In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_csv('uber.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [3]:
```
df.shape
```

Out[3]: (200000, 9)

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitu
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.7232
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.7503
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.7726
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.8033
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.7612

In [5]:
```
df.isnull()
```

Out[5]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
199995	False	False	False	False	False	False	False	False	
199996	False	False	False	False	False	False	False	False	
199997	False	False	False	False	False	False	False	False	
199998	False	False	False	False	False	False	False	False	
199999	False	False	False	False	False	False	False	False	

200000 rows × 9 columns

In [6]:
```
df.drop(columns=["Unnamed: 0", "key"], inplace=True)
df.head()
```

Out[6]:

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

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [8]:
```
df.fillna({
    'dropoff_latitude': df['dropoff_latitude'].mean(),
    'dropoff_longitude': df['dropoff_longitude'].median()
}, inplace=True)
```

In [9]:
```
df.dtypes
```

Out[9]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [10]:
```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime)
df.dtypes
```

Out[10]:
```
fare_amount                        float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                   float64
pickup_latitude                    float64
dropoff_longitude                  float64
dropoff_latitude                   float64
passenger_count                      int64
dtype: object
```

In [11]:
```
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)
```

In [12]:
```
df
```

Out[12]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hou
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1	1
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1	2
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1	2
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3	
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5	1
...	...	...	...	...	...	...	...	
199995	3.0	2012-10-28 10:49:00+00:00	-73.987042	40.739367	-73.986525	40.740297	1	1
199996	7.5	2014-03-14 01:09:00+00:00	-73.984722	40.736837	-74.006672	40.739620	1	
199997	30.9	2009-06-29 00:42:00+00:00	-73.986017	40.756487	-73.858957	40.692588	2	
199998	14.5	2015-05-20 14:56:25+00:00	-73.997124	40.725452	-73.983215	40.695415	1	1
199999	14.1	2010-05-15 04:08:00+00:00	-73.984395	40.720077	-73.985508	40.768793	1	

200000 rows × 12 columns

In [13]:
```
df = df.drop(["pickup_datetime"], axis =1)
df
```

Out[13]:

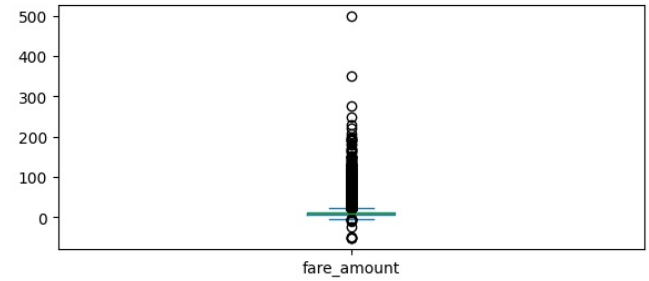
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	y
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1	19	7	5	2
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1	20	17	7	2
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1	21	24	8	2
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3	8	26	6	2
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5	17	28	8	2
...	...	...	...	...	...	...	...	...	...	
199995	3.0	-73.987042	40.739367	-73.986525	40.740297	1	10	28	10	2
199996	7.5	-73.984722	40.736837	-74.006672	40.739620	1	1	14	3	2
199997	30.9	-73.986017	40.756487	-73.858957	40.692588	2	0	29	6	2
199998	14.5	-73.997124	40.725452	-73.983215	40.695415	1	14	20	5	2
199999	14.1	-73.984395	40.720077	-73.985508	40.768793	1	4	15	5	2

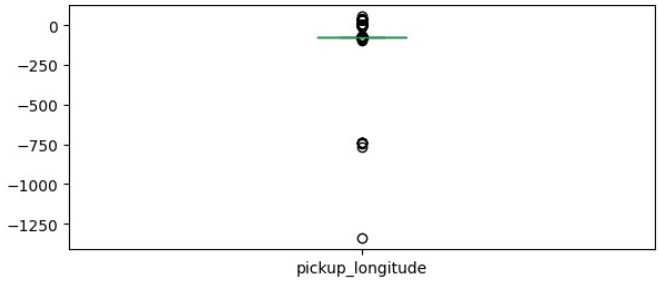
200000 rows × 11 columns

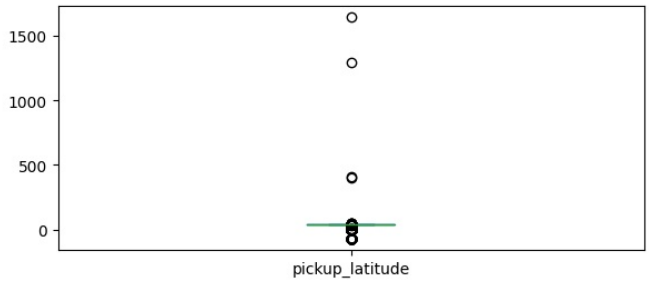
In [14]:
```
from math import *

def distance_formula(longitude1, latitude1, longitude2, latitude2):
    travel_dist = []

    for pos in range (len(longitude1)):
        lon1, lan1, lon2, lan2 = map(radians, [longitude1[pos], latitude1[pos], longitude2[pos], latitude2[pos]
        dist_lon = lon2 - lon1
        dist_lan = lan2 - lan1

        a = sin(dist_lan/2)**2 + cos(lan1) * cos(lan2) * sin(dist_lon/2)**2

        #radius of earth = 6371
        c = 2 * asin(sqrt(a)) * 6371
        travel_dist.append(c)

    return  travel_dist
```

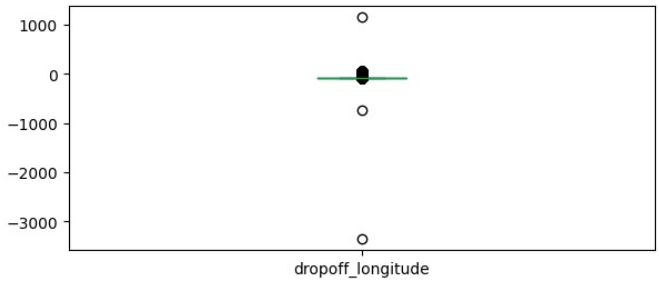
In [15]:
```
df['dist_travel_km'] = distance_formula(df.pickup_longitude.to_numpy(), df.pickup_latitude.to_numpy(), df.dropo
```

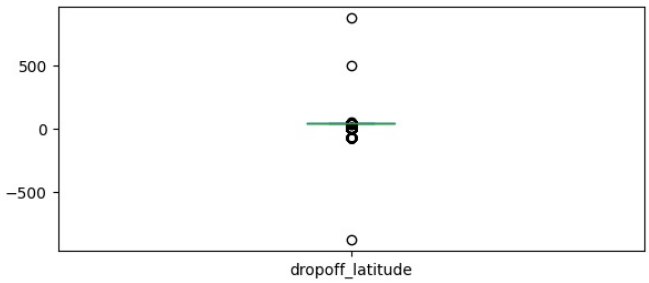
In [16]:
```
df.plot(kind = "box",subplots = True,layout = (6,2),figsize=(15,20)) #Boxplot to check the outliers
plt.show()
```

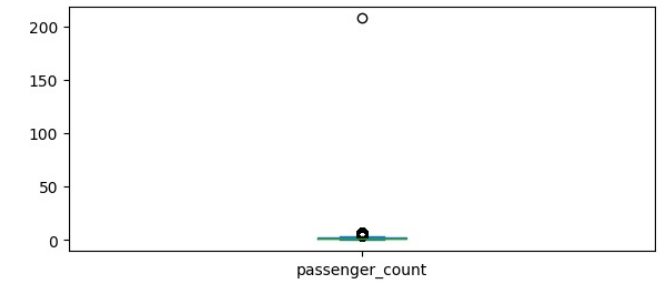


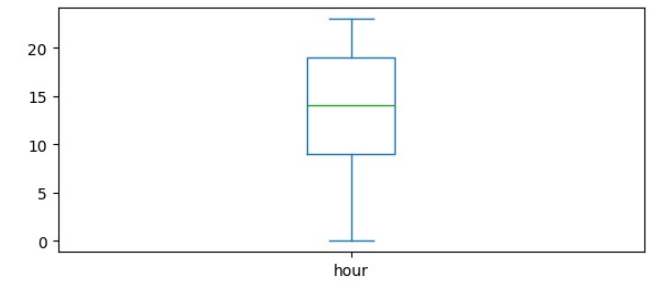


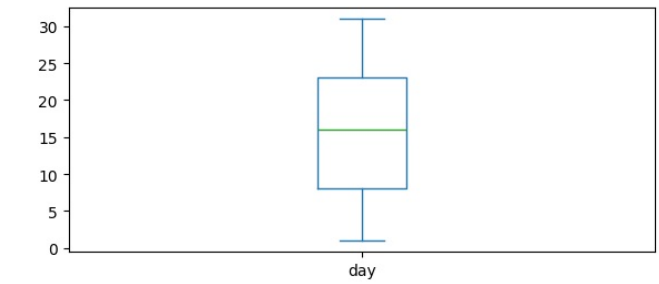


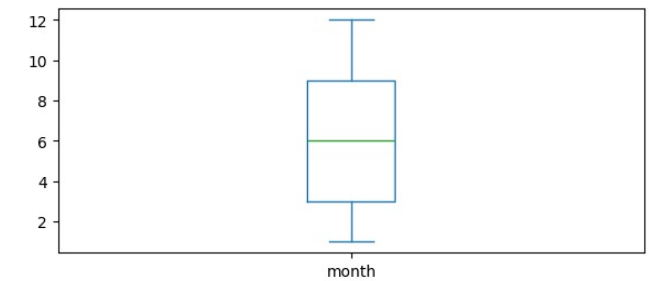


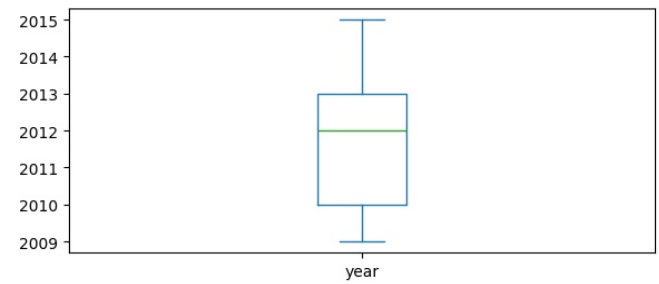


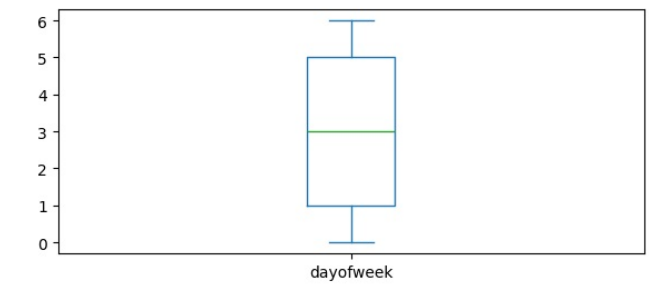


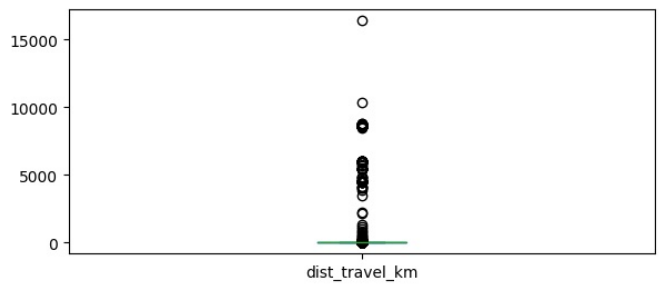








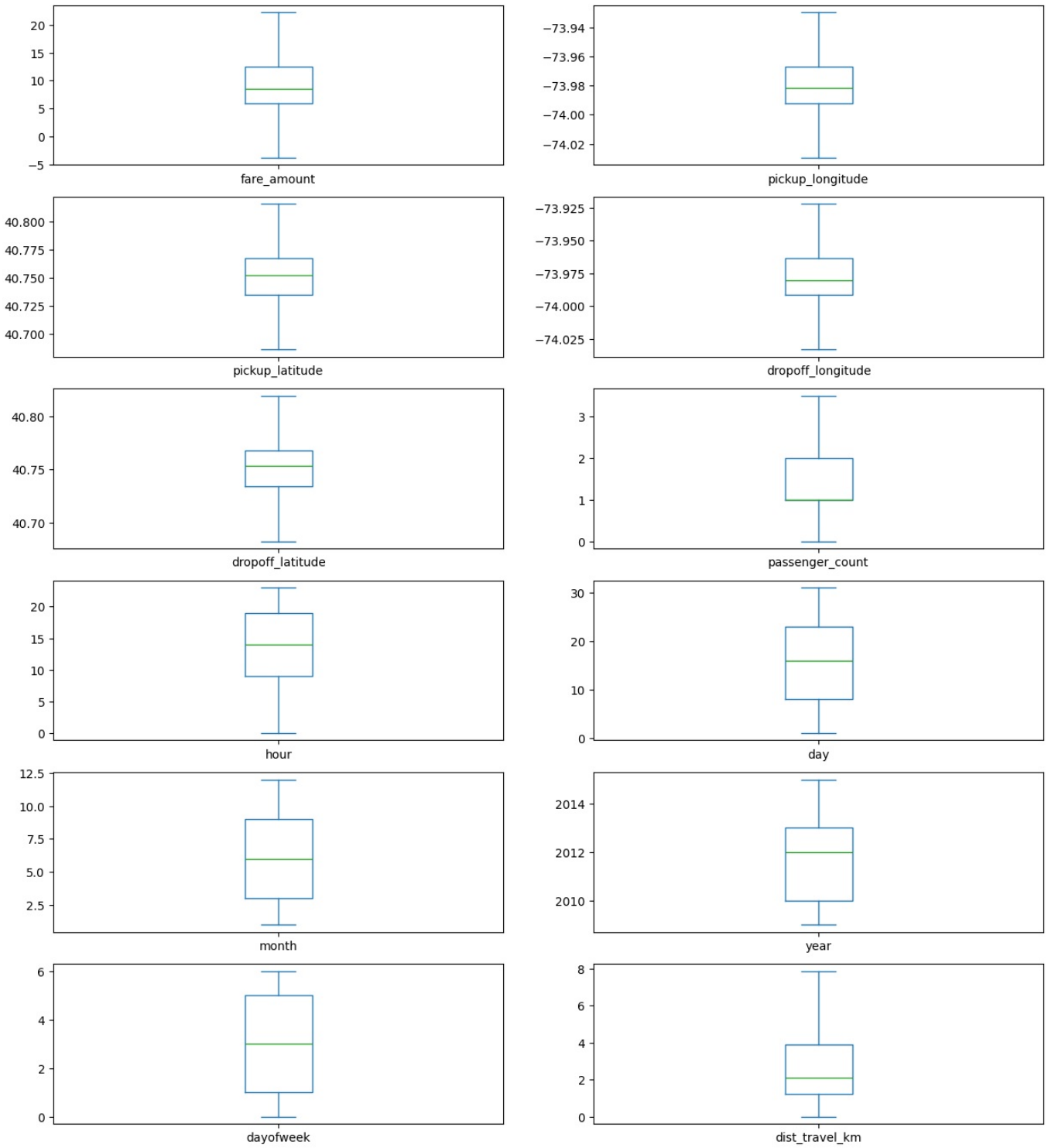




In [17]:
```
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1

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

In [18]:
```
df = treat_outliers_all(df , df.iloc[: , 0::])
```

In [19]:
```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
plt.show()
```



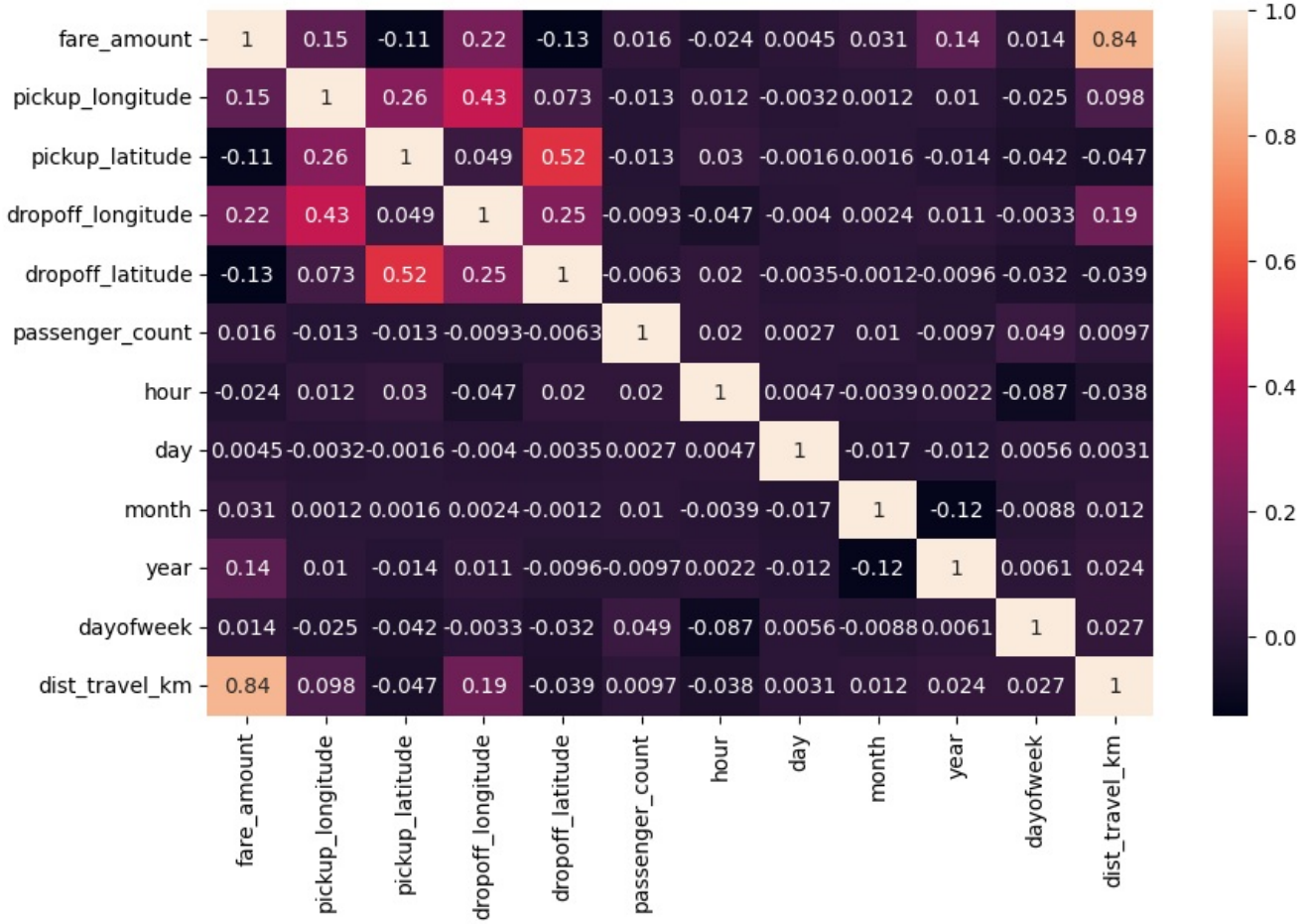
In [20]:
```
corr = df.corr()
corr
```

Out[20]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour
fare_amount	1.000000	0.154069	-0.110842	0.218675	-0.125898	0.015778	-0.023623
pickup_longitude	0.154069	1.000000	0.259497	0.425619	0.073290	-0.013213	0.011579
pickup_latitude	-0.110842	0.259497	1.000000	0.048889	0.515714	-0.012889	0.029681
dropoff_longitude	0.218675	0.425619	0.048889	1.000000	0.245667	-0.009303	-0.046558
dropoff_latitude	-0.125898	0.073290	0.515714	0.245667	1.000000	-0.006308	0.019783
passenger_count	0.015778	-0.013213	-0.012889	-0.009303	-0.006308	1.000000	0.020274
hour	-0.023623	0.011579	0.029681	-0.046558	0.019783	0.020274	1.000000
day	0.004534	-0.003204	-0.001553	-0.004007	-0.003479	0.002712	0.004677
month	0.030817	0.001169	0.001562	0.002391	-0.001193	0.010351	-0.003926
year	0.141277	0.010198	-0.014243	0.011346	-0.009603	-0.009749	0.002156
dayofweek	0.013652	-0.024652	-0.042310	-0.003336	-0.031919	0.048550	-0.086947
dist_travel_km	0.844374	0.098094	-0.046812	0.186531	-0.038900	0.009709	-0.038366

In [21]:
```
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True)
```

Out[21]: <Axes: >



In [22]:
```
df_x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_count','hour'
df_y = df['fare_amount']
```

In [23]:
```
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=1)
```

In [24]:
```
df
```

Out[24]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	y
0	7.50	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	5	2
1	7.70	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	7	2
2	12.90	-74.005043	40.740770	-73.962565	40.772647	1.0	21	24	8	2
3	5.30	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	6	2
4	16.00	-73.929786	40.744085	-73.973082	40.761247	3.5	17	28	8	2
...	...	...	...	...	...	...	...	...	...	
199995	3.00	-73.987042	40.739367	-73.986525	40.740297	1.0	10	28	10	2
199996	7.50	-73.984722	40.736837	-74.006672	40.739620	1.0	1	14	3	2
199997	22.25	-73.986017	40.756487	-73.922036	40.692588	2.0	0	29	6	2
199998	14.50	-73.997124	40.725452	-73.983215	40.695415	1.0	14	20	5	2
199999	14.10	-73.984395	40.720077	-73.985508	40.768793	1.0	4	15	5	2

200000 rows × 12 columns

In [25]:
```
from sklearn.linear_model import LinearRegression

# initialize the linear regression model
reg = LinearRegression()

# Train the model with our training data
reg.fit(x_train, y_train)
```

Out[25]:

LinearRegression

LinearRegression()

In [26]:
```
y_pred_lin = reg.predict(x_test)
print(y_pred_lin)
```

```
[ 6.27615184  5.09986098  9.43641238 ... 11.07663949 12.15392248
 11.41496075]
```

In [27]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100)
rf.fit(x_train,y_train)
```

Out[27]:

RandomForestRegressor

RandomForestRegressor()

In [28]:
```
y_pred_rf = rf.predict(x_test)
print(y_pred_rf)
```

```
[ 5.26    5.998   9.27   ... 11.4375 11.268  13.69  ]
```

In [29]:
```
cols = ['Model', 'RMSE', 'R-Squared']

# create a empty dataframe of the colums
# columns: specifies the columns to be selected
result_tabulation = pd.DataFrame(columns = cols)
```

In [31]:
```
from sklearn import metrics
from sklearn.metrics import r2_score

reg_RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred_lin))
reg_squared = r2_score(y_test, y_pred_lin)

full_metrics = pd.Series({'Model': "Linear Regression", 'RMSE' : reg_RMSE, 'R-Squared' : reg_squared})

# append our result table using append()
# ignore_index=True: does not use the index labels
# python can only append a Series if ignore_index=True or if the Series has a name
result_tabulation = pd.concat([result_tabulation, full_metrics], ignore_index=True)

# print the result table
result_tabulation
```

Out[31]:

	Model	RMSE	R-Squared	0
0	NaN	NaN	NaN	Linear Regression
1	NaN	NaN	NaN	2.703957
2	NaN	NaN	NaN	0.753906

In [33]:

```
rf_RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf))
rf_squared = r2_score(y_test, y_pred_rf)


full_metrics = pd.Series({'Model': "Random Forest ", 'RMSE':rf_RMSE, 'R-Squared': rf_squared})
# append our result table using append()
# ignore_index=True: does not use the index labels
# python can only append a Series if ignore_index=True or if the Series has a name
# Convert the Series to a DataFrame and use pd.concat
result_tabulation = pd.concat([result_tabulation, full_metrics.to_frame().T], ignore_index=True)


# print the result table
result_tabulation
```

Out[33]:

	Model	RMSE	R-Squared	0
0	NaN	NaN	NaN	Linear Regression
1	NaN	NaN	NaN	2.703957
2	NaN	NaN	NaN	0.753906
3	Random Forest	2.363921	0.811909	NaN

In [ ]: