In [46]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [47]:
```python
df = pd.read_csv('uber.csv')
```

In [48]:
```python
df.head()
```

Out[48]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pick
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	
3	25894730	2009-06-26 8:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	

In [49]:
```python
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 [50]:
```python
df.columns
```

Out[50]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [62]:
```python
df = df.drop(['Unnamed: 0', 'key'], axis=1)
```

In [63]:
```
df.shape
```

Out[63]:
```
(200000, 7)
```

In [64]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   fare_amount        200000 non-null  float64
 1   pickup_datetime    200000 non-null  object 
 2   pickup_longitude   200000 non-null  float64
 3   pickup_latitude    200000 non-null  float64
 4   dropoff_longitude  199999 non-null  float64
 5   dropoff_latitude   199999 non-null  float64
 6   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 10.7+ MB
```

In [65]:
```
df.describe()
```

Out[65]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	drop
count	200000.000000	200000.000000	200000.000000	199999.000000	19
mean	11.359955	-72.527638	39.935885	-72.525292	
std	9.901776	11.437787	7.720539	13.117408	
min	-52.000000	-1340.648410	-74.015515	-3356.666300	
25%	6.000000	-73.992065	40.734796	-73.991407	
50%	8.500000	-73.981823	40.752592	-73.980093	
75%	12.500000	-73.967153	40.767158	-73.963659	
max	499.000000	57.418457	1644.421482	1153.572603	

In [66]:
```
df.dtypes
```

Out[66]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [67]:
```
df.isnull().sum()
```

Out[67]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [68]:
```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(), inplace=Tru
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].mean(), inplace=T
```

```
C:\Users\Rohan\AppData\Local\Temp\ipykernel_1608\3614752822.py:1: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(), inplace=Tr
ue)
C:\Users\Rohan\AppData\Local\Temp\ipykernel_1608\3614752822.py:2: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].mean(), inplac
e=True)
```

In [69]:
```
df.isnull().sum()
```

Out[69]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [70]:
```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors= 'coerce')
```

In [72]:
```python
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 [73]:
```python
df.head()
```

Out[73]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_lon
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.9
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.9
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.9
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.9
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.9

In [75]:
```python
df = df.drop('pickup_datetime', axis=1)
```

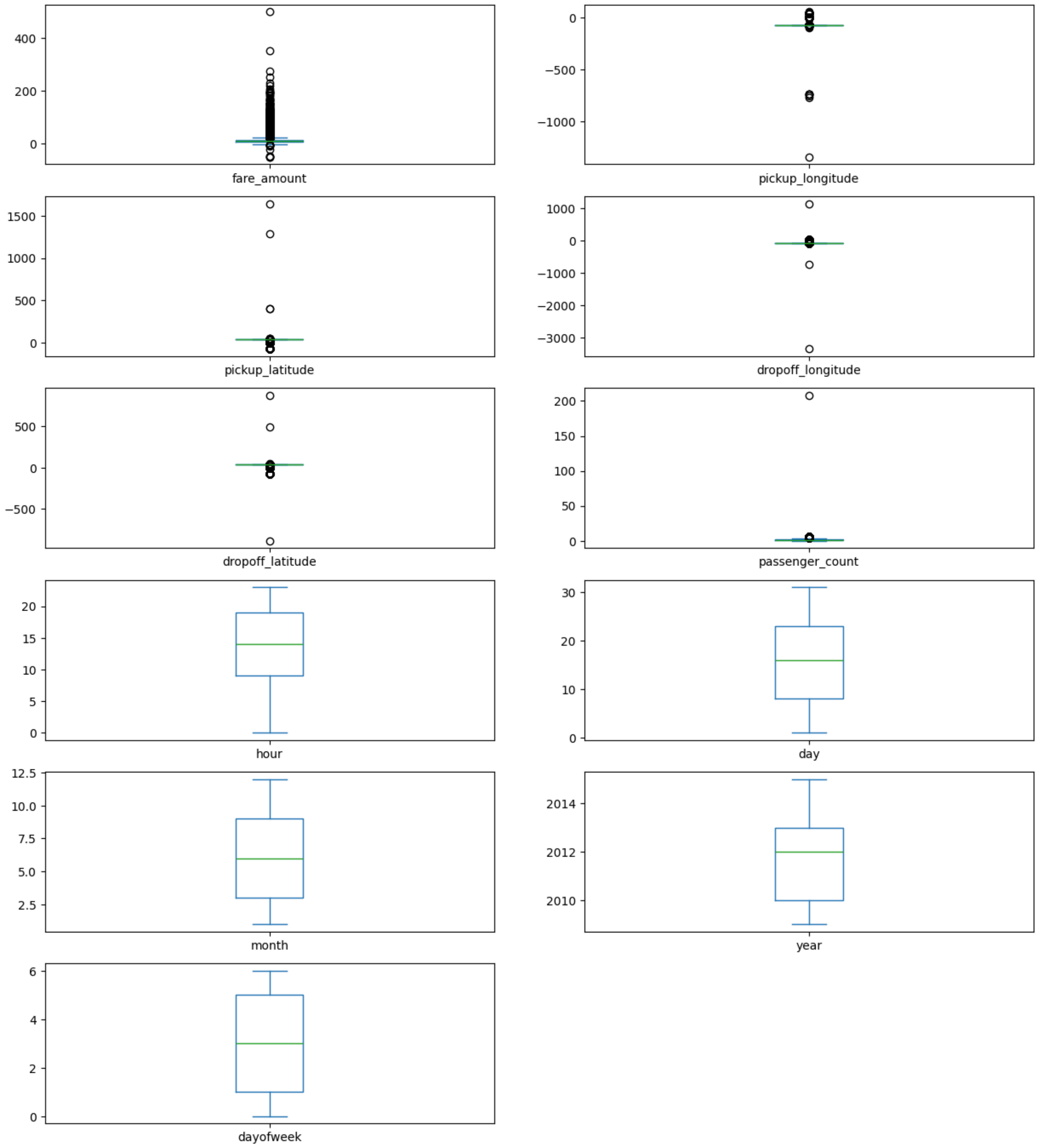
In [76]:
```python
df.head()
```

Out[76]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
0	7.5	-73.999817	40.738354	-73.999512	40.7
1	7.7	-73.994355	40.728225	-73.994710	40.7
2	12.9	-74.005043	40.740770	-73.962565	40.7
3	5.3	-73.976124	40.790844	-73.965316	40.8
4	16.0	-73.925023	40.744085	-73.973082	40.7

In [77]:
```python
df.plot(kind = 'box', subplots=True, layout=(7,2), figsize=(15,20))
```

Out[77]:
```
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)
hour                    Axes(0.125,0.448049;0.352273x0.0939024)
day                  Axes(0.547727,0.448049;0.352273x0.0939024)
month                   Axes(0.125,0.335366;0.352273x0.0939024)
year                 Axes(0.547727,0.335366;0.352273x0.0939024)
dayofweek               Axes(0.125,0.222683;0.352273x0.0939024)
dtype: object
```



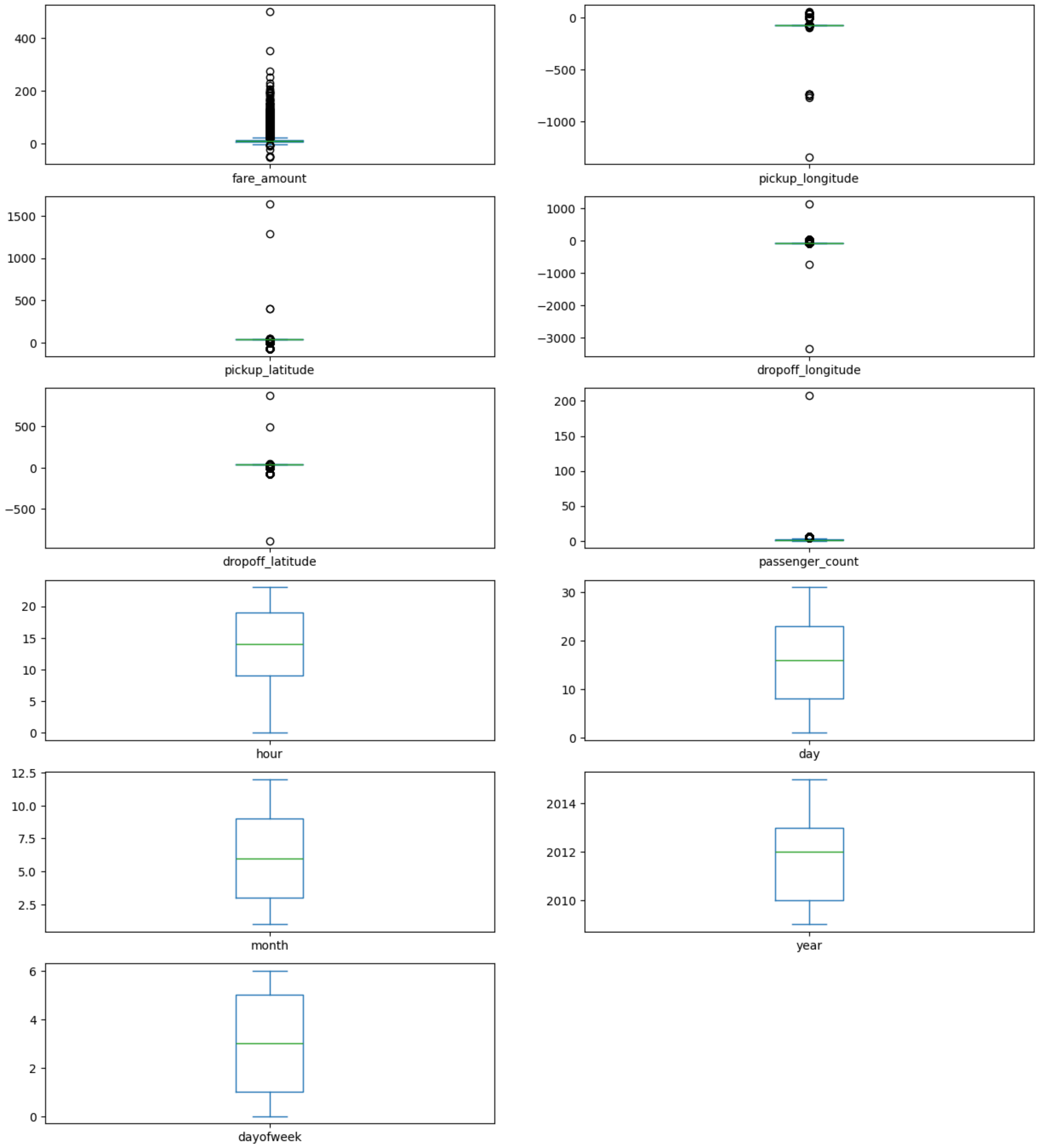
In [78]:

```
#Using the interquartile range to fill the values
def remove_outlier(df1, col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = np.clip(df[col], lower, upper)
    return df1

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

In [79]:
```
df.plot(kind='box', subplots=True, layout=(7,2), figsize=(15,20))
```

Out[79]:
```
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)
hour                       Axes(0.125,0.448049;0.352273x0.0939024)
day                  Axes(0.547727,0.448049;0.352273x0.0939024)
month                      Axes(0.125,0.335366;0.352273x0.0939024)
year                 Axes(0.547727,0.335366;0.352273x0.0939024)
dayofweek                  Axes(0.125,0.222683;0.352273x0.0939024)
dtype: object
```



In [85]:

```
import haversine as hs

travel_dist = []

# Ensure valid coordinates
df = df.loc[
    (df['pickup_latitude'].between(-90, 90)) &
    (df['dropoff_latitude'].between(-90, 90)) &
    (df['pickup_longitude'].between(-180, 180)) &
    (df['dropoff_longitude'].between(-180, 180))
].copy()
```

```
for pos in range(len(df)):
    lat1, lon1 = df['pickup_latitude'].iloc[pos], df['pickup_longitude'].iloc[
    lat2, lon2 = df['dropoff_latitude'].iloc[pos], df['dropoff_longitude'].ilo
    loc1 = (lat1, lon1)
    loc2 = (lat2, lon2)
    dist = hs.haversine(loc1, loc2)  # distance in kilometers
    travel_dist.append(dist)

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

Out[85]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
0	7.5	-73.999817	40.738354	-73.999512	40.7
1	7.7	-73.994355	40.728225	-73.994710	40.7
2	12.9	-74.005043	40.740770	-73.962565	40.7
3	5.3	-73.976124	40.790844	-73.965316	40.8
4	16.0	-73.925023	40.744085	-73.973082	40.7

In [87]:

```
# Uber doesn't travel over 130kms so minimize distance
df = df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130)]
print("Remaining observations in the dataset:", df.shape)
```

```
Remaining observations in the dataset: (199988, 12)
```

In [88]:

```
#Finding inccorect latitude (Less than or greater than 90) and longitude (grea
incorrect_coordinates = df.loc[(df.pickup_latitude > 90) |(df.pickup_latitude 
                                   (df.dropoff_latitude > 90) |(df.dropoff_lat
                                   (df.pickup_longitude > 180) |(df.pickup_lor
                                   (df.dropoff_longitude > 90) |(df.dropoff_lc
                                    ]
```

In [90]:

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

In [91]:

```
df.head()
```

Out[91]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
0	7.5	-73.999817	40.738354	-73.999512	40.7
1	7.7	-73.994355	40.728225	-73.994710	40.7
2	12.9	-74.005043	40.740770	-73.962565	40.7
3	5.3	-73.976124	40.790844	-73.965316	40.8
4	16.0	-73.925023	40.744085	-73.973082	40.7

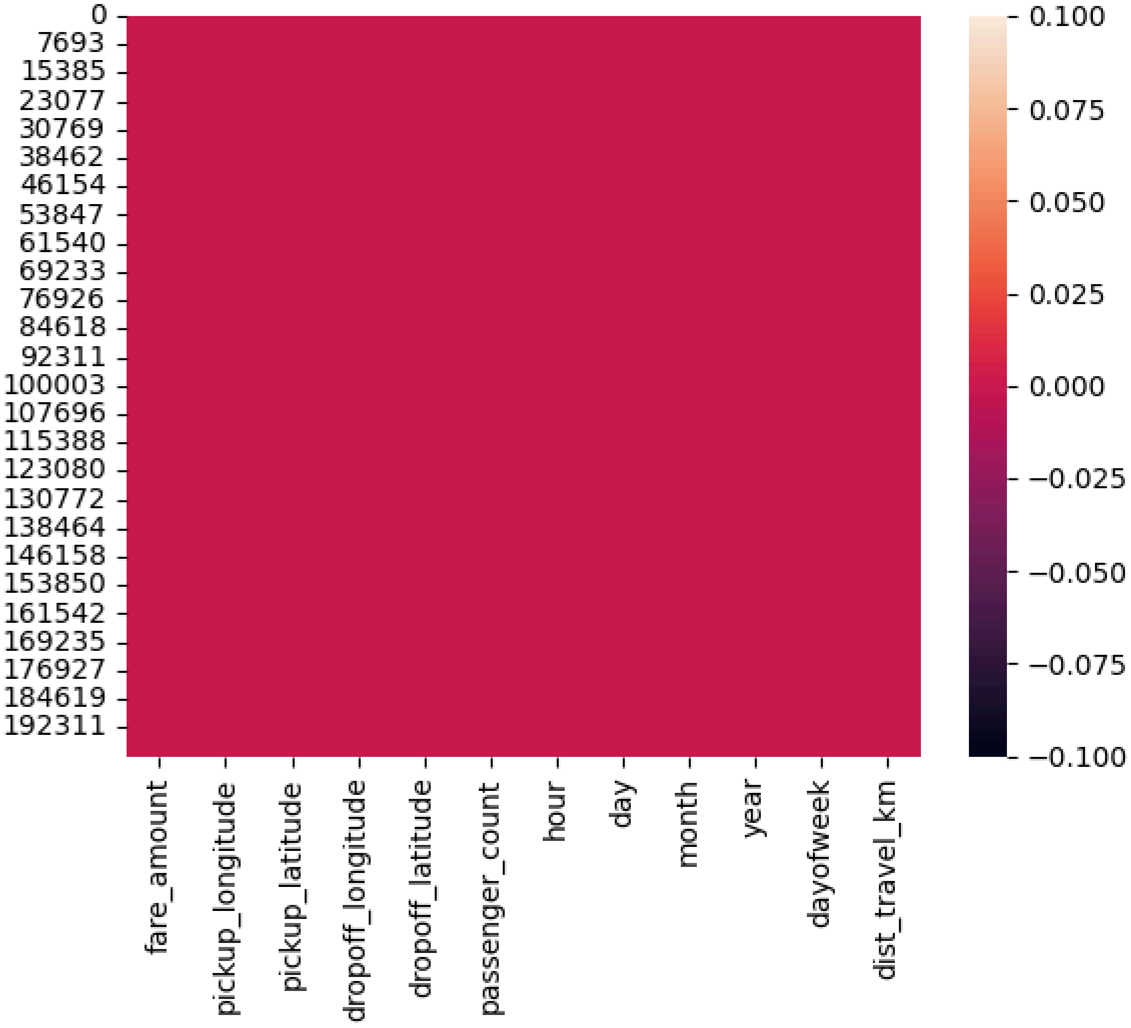
In [92]:

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

```
Out[92]: 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
```

```
In [93]: sns.heatmap(df.isnull())
```

```
Out[93]: <Axes: >
```



```
In [94]: corr = df.corr()
```

```
In [95]: corr
```

Out[95]:

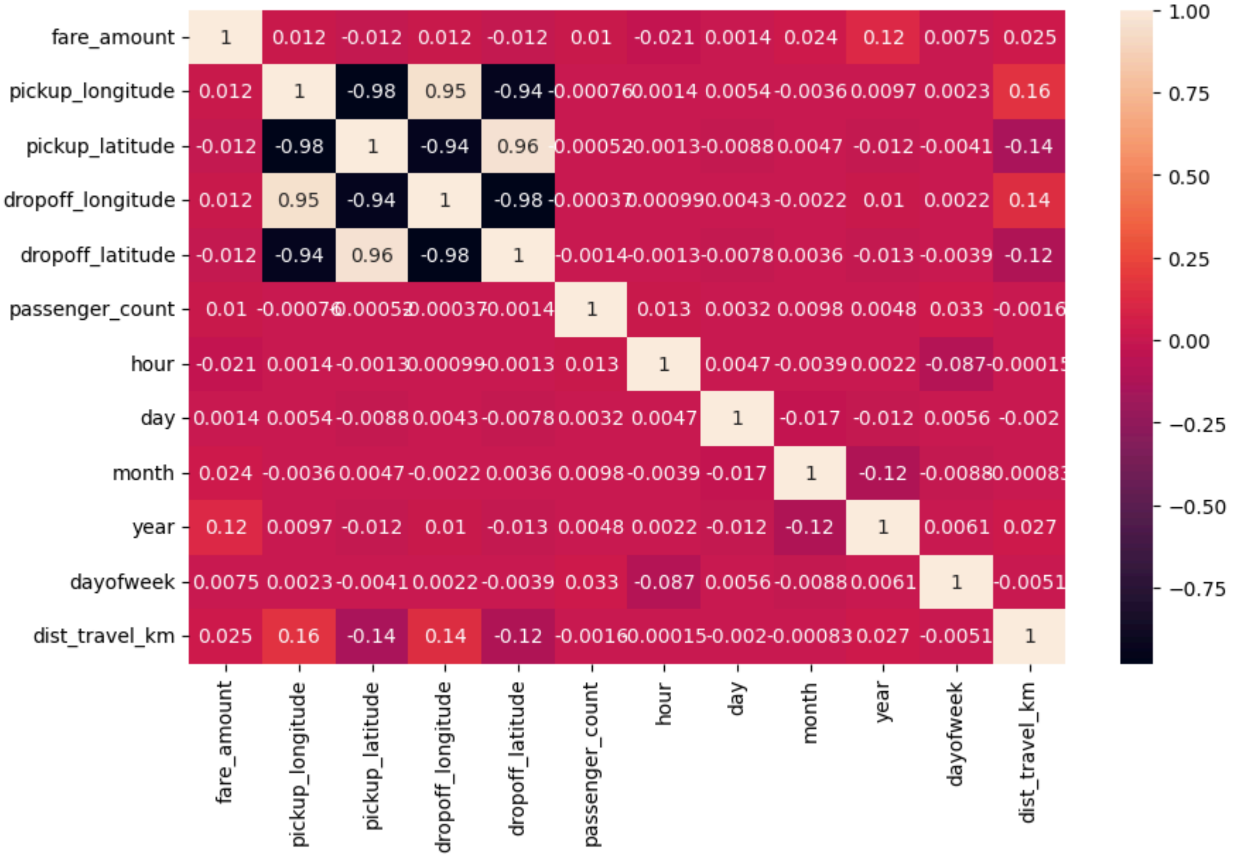
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longi
fare_amount	1.000000	0.011635	-0.011561	0.01
pickup_longitude	0.011635	1.000000	-0.979048	0.94
pickup_latitude	-0.011561	-0.979048	1.000000	-0.93
dropoff_longitude	0.011870	0.949698	-0.936690	1.00
dropoff_latitude	-0.012258	-0.936642	0.958143	-0.97
passenger_count	0.010159	-0.000756	-0.000523	-0.00
hour	-0.021495	0.001393	-0.001346	0.00
day	0.001395	0.005368	-0.008814	0.00
month	0.023795	-0.003574	0.004718	-0.00
year	0.118339	0.009736	-0.012036	0.01
dayofweek	0.007492	0.002312	-0.004148	0.00
dist_travel_km	0.024723	0.163547	-0.142200	0.14

In [96]:

```
fig, axis = plt.subplots(figsize=(10,6))
sns.heatmap(df.corr(), annot=True)
```

Out[96]:

```
<Axes: >
```



In [100…

```python
x = df[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_l
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 = 0.33)
```

In [ ]:

```python
# Linear Regression
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train, y_train)
```

Out[ ]:

▾ LinearRegression ⓘ ?

Parameters

⧉	fit_intercept	True
⧉	copy_X	True
⧉	tol	1e-06
⧉	n_jobs	None
⧉	positive	False

In [102…

```python
regression.intercept_
```

Out[102…

```
np.float64(-1308.5763838650173)
```

In [103…

```python
regression.coef_
```

Out[103…

```
array([ 1.82048902e-02,  6.29490061e-02, -2.44374099e-02, -8.05621034e-02,
        6.22658606e-02, -3.02383898e-02,  4.64851748e-03,  1.05018214e-01,
        6.55984557e-01,  2.34767432e-02,  5.40340472e-04])
```

In [105…

```python
prediction = regression.predict(X_test)
prediction
```

Out[105…

```
array([12.42117782, 11.66326224, 11.46637051, ..., 10.86121541,
       11.32915104, 12.33336062], shape=(65997,))
```

In [106…

```python
y_test
```

```
Out[106…  52153     19.5
          47578      8.9
          39397      8.0
          35679      5.5
          63534      8.5
                    ...
          66104      6.1
          41390     12.0
          76456      8.5
          147242    11.7
          19361     18.0
          Name: fare_amount, Length: 65997, dtype: float64
```

```
In [107…  #R2, MSE, RMSE
          from sklearn.metrics import r2_score
          r2_score(y_test, prediction)
```

```
Out[107…  0.016433455965117583
```

```
In [108…  from sklearn.metrics import mean_squared_error
          MSE = mean_squared_error(y_test, prediction)
          MSE
```

```
Out[108…  96.48994462606969
```

```
In [112…  RMSE = np.sqrt(MSE)
          RMSE
```

```
Out[112…  np.float64(9.822929533803533)
```