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
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,r2_score
```

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

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_lat
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.72
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.74
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.79
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.74

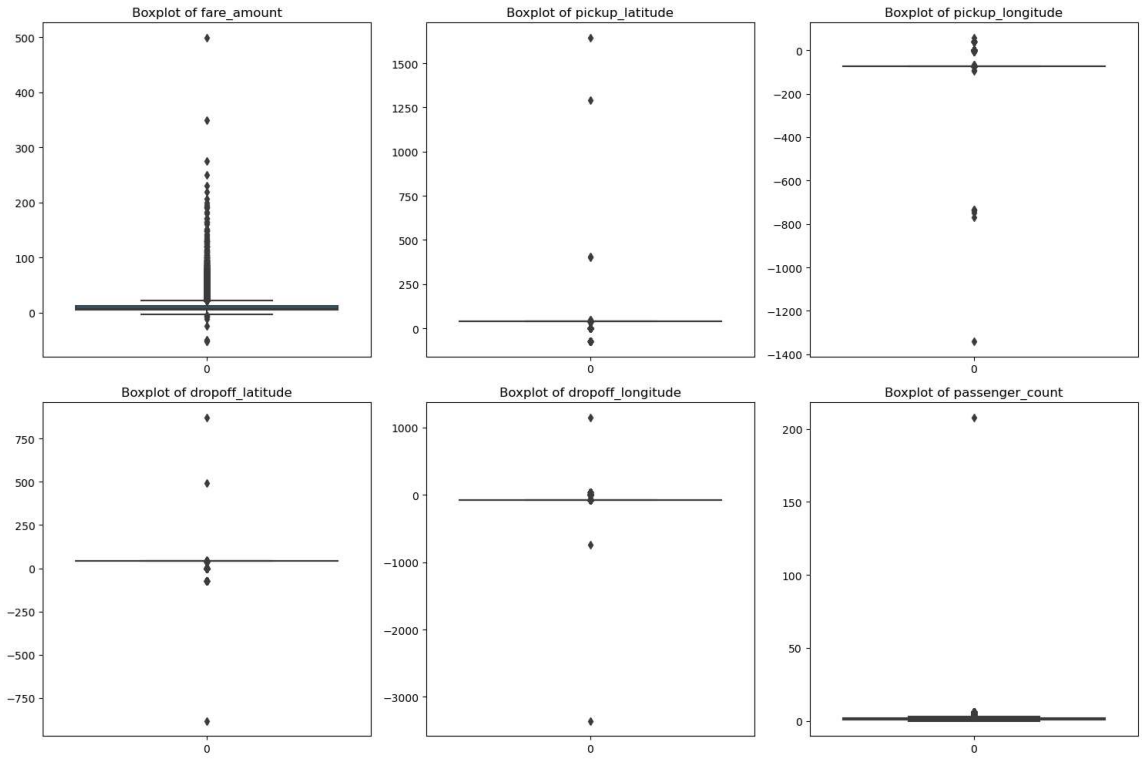
In [4]:
```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [5]:
```
df['hour']=df['pickup_datetime'].dt.hour
df['day']=df['pickup_datetime'].dt.day
df['day_of_week']=df['pickup_datetime'].dt.dayofweek
df['month']=df['pickup_datetime'].dt.month
```

In [6]:
```
df=df.dropna()
```

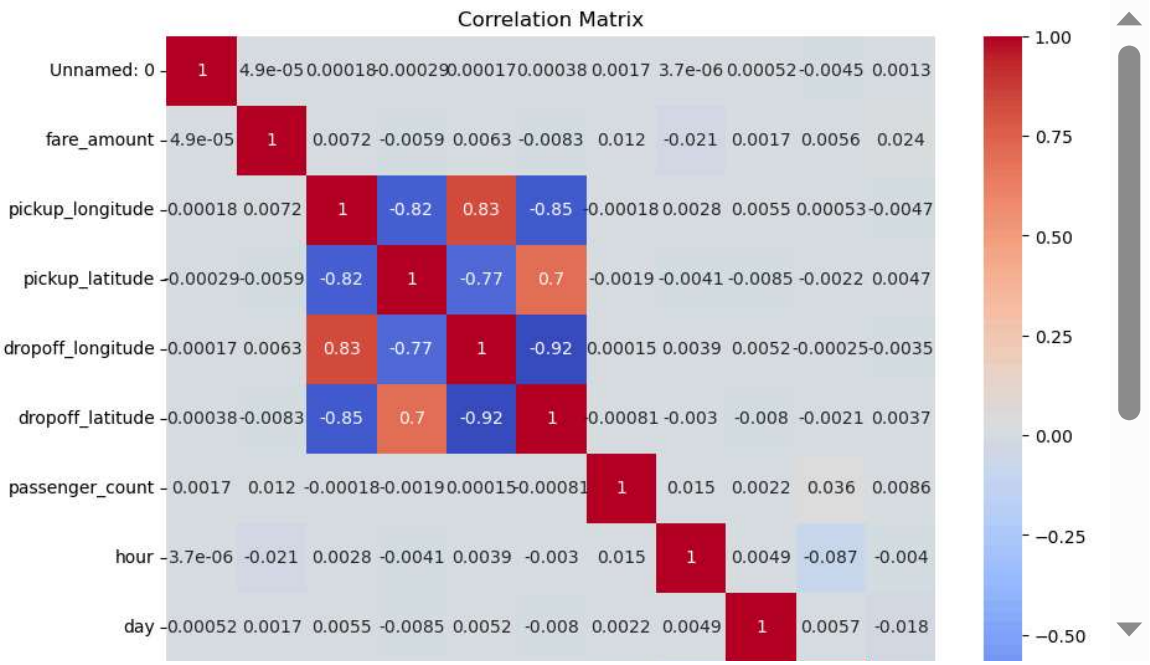
In [7]:
```
df=df.drop(['pickup_datetime'], axis=1)
```

In [8]:
```
numeric_features=['fare_amount','pickup_latitude','pickup_longitude','dropo
plt.figure(figsize=(15,10))
for i,feature in enumerate(numeric_features):
    plt.subplot(2,3,i+1)
    sns.boxplot(df[feature])
    plt.title(f'Boxplot of {feature}')
plt.tight_layout()
plt.show()
```



In [9]:
```
df = df[(df['fare_amount']>0) & (df['fare_amount']<100)]
df = df[(df['passenger_count']>0) & (df['passenger_count']<=6)]
```

In [10]:
```
numeric_df = df.select_dtypes(include=['number'])
corr_matrix = numeric_df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix, annot=True,cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [11]:
```
X = df[['pickup_latitude','pickup_longitude','dropoff_latitude','dropoff_lo
y = df['fare_amount']
```

In [12]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_sta
```

In [17]:
```
lr_model = LinearRegression()
lr_model.fit(X_train,y_train)
y_pred_lr = lr_model.predict(X_test)
```

In [18]:
```
rf_model = RandomForestRegressor(n_estimators=100,random_state=42)
rf_model.fit(X_train,y_train)
y_pred_rf = rf_model.predict(X_test)
```

In [19]:
```
def evaluate_model(y_test,y_pred,model_name):
    r2 = r2_score(y_test,y_pred)
    rmse = np.sqrt(mean_squared_error(y_test,y_pred))
    print(f'{model_name}-R2Score:{r2:2f},RMSE:{rmse:.2f}')

evaluate_model(y_test,y_pred_lr,'Linear Regression')
evaluate_model(y_test,y_pred_rf,'Random Forest Regression')
```

```
Linear Regression-R2Score:0.001059,RMSE:9.45
Random Forest Regression-R2Score:0.803208,RMSE:4.19
```

In [20]:
```
print("Linear Regression vs Random Forest Regression:")
print("Linear Regression R2 Score: ",r2_score(y_test,y_pred_lr))
print("Linear Regression RMSE:",np.sqrt(mean_squared_error(y_test,y_pred_lr
print("Random Forest Regression R2 Score: ",r2_score(y_test,y_pred_rf))
print("Random Forest Regression RMSE: ",np.sqrt(mean_squared_error(y_test,y
```

```
Linear Regression vs Random Forest Regression:
Linear Regression R2 Score:  0.0010594227072876494
Linear Regression RMSE: 9.45007180929204
Random Forest Regression R2 Score:  0.8032075026481964
Random Forest Regression RMSE:  4.194397152263582
```