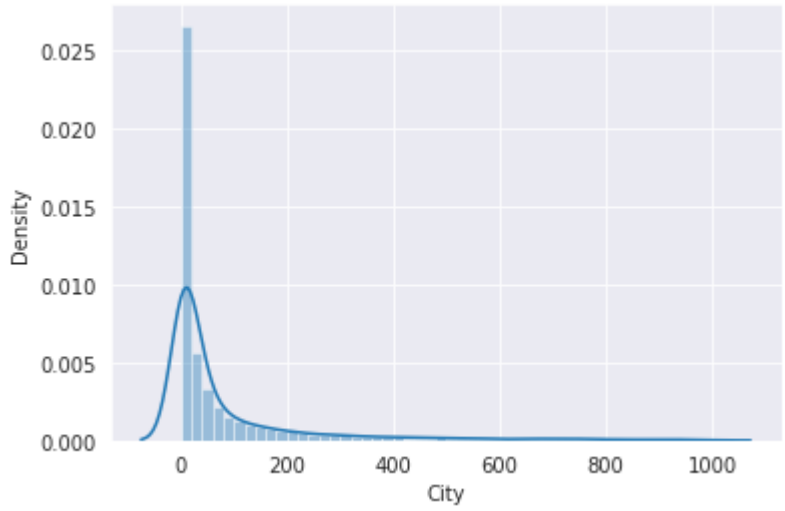
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf83640d10>
```




```
sns.distplot(low_accident_cities)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf837e49d0>
```



START TIME

```
df.Start_Time=pd.to_datetime(df.Start_Time)
```

```
df.Start_Time[0]
```

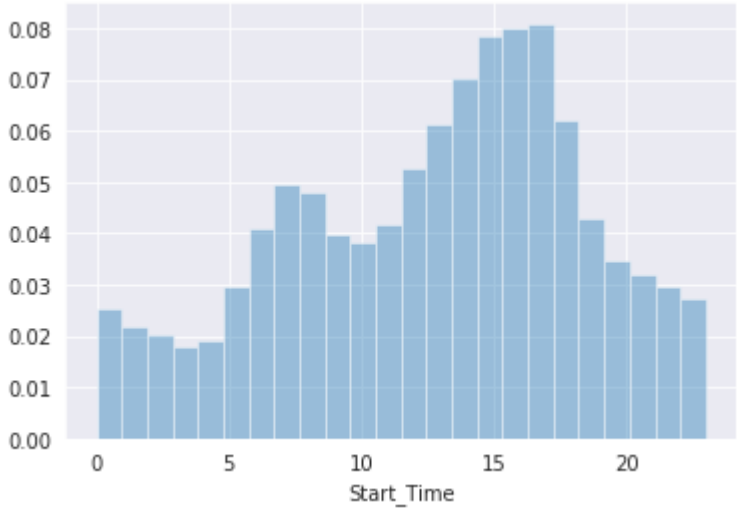
```
Timestamp('2016-02-08 00:37:08')
```

```
sns.distplot(df.Start_Time.dt.hour ,bins=24,kde=False,norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please
```

```
adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf830182d0>
```

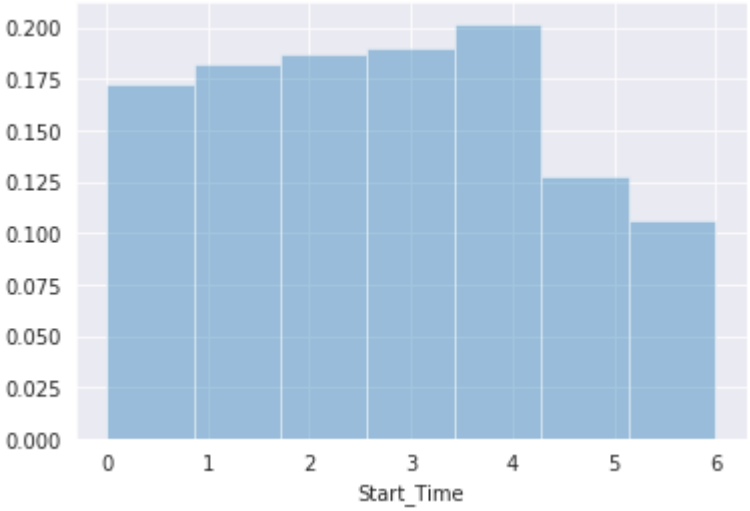


High percentage of accidents berween 3-6 pm proobably people returning from work Second Highest percantage is between 6-10 am

```
sns.distplot(df.Start_Time.dt.dayofweek ,bins=7,kde=False,norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf82ea8b50>
```



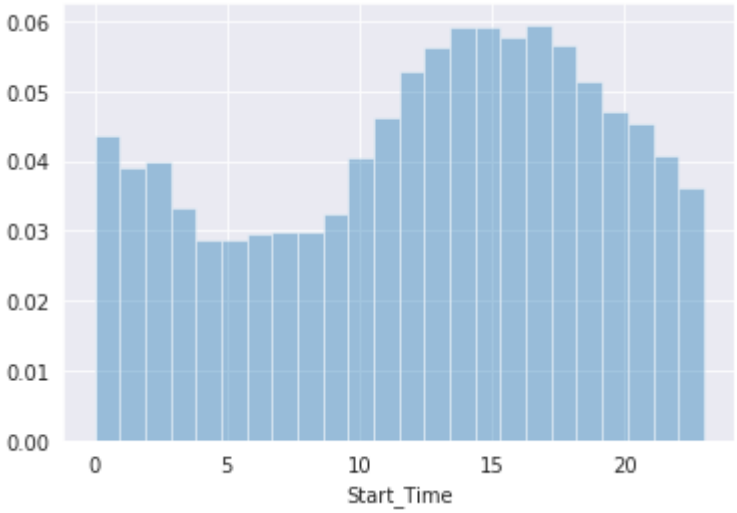
Most accident occur on wednesday and thursday

Is Distribution of work same on weekends as weekdays

```
sunday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==6]
sns.distplot(sunday_start_time.dt.hour ,bins=24,kde=False,norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

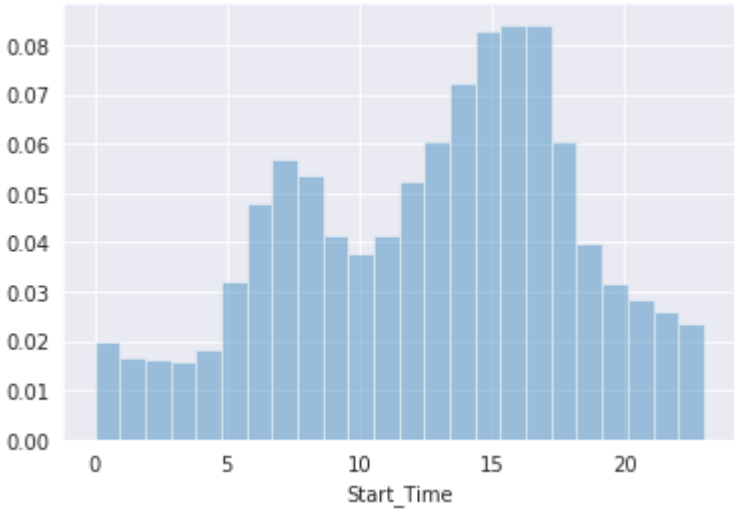
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf82ff5790>
```




```
monday_start_time=df.Start_Time[df.Start_Time.dt.dayofweek==0]
sns.distplot(monday_start_time.dt.hour ,bins=24,kde=False,norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf82ca97d0>
```



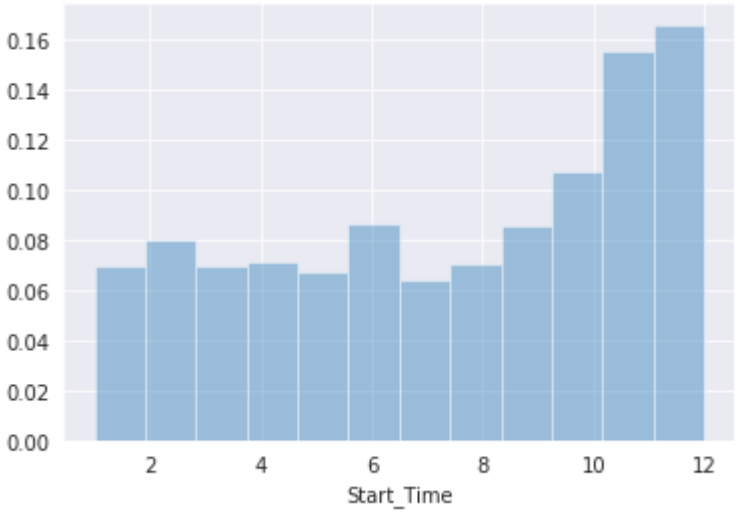
On sundays peak occur between 9am-3pm unllike weekdays

```
sns.distplot(monday_start_time.dt.month ,bins=12,kde=False,norm_hist=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
```

```
`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf82bc1c50>
```

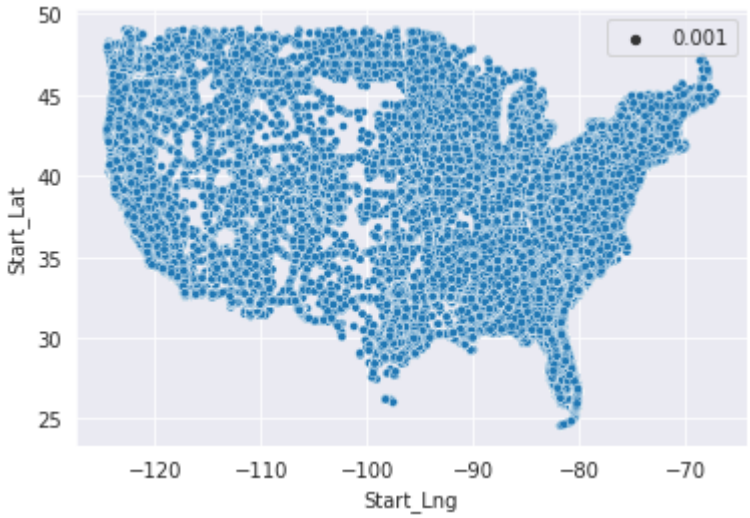


More accident occur during Winter time .

START LATITUDE AND START LONGITUDE

```
sns.scatterplot(x=df.Start_Lng,y=df.Start_Lat,size=.001)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf82b0a290>
```




```
import folium
```

```
lat,lon=df.Start_Lat[0],df.Start_Lng[0]
lat,lon
```

```
(40.10891, -83.09286)
```

```
for x in df[['Start_Lat', 'Start_Lng']].sample(100).iteritems():
    print(x[1])
```

```
2183265    34.059483
1765606    43.016892
369112     25.944274
1460345    42.946863
2382617    34.165503
             ...
2591041    45.787723
37288      41.390110
1968005    25.938449
15562      38.872220
1504751    38.956841
Name: Start_Lat, Length: 100, dtype: float64
2183265   -118.001377
1765606    -78.256806
369112     -80.307124
1460345    -85.659090
2382617   -118.492347
              ...
2591041   -118.146677
37288      -88.192740
1968005    -80.188624
15562     -104.719120
1504751    -76.864554
Name: Start_Lng, Length: 100, dtype: float64
```

```
zip(list(df.Start_Lat), list(df.Start_Lng))
```

```
<zip at 0x7fdf80f7c050>
```

```
from folium.plugins import HeatMap
sample_df = df.sample(int(0.001 * len(df)))
lat_lon_pairs = list(zip(list(sample_df.Start_Lat), list(sample_df.Start_Lng)))
map = folium.Map()
HeatMap(lat_lon_pairs).add_to(map)
map
```

Make this Notebook Trusted to load map: File -> Trust Notebook

ASK AND ANSWER QUESTION

- Are there accidents in warmer or colder region
- WHich 5 states have more number of accidents
- Does New york Show up in data
- Among top 100 cities which state has most number of accidents
- What time of day accident frequently occur
- On which day accidents mostly occur?
- WHich months have most accident
- List item
- Trends of accidents year over year
-

SUMMARY AND CONCLUSION

- No data for new york
- Less than 5% cities have more than 1000 accidents
- No of accidents per city has decreased exponentially
- List item
- Over 1200 cities reported just 1 accident
- List item

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/simar4447/us-accident-eda
```

```
'https://jovian.ai/simar4447/us-accident-eda'
```