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

```
df = pd.read_csv("uber.csv")
```

```
df.head()
df.info() #To get the required information of the dataset
df.columns #TO get number of columns in the dataset
df = df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed c
df.head()
df.shape #To get the total (Rows,Columns)
df.dtypes #To get the type of each column
df.info()
df.describe() #To get statistics of each columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8918 entries, 0 to 8917
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         8918 non-null   int64  
 1   key                8918 non-null   object 
 2   fare_amount        8917 non-null   float64
 3   pickup_datetime    8917 non-null   object 
 4   pickup_longitude   8917 non-null   float64
 5   pickup_latitude    8917 non-null   float64
 6   dropoff_longitude  8917 non-null   float64
 7   dropoff_latitude   8917 non-null   float64
 8   passenger_count    8917 non-null   float64
dtypes: float64(6), int64(1), object(2)
memory usage: 627.2+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8918 entries, 0 to 8917
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   fare_amount        8917 non-null   float64
 1   pickup_datetime    8917 non-null   object 
 2   pickup_longitude   8917 non-null   float64
 3   pickup_latitude    8917 non-null   float64
 4   dropoff_longitude  8917 non-null   float64
 5   dropoff_latitude   8917 non-null   float64
 6   passenger_count    8917 non-null   float64
 6   passenger_count    8917 non-null   float64
dtypes: float64(6), object(1)
memory usage: 487.8+ KB
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
count	8917.00000	8917.000000	8917.000000	8917.000000	8917.
mean	11.44521	-72.700539	40.000828	-72.673399	40.
std	10.41732	12.320446	5.895643	9.855862	5.
min	2.50000	-748.016667	-74.009697	-75.350437	-73.
25%	6.00000	-73.992015	40.734997	-73.991472	40.
50%	8.50000	-73.981582	40.752407	-73.979908	40.
75%	12.50000	-73.967155	40.767058	-73.963588	40.
max	350.00000	40.770667	41.366138	40.761672	41.

```
df.isnull().sum()
```

```
fare_amount          1
pickup_datetime      1
pickup_longitude     1
pickup_latitude      1
dropoff_longitude    1
dropoff_latitude     1
```

```
passenger_count      1
dtype: int64
```

+ Code + Text

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].me
```

```
df.isnull().sum()
```

```
fare_amount          1
pickup_datetime      1
pickup_longitude     1
pickup_latitude      1
dropoff_longitude    0
dropoff_latitude     0
passenger_count      1
dtype: int64
```

```
df.dtypes
```

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
```

```
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
dtype: object
```

```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors=
df.dtypes
```

```
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
passenger_count                  float64
dtype: object
```

```
df= df.assign(hour = df.pickup_datetime.dt.hour,
day= df.pickup_datetime.dt.day,
month = df.pickup_datetime.dt.month,
year = df.pickup_datetime.dt.year,
dayofweek = df.pickup_datetime.dt.dayofweek)
df.head()
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitud
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.99951
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.99471
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.96256

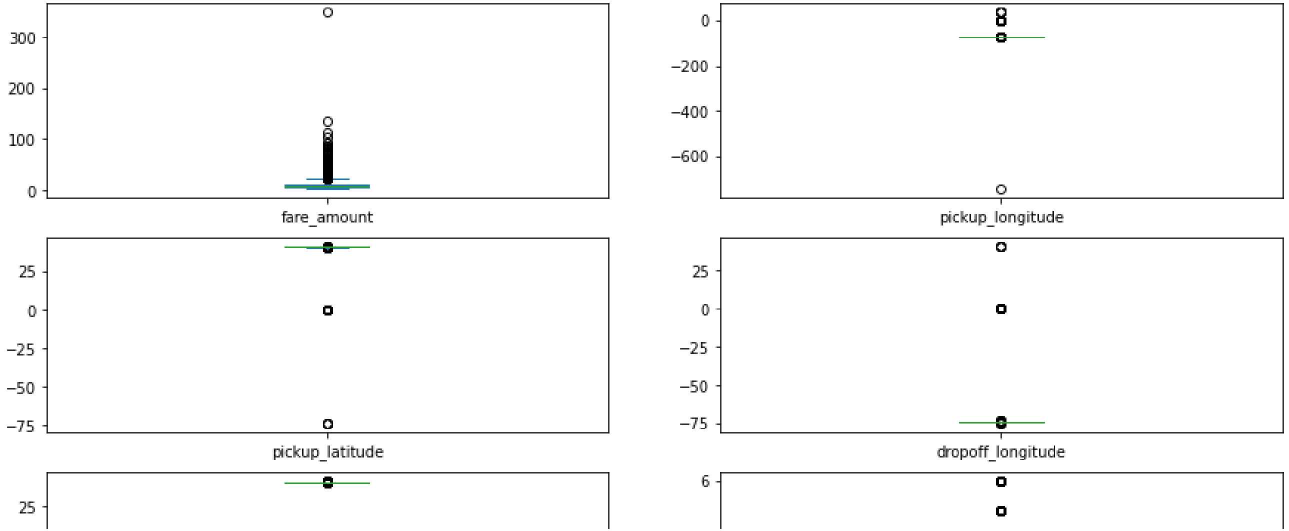
```
df = df.drop('pickup_datetime',axis=1)
```

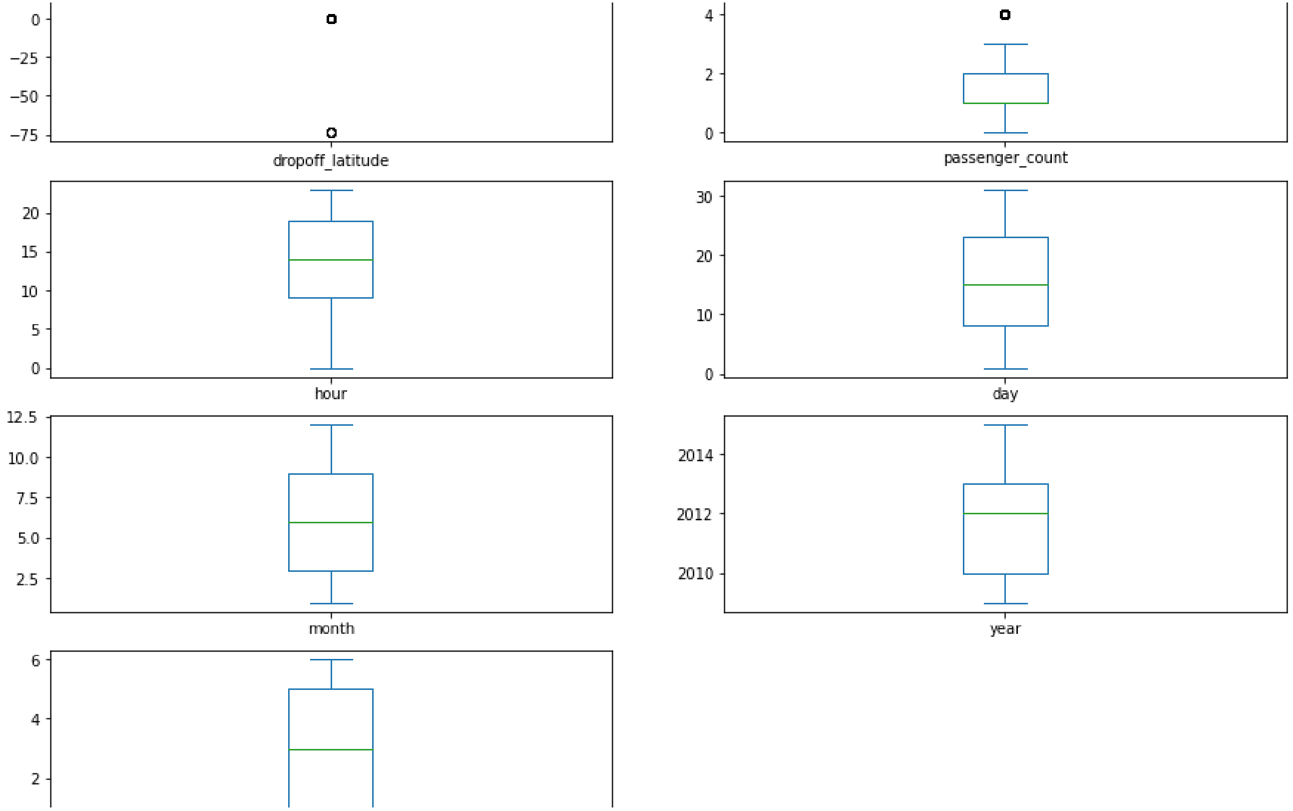
```
df.dtypes
```

```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
```

```
hour                    float64
day                     float64
month                   float64
year                    float64
dayofweek               float64
dtype: object
```

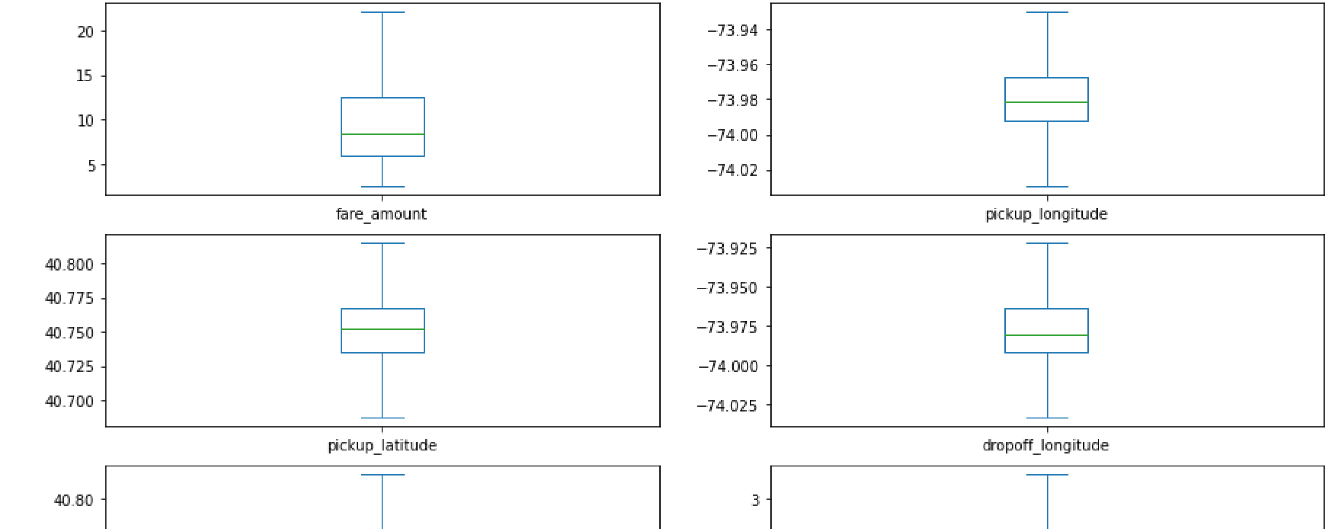
```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15
fare_amount            AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude    AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude        AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude   AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude       AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count     AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                   AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                 AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                  AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek              AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```

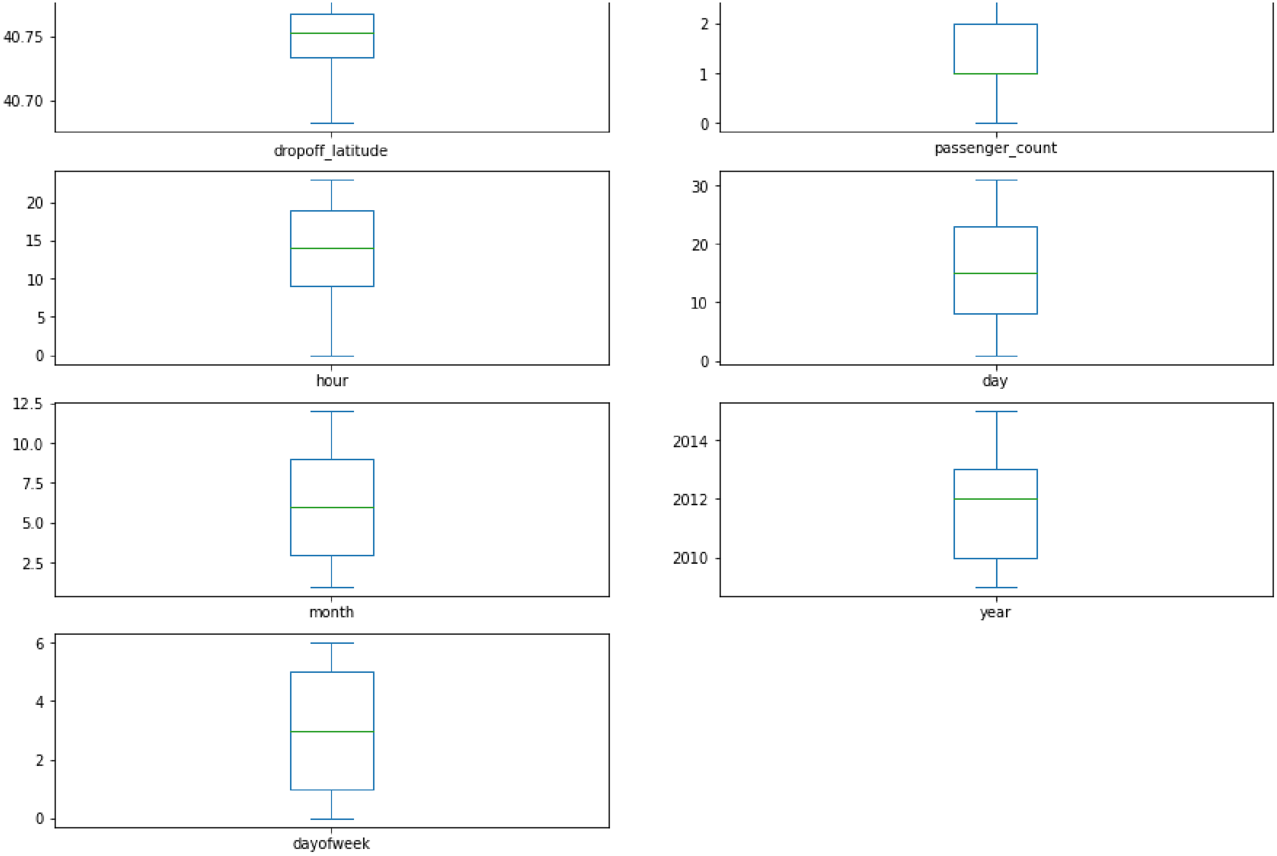




```
def remove_outlier(df1 , col):
 Q1 = df1[col].quantile(0.25)
 Q3 = df1[col].quantile(0.75)
 IQR = Q3 - Q1
 lower_whisker = Q1-1.5*IQR
 upper_whisker = Q3+1.5*IQR
 df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
 return df1


def treat_outliers_all(df1 , col_list):
 for c in col_list:
     df1 = remove_outlier(df , c)
 return df1
df = treat_outliers_all(df , df.iloc[: , 0::])
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15
fare_amount             AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude     AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude         AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude    AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude        AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count      AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                    AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                  AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                   AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                 AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek               AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```




40.75
40.70
dropoff_latitude
2
1
0
passenger_count
20
15
10
5
0
hour
30
20
10
0
day
12.5
10.0
7.5
5.0
2.5
month
2014
2012
2010
year
6
4
2
0
dayofweek

```
pip install haversine
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting haversine
  Downloading haversine-2.7.0-py2.py3-none-any.whl (6.9 kB)
Installing collected packages: haversine
Successfully installed haversine-2.7.0
```

```
import haversine as hs
```

```
travel_dist = []
for pos in range(len(df['pickup_longitude'])):
    long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['
    loc1=(lati1,long1)
    loc2=(lati2,long2)
```

```
        c = hs.haversine(loc1,loc2)
        travel_dist.append(c)
    print(travel_dist)
    df['dist_travel_km'] = travel_dist
    df.head()
```

```
[1.6833250775073447, 2.4575932783467835, 5.036384146783453, 1.661685753650294, 4.110
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitu
0	7.5	-73.999817	40.738354	-73.999512	40.7232
1	7.7	-73.994355	40.728225	-73.994710	40.7503
2	12.9	-74.005043	40.740770	-73.962565	40.7726
3	5.3	-73.976124	40.790844	-73.965316	40.8033
4	16.0	-73.929865	40.744085	-73.973082	40.7612

```
df= df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130
print("Remaining observastions in the dataset:", df.shape)
```

```
    Remaining observastions in the dataset: (8917, 12)
```

```
incorrect_coordinates = df.loc[(df.pickup_latitude>90)|(df.pick
 (df.dropoff_latitude>90)|(df.dropoff_latitude<-90)|
 (df.pickup_longitude>180)|(df.pickup_longitude<-180)|
 (df.dropoff_longitude>90)|(df.dropoff_longitude<-90)
 ]

df.drop(incorrect_coordinates, inplace = True, errors = 'ignore
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
df.drop(incorrect_coordinates, inplace = True, errors = 'ignore
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  errors=errors,
```

```
df.head()
```

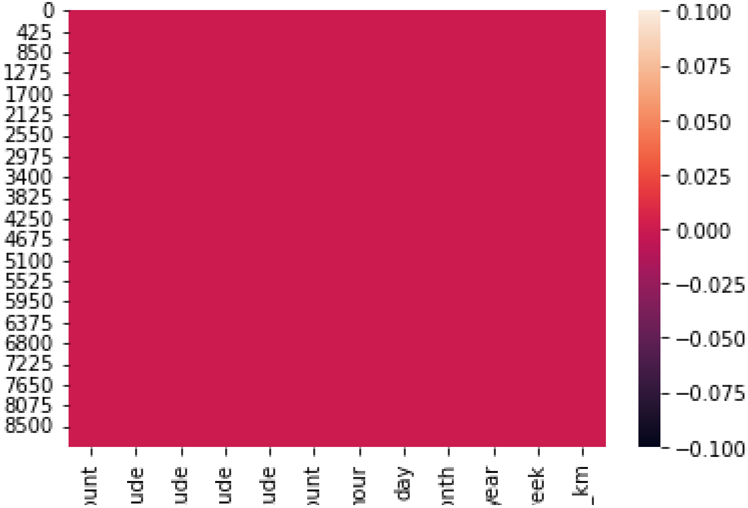
fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitu

```
df.isnull().sum()
```

```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
hour                 0
day                  0
month                0
year                 0
dayofweek            0
dist_travel_km       0
dtype: int64
```

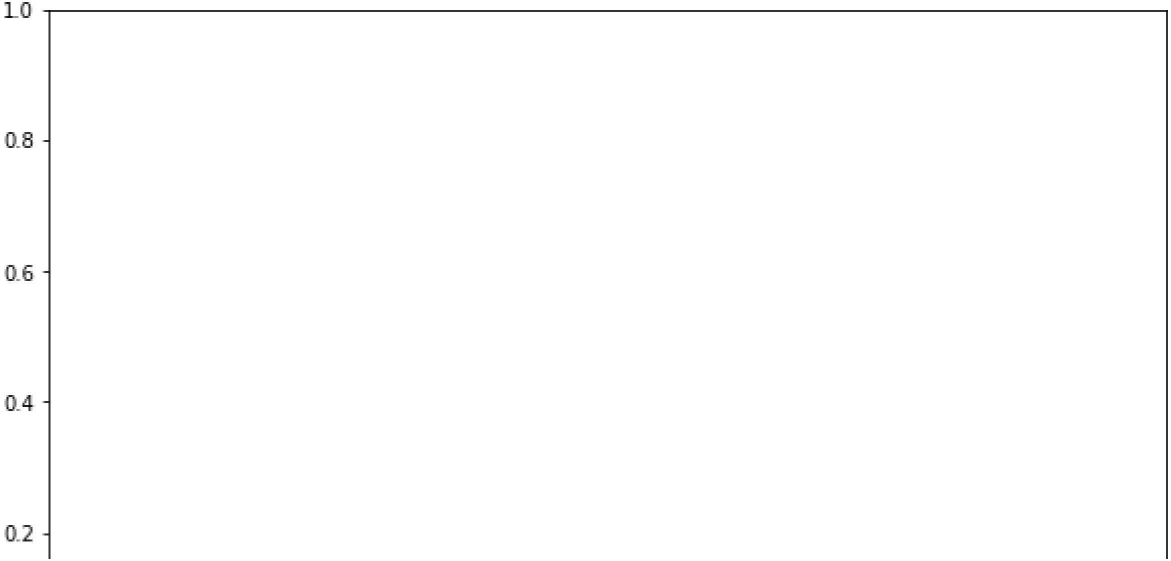
```
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ab5cbc6d0>
```



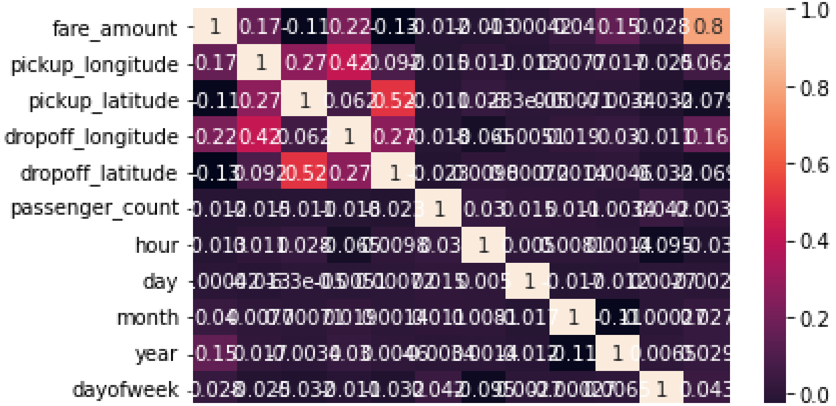
```
corr=df.corr() #function to find corelation
```

```
fig,axis = pl.subplots(figsize = (10,6))
```



```
sns.heatmap(df.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ab3216b10>
```



```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude
y = df['fare_amount']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size
```

```
from sklearn.linear_model import LinearRegression
```

```
regression = LinearRegression()
regression.fit(X_train,y_train)
regression.coef_ #To find the linear coeeficient
regression.intercept_ #To find the linear intercept
prediction = regression.predict(X_test) #To predict the target
print(prediction)
y_test
```

```
[ 8.15350341 11.45372045 20.88795037 ...  4.57442819  8.94565272
  6.08170728]
3731     8.10
8472    22.25
8255    21.70
8776     7.30
5638     5.30
        ...  
7491     5.50
3294     6.90
1833     3.30
6797     9.30
481      5.30
Name: fare_amount, Length: 2943, dtype: float64
```

```
regression = LinearRegression()
regression.fit(X_train,y_train)
regression.coef_ #To find the linear coeeficient
regression.intercept_ #To find the linear intercept
prediction = regression.predict(X_test) #To predict the target
print(prediction)
y_test
```

```
[ 8.15350341 11.45372045 20.88795037 ...  4.57442819  8.94565272
  6.08170728]
3731     8.10
8472    22.25
8255    21.70
8776     7.30
5638     5.30
        ...  
7491     5.50
3294     6.90
1833     3.30
6797     9.30
481      5.30
Name: fare_amount, Length: 2943, dtype: float64
```

```
from sklearn.metrics import r2_score
```

```
r2_score(y_test,prediction)
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
MSE
RMSE = np.sqrt(MSE)
RMSE
```

```
    3.0423611997128597
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf = RandomForestRegressor(n_estimators=100)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
y_pred
```

```
    array([ 8.099, 13.686, 21.704, ...,  4.451,  8.822,  4.039])
```

```
R2_Random = r2_score(y_test,y_pred)
R2_Random
```

```
0.7917656690094285
```

```
MSE_Random = mean_squared_error(y_test,y_pred)
MSE_Random
```

```
6.286907173010533
```

```
RMSE_Random = np.sqrt(MSE_Random)
RMSE_Random
```

```
2.507370569543029
```