In [1]:

```
import pandas as pd
```

In [2]:

```
df=pd.read_csv('C:/shubhangi/2023-24/LP-III_ML/Assignment 1/uber.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5

In [4]:

```
df=df.drop(['Unnamed: 0','key','pickup_datetime'],axis=1)
```

In [5]:

```
df.shape
```

Out[5]:

```
(200000, 6)
```

In [6]:

```
df.dtypes
```

Out[6]:

```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [7]:

```
set(df.dtypes)
```

Out[7]:

```
{dtype('int64'), dtype('float64')}
```

In [8]:

```
df.dropna()
```

Out[8]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...
199995	3.0	-73.987042	40.739367	-73.986525	40.740297	1
199996	7.5	-73.984722	40.736837	-74.006672	40.739620	1
199997	30.9	-73.986017	40.756487	-73.858957	40.692588	2
199998	14.5	-73.997124	40.725452	-73.983215	40.695415	1
199999	14.1	-73.984395	40.720077	-73.985508	40.768793	1

199999 rows × 6 columns

In [9]:

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

Out[9]:

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

In [10]:

```
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].median(),inplace=True)
```

In [11]:

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inplace=True)
```

In [12]:

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

Out[12]:

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

In [13]:

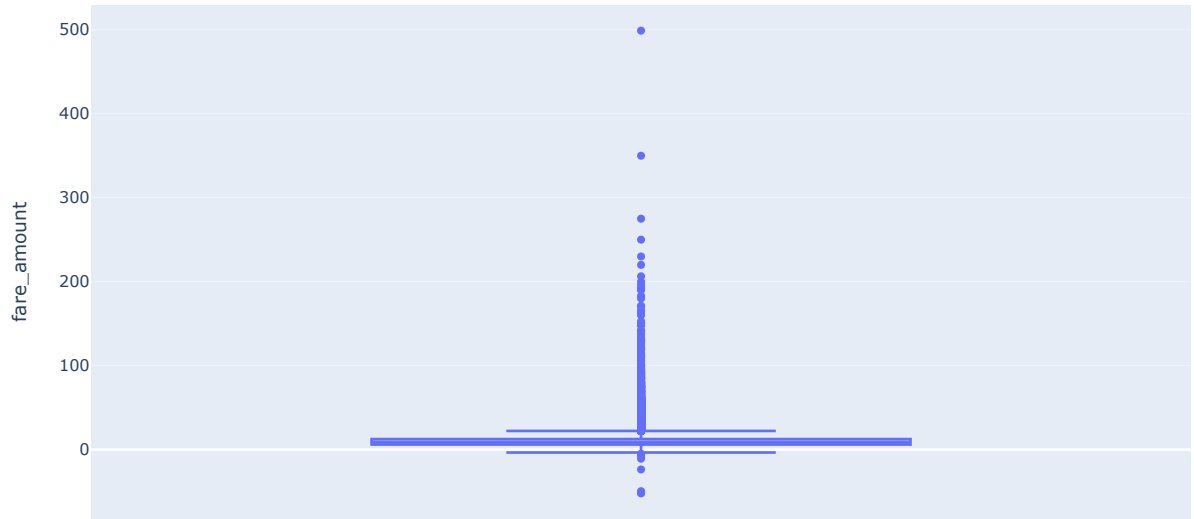
```
import plotly.express as px
```

In [14]:

```
fig=px.box(df,y='fare_amount')
```

In [15]:

```
fig.show()
```



In [16]:

```
x=df.drop(['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude'],axis=1)
```

In [17]:

```
df.describe()[['fare_amount', 'passenger_count']]
```

Out[17]:

	fare_amount	passenger_count
count	200000.000000	200000.000000
mean	11.359955	1.684535
std	9.901776	1.385997
min	-52.000000	0.000000
25%	6.000000	1.000000
50%	8.500000	1.000000
75%	12.500000	2.000000
max	499.000000	208.000000

In [47]:

```
import numpy as np
```

In [48]:

```
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
```

In [49]:

```
def treat_outliers_all(df1 , col_list):
    for c in col_list:
        df1 = remove_outlier(df , c)
    return df1
```

In [50]:

```
df = treat_outliers_all(df , df.iloc[: , 0::])
```

In [52]:

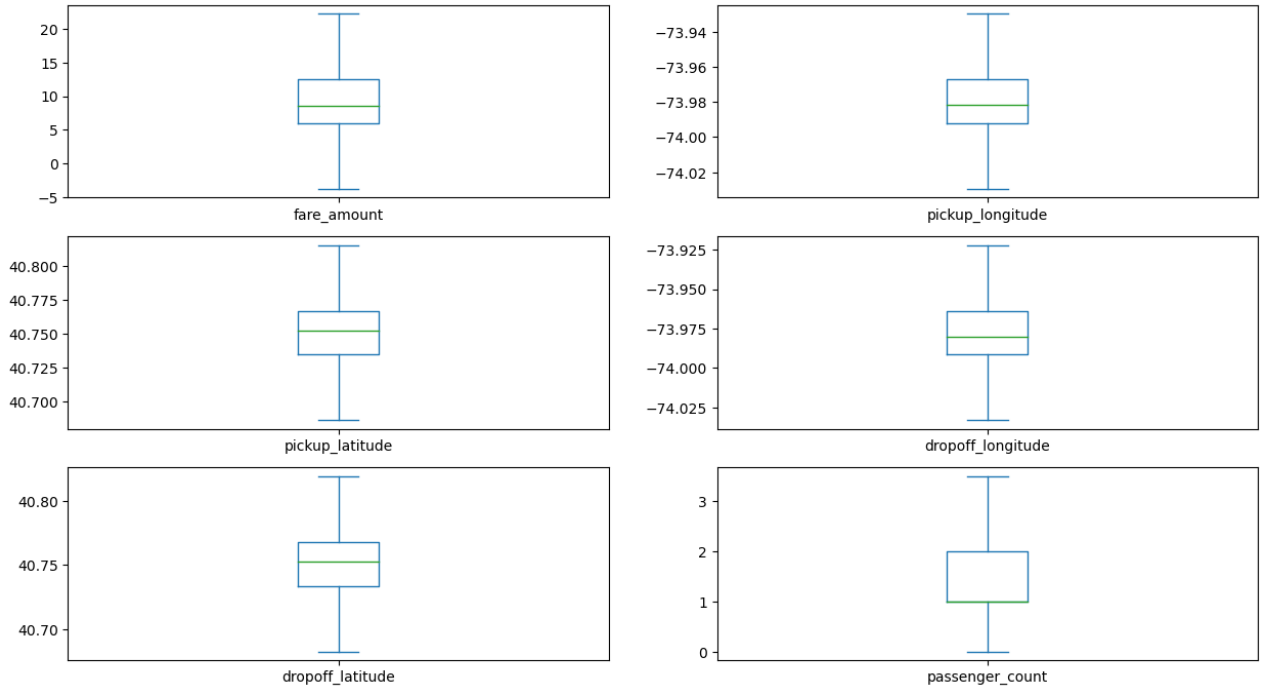
```
import matplotlib.pyplot as plt
```

In [53]:

```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
```

Out[53]:

```
fare_amount            Axes(0.125,0.786098;0.352273x0.0939024)
pickup_longitude     Axes(0.547727,0.786098;0.352273x0.0939024)
pickup_latitude        Axes(0.125,0.673415;0.352273x0.0939024)
dropoff_longitude    Axes(0.547727,0.673415;0.352273x0.0939024)
dropoff_latitude       Axes(0.125,0.560732;0.352273x0.0939024)
passenger_count      Axes(0.547727,0.560732;0.352273x0.0939024)
dtype: object
```



In [54]:

```
pip install haversine
```

```
Requirement already satisfied: haversine in c:\programdata\anaconda3\lib\site-packages (2.8.0)
Note: you may need to restart the kernel to use updated packages.
```

In [56]:

```
import haversine as hs
```

In [57]:

```
travel_dist = []
for pos in range(len(df['pickup_longitude'])):
        long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['pickup_latitude'][pos],df['dropoff_longitude'][pos],df['dropoff_latitu
        loc1=(lati1,long1)
        loc2=(lati2,long2)
        c = hs.haversine(loc1,loc2)
        travel_dist.append(c)
```

In [58]:

```
print(travel_dist)
df['dist_travel_km'] = travel_dist
df.head()
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Out[58]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	dist_travel_km
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	1.683325
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	2.457593
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	5.036384
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	1.661686
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	4.116088

In [59]:

```
#Uber doesn't travel over 130 kms so minimize the distance
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: (200000, 7)
```

In [60]:

```
0) and longitude (greater than or less than 180)
 |(df.pickup_latitude < -90)|(df.dropoff_latitude > 90) |(df.dropoff_latitude < -90) |(df.pickup_longitude > 180) |(df.pickup_longitude <
```

In [61]:

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

Out[61]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	dist_travel_km
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	1.683325
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	2.457593
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	5.036384
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	1.661686
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	4.116088

In [62]:

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

Out[62]:

```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dist_travel_km       0
dtype: int64
```

In [63]:

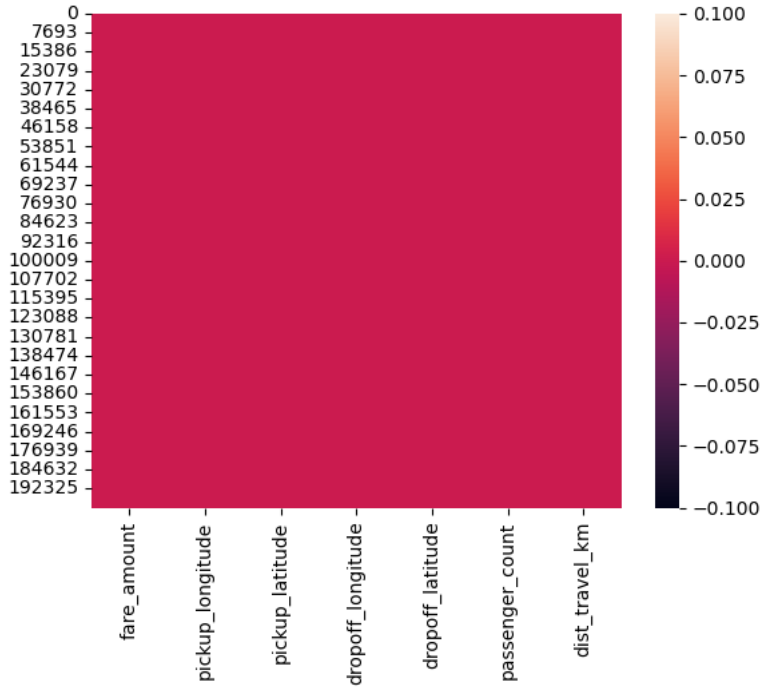
```
import seaborn as sns
```

In [64]:

```
sns.heatmap(df.isnull()) #Free for null values
```

Out[64]:

<Axes: >



In [65]:

```
corr = df.corr() #Function to find the correlation
print(corr)
```

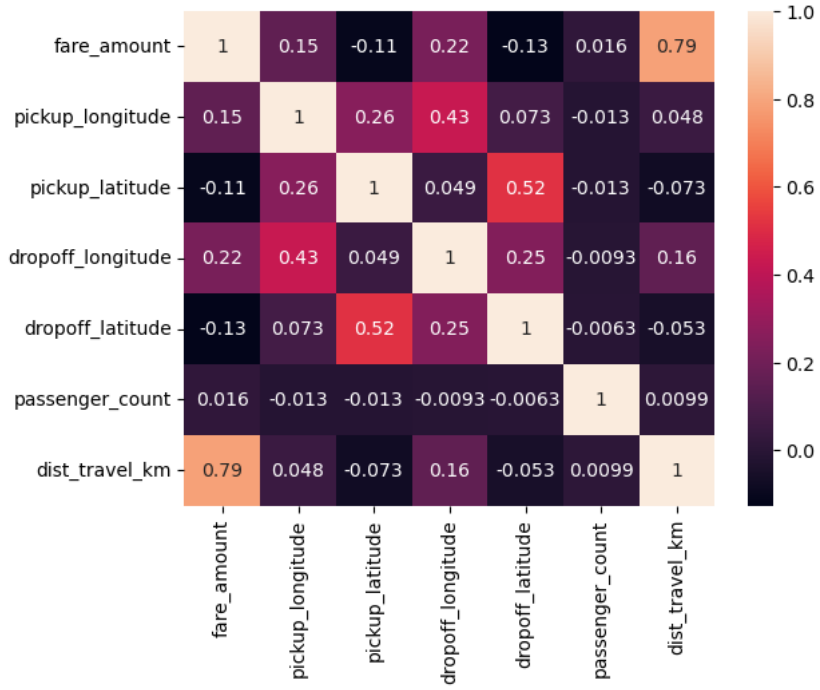
```
                  fare_amount  pickup_longitude  pickup_latitude  \
fare_amount          1.000000          0.154069        -0.110842
pickup_longitude     0.154069          1.000000         0.259497
pickup_latitude     -0.110842          0.259497         1.000000
dropoff_longitude    0.218675          0.425619         0.048889
dropoff_latitude    -0.125898          0.073290         0.515714
passenger_count      0.015778         -0.013213        -0.012889
dist_travel_km       0.786385          0.048446        -0.073362

                  dropoff_longitude  dropoff_latitude  passenger_count  \
fare_amount                0.218675         -0.125898         0.015778
pickup_longitude           0.425619          0.073290        -0.013213
pickup_latitude            0.048889          0.515714        -0.012889
dropoff_longitude          1.000000          0.245667        -0.009303
dropoff_latitude           0.245667          1.000000        -0.006308
passenger_count           -0.009303         -0.006308         1.000000
dist_travel_km             0.155191         -0.052701         0.009884

                  dist_travel_km
fare_amount             0.786385
pickup_longitude        0.048446
pickup_latitude        -0.073362
dropoff_longitude       0.155191
dropoff_latitude       -0.052701
passenger_count         0.009884
dist_travel_km          1.000000
```

In [66]:

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

Out[66]:

```
<Axes: >
```



In [67]:

```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_count','dist_travel_km']]

y = df['fare_amount']
```

In [68]:

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

In [69]:

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.33)
```

In [70]:

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
```

In [71]:

```
regression.fit(X_train,y_train)
```

Out[71]:

```
LinearRegression
LinearRegression()
```

In [72]:

```
regression.intercept_
```

Out[72]:

```
4461.8731571535045
```

In [73]:

```
regression.coef_
```

Out[73]:

```
array([ 26.29632195,  -7.60159329,  19.73368384, -18.21120668,
         0.05898655,   1.8490378 ])
```

In [74]:

```
prediction = regression.predict(X_test) #To predict the target values
print(prediction)
```

```
[ 6.49105246  6.92068004  5.82905968 ... 13.55261447  7.52776996
  7.4194044 ]
```

In [75]:

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

In [76]:

```
r2_score(y_test,prediction)
```

Out[76]:

```
0.6475045527243914
```

In [77]:

```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
print(MSE)
```

```
10.429294359791001
```

In [78]:

```
RMSE = np.sqrt(MSE)
print(RMSE)
```

```
3.229441803128058
```

In [79]:

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

In [80]:

```
rf = RandomForestRegressor(n_estimators=100)
```

In [81]:

```
rf.fit(X_train,y_train)
```

Out[81]:

```
RandomForestRegressor
RandomForestRegressor()
```

In [84]:

```
y_pred = rf.predict(X_test)
y_pred
```

Out[84]:

```
array([ 6.209,  6.919,  4.642, ..., 15.599,  8.569,  5.437])
```

In [85]:

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

Out[85]:

```
0.7612178302829902
```

In [86]:

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

In [87]:

```
print(MSE_Random)
```

```
7.064855887063792
```

In [88]:

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

```
2.657979662650524
```

In [89]:

```
print("OK")
```

```
OK
```

In [ ]:

```

```