```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

df  = pd.read_csv("/content/drive/MyDrive/uber.csv")

df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
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
```

```
df.columns
```

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

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

df.dtypes
```

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
```

```
passenger_count        int64
dtype: object
```

```
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
```

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n{\n      \"column\": \"fare_amount\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 70685.88306171099,\n\"min\": -52.0,\n        \"max\": 200000.0,\n\"num_unique_values\": 8,\n        \"samples\": [\n11.359955250000002,\n          8.5,\n          200000.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"pickup_longitude\",\n\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 70791.27149688502,\n        \"min\": -1340.64841,\n        \"max\": 200000.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n-72.5276379162372,\n          -73.98182299999999,\n          200000.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"pickup_latitude\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 70625.08842178025,\n        \"min\": -74.01551500000001,\n        \"max\": 200000.0,\n\"num_unique_values\": 8,\n        \"samples\": [\n39.93588537801235,\n          40.752592,\n          200000.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"dropoff_longitude\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 70847.64282023362,\n        \"min\": -3356.6663,\n        \"max\": 199999.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          -72.52529162747415,\n          -73.98009300000001,\n          199999.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"dropoff_latitude\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 70703.81752683062,\n   \"min\": -881.9855130000001,\n   \"max\": 199999.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   39.92389040183263,\n   40.753042,\n   199999.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"passenger_count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 70699.85246121645,\n   \"min\": 0.0,\n   \"max\": 200000.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   200000.0,\n   1.684535,\n   2.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

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

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

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

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

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

```
df.dtypes
```

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

```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime,
errors='coerce')

df.dtypes
```

```
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
```

```
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()
```

{"type":"dataframe","variable_name":"df"}

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

df.head()
```

{"type":"dataframe","variable_name":"df"}

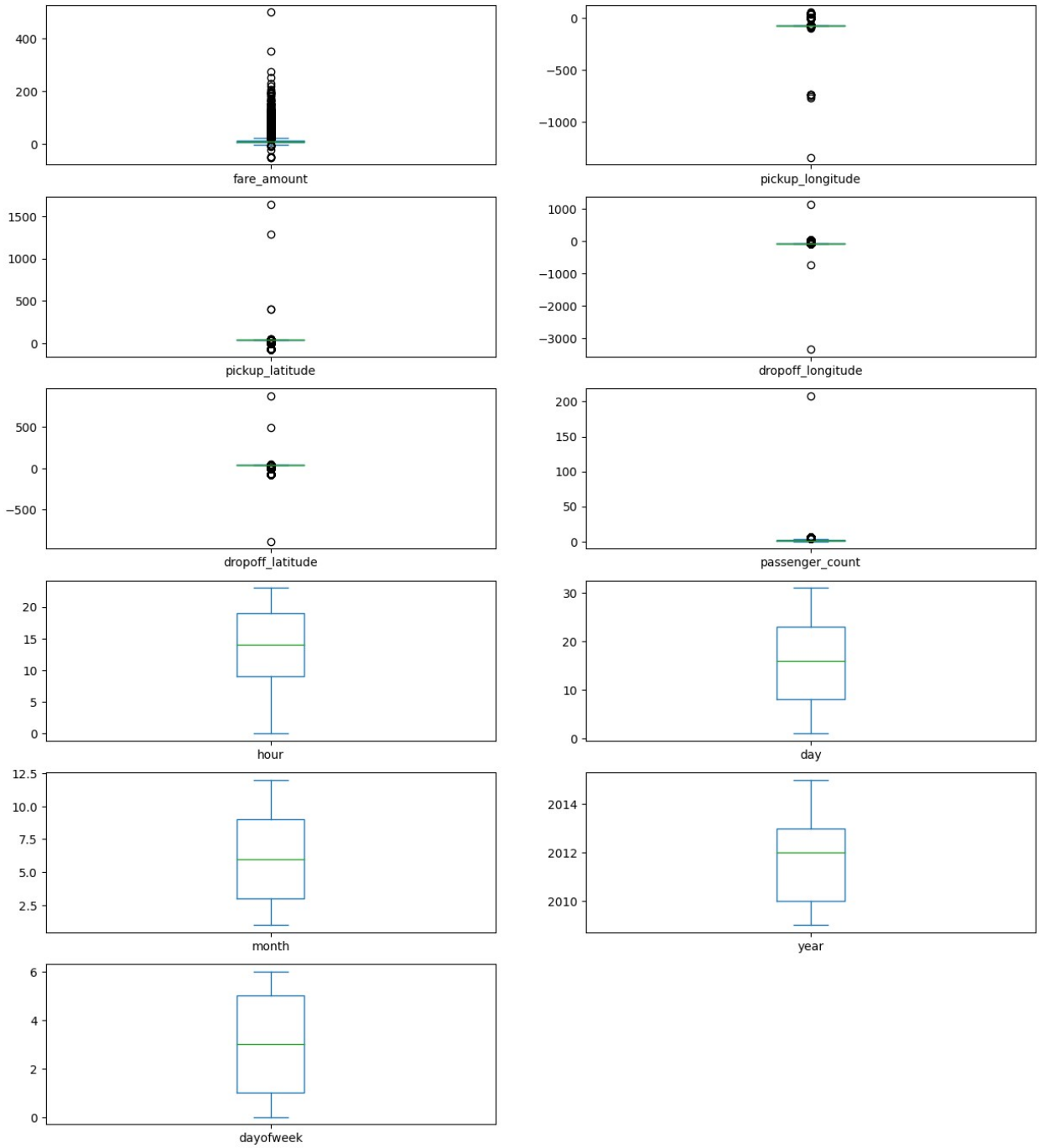
```
df.dtypes
```

```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
hour                   int32
day                    int32
month                  int32
year                   int32
dayofweek              int32
dtype: object
```

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

```
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
```



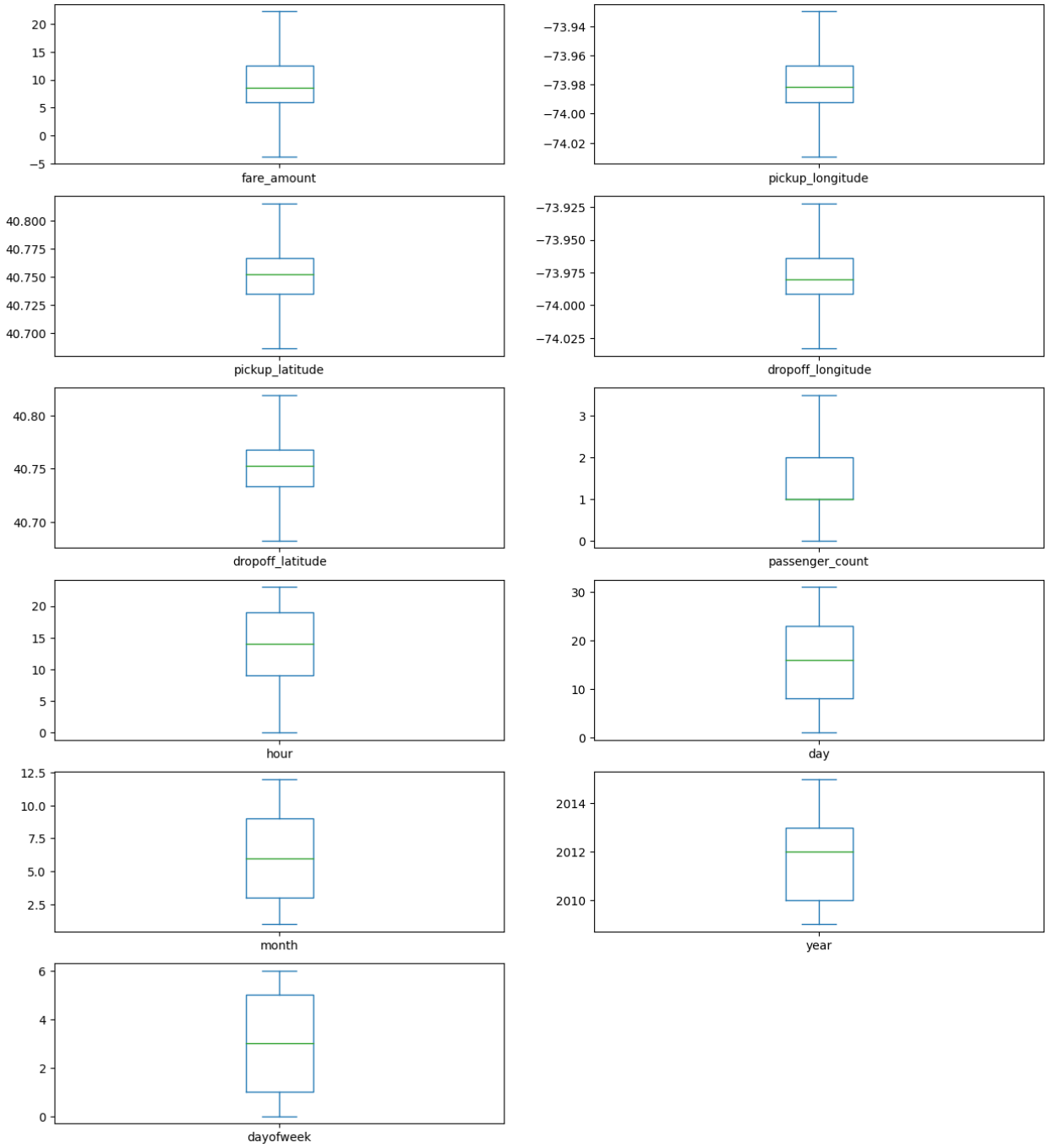
```
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1

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=(15,20))
```

```
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
```




```
pip install haversine
```

```
Collecting haversine
  Downloading haversine-2.8.1-py2.py3-none-any.whl.metadata (5.9 kB)
Downloading haversine-2.8.1-py2.py3-none-any.whl (7.7 kB)
Installing collected packages: haversine
Successfully installed haversine-2.8.1
```

```
import haversine as hs  #Calculate the distance using Haversine to
calculate the distance between to points. Can't use Eucladian as it is
for flat surface.
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_latitude'][pos]]
        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()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

```
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, 12)
```

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

df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

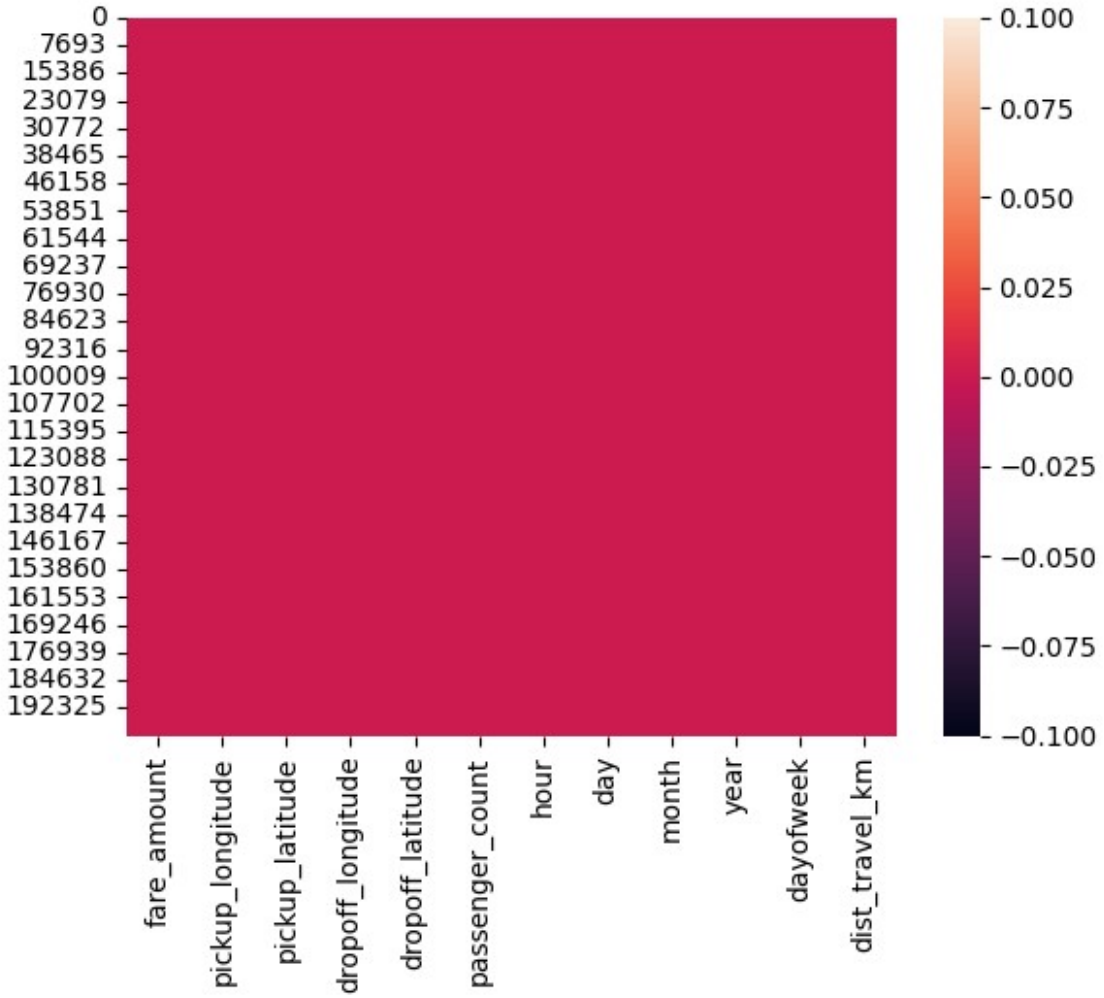
```
df.isnull().sum()
```

```
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())
```

```
<Axes: >
```




```
corr = df.corr()
```

```
corr
```

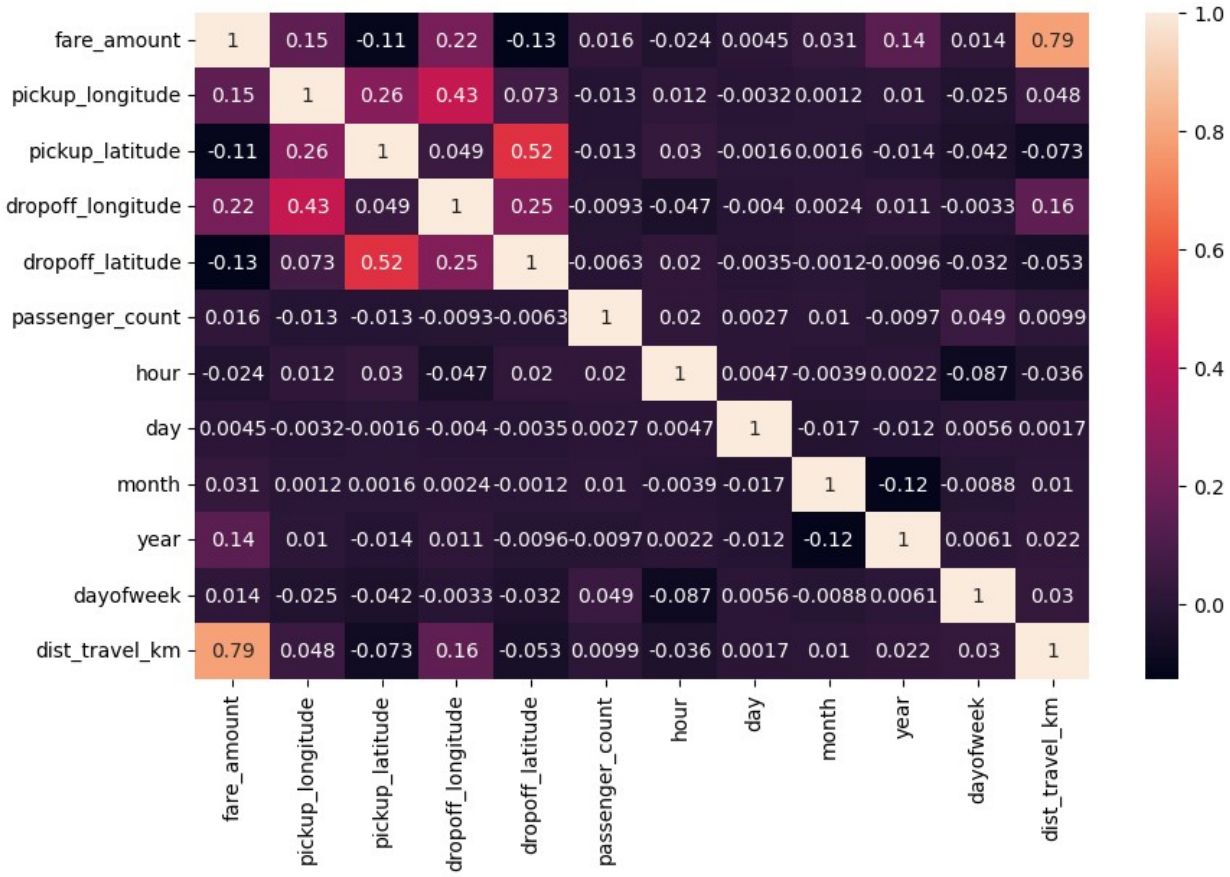

```
{"summary":"{\n  \"name\": \"corr\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"fare_amount\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.3530164084793291,\
n        \"min\": -0.12589812051371674,\n        \"max\": 1.0,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
0.013651533668647681,\n          0.1412768795014389,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    },\n    {\n      \"column\": \"pickup_longitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2964087476143438,\n        \"min\": -0.02465176263508481,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          -0.02465176263508481,\n
0.010197727853871167,\n          0.15406867814231004\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"pickup_latitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.3223535989273809,\n        \"min\": -0.11084153642791626,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          -0.042309798615848375,\n          -
0.01424282079255877,\n          -0.11084153642791626\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"dropoff_longitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.29693214652563055,\n        \"min\": -0.04655801753989324,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          -0.003336155766878457,\n
0.011346328673746155,\n          0.21867546642995583\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"dropoff_latitude\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.3208315591338511,\n        \"min\": -0.12589812051371674,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          -0.03191864717514768,\n          -
0.009603062607838405,\n          -0.12589812051371674\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"passenger_count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.28775748468485607,\n        \"min\": -0.013213402695744067,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          0.04855041655159065,\n          -
0.00974909902879583,\n          0.01577806364994497\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"hour\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.29344432808616494,\n
\"min\": -0.08694676294082432,\n        \"max\": 1.0,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          -
0.08694676294082432,\n          0.0021564681469461466,\n          -
0.023623024527920176\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"day\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.2893492493162671,\n        \"min\": -0.0173599670029169,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          0.005617375927261731,\n          -
0.01217044446998976,\n          0.00453441527576609\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"month\",\n      \"properties\": {\
```

```
n        \"dtype\": \"number\",\n        \"std\":
0.29327251641188573,\n        \"min\": -0.11585892363832388,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          -0.008786062451844028,\n          -
0.11585892363832388,\n          0.03081716357671837\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"year\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.29325474907255933,\n
\"min\": -0.11585892363832388,\n        \"max\": 1.0,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
0.006112503974573967,\n          1.0,\n          0.1412768795014389\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"dayofweek\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.29327958644365965,\n        \"min\": -0.08694676294082432,\n
\"max\": 1.0,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          1.0,\n          0.006112503974573967,\n
0.013651533668647681\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"dist_travel_km\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.350874390856739,\n        \"min\": -
0.07336159370966586,\n        \"max\": 1.0,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
0.030382419735173936,\n          0.022293854606573498,\n
0.7863854173173256\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"corr"}
```


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

```
<Axes: >
```




```
x =
df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_
latitude','passenger_count','hour','day','month','year','dayofweek','d
ist_travel_km']]

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)

from sklearn import linear_model

regression= linear_model.LinearRegression()

regression.fit(X_train,y_train)

LinearRegression()

regression.intercept_

3673.6404694352077

regression.coef_
```

```
array([ 2.54819531e+01, -6.75553931e+00,  2.02962916e+01, -
1.84152987e+01,
        5.89922000e-02,  6.70469087e-03,  3.23698958e-03,
6.06595475e-02,
        3.69349986e-01, -3.13959610e-02,  1.84769250e+00])

prediction = regression.predict(X_test)

print(prediction)

[22.69472796  6.86882506 14.15656814 ...  6.5880424   7.55468035
  5.8841651 ]

y_test

117370    22.25
104918     4.10
117678    22.25
19788     10.50
115794     4.50
          ...
133954    22.25
149606    11.30
71458      4.50
142421     6.50
38836      4.50
Name: fare_amount, Length: 66000, dtype: float64

from sklearn.metrics import r2_score

r2_score(y_test,prediction)

0.6665912749930101

from sklearn.metrics import mean_squared_error

MSE = mean_squared_error(y_test,prediction)

MSE

9.863999527140187

RMSE = np.sqrt(MSE)

RMSE

3.140700483513222

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100)

rf.fit(X_train,y_train)
```

```
RandomForestRegressor()
```

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

```
y_pred
```

```
array([18.963,  5.795, 22.24 , ...,  4.12 ,  5.876,  5.064])
```

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

```
R2_Random
```

```
0.7972354569480166
```

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

```
MSE_Random
```

```
5.998851280042625
```

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

```
RMSE_Random
```

```
2.449255250079629
```