In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
df=pd.read_csv("C:/Users/daksh/OneDrive/Documents/Desktop/datasets/uber.csv")
```

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

Out[4]:

	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 [5]:
```
df.shape
```

Out[5]:
```
(200000, 9)
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Unnamed: 0           0
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [7]:
```
df = df.dropna(subset=['dropoff_latitude', 'dropoff_longitude'])
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Unnamed: 0           0
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [9]:
```
int(df['pickup_latitude'].isnull().sum())
df.dtypes
```

Out[9]:
```
Unnamed: 0             int64
key                   object
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]:
```
Unnamed: 0                           int64
key                                 object
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]:
```
# Remove invalid latitude and longitude values
df = df[
    (df['pickup_latitude'].between(-90, 90)) &
    (df['dropoff_latitude'].between(-90, 90)) &
    (df['pickup_longitude'].between(-180, 180)) &
    (df['dropoff_longitude'].between(-180, 180))
]
```

In [12]:
```
from geopy.distance import great_circle

# Compute haversine distance
def calculate_distance(row):
    pickup = (row['pickup_latitude'], row['pickup_longitude'])
    dropoff = (row['dropoff_latitude'], row['dropoff_longitude'])
    return great_circle(pickup, dropoff).km

df['distance_km'] = df.apply(calculate_distance, axis=1)

# Remove extreme values
df = df[(df['distance_km'] > 0) & (df['distance_km'] < 50)]
df = df[df['fare_amount'] < 100]

#geopy is a Python library that makes it easy to calculate distances, latitudes, and longitudes between two points on the surface of the Earth
#The great circle distance formula provides an accurate calculation of the distance between two points on the Earth's surface.
```

In [13]:
```
df.shape
```

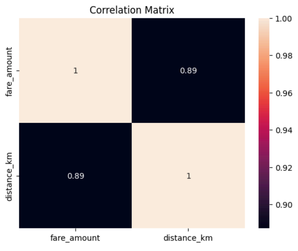
Out[13]:
```
(193797, 10)
```

In [14]:
```
print(df.columns)
```

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

In [15]:
```
from geopy.distance import great_circle

def calculate_distance(row):
    try:
        pickup = (row['pickup_latitude'], row['pickup_longitude'])
        dropoff = (row['dropoff_latitude'], row['dropoff_longitude'])
        return great_circle(pickup, dropoff).km
    except:
        return None

df['distance_km'] = df.apply(calculate_distance, axis=1)
df = df.dropna(subset=['distance_km'])
```

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt

corr = df[['fare_amount', 'distance_km']].corr()
sns.heatmap(corr, annot=True)
plt.title("Correlation Matrix")
plt.show()
```



In [22]:
```
# Step 4: Train Linear Regression & Random Forest Models
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

In [27]:
```
# Select features and target
X = df[['distance_km']]
y = df[['fare_amount']]
```

In [28]:
```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [29]:
```
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_preds = lr.predict(X_test)
```

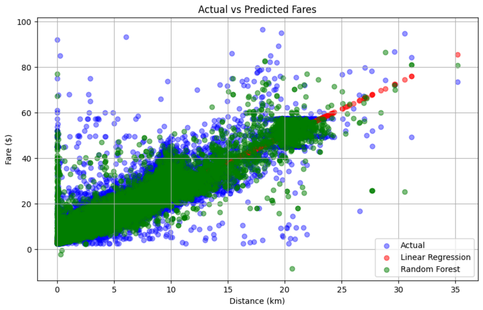
In [30]:
```
# Random Forest Regression
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_preds = rf.predict(X_test)
```

```
C:\Users\daksh\AppData\Local\Programs\Python\Python314\Lib\site-packages\sklearn\base.py:1365: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

In [31]:
```
# Step 5: Evaluate the Models
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

def evaluate(y_true, y_pred, model_name):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    print(f"{model_name} Results:")
    print(f"  RMSE: {rmse:.2f}")
    print(f"  R² Score: {r2:.2f}\n")

evaluate(y_test, lr_preds, "Linear Regression")
evaluate(y_test, rf_preds, "Random Forest Regression")
```

```
Linear Regression Results:
  RMSE: 4.20
  R² Score: 0.80

Random Forest Regression Results:
  RMSE: 4.99
  R² Score: 0.71
```

In [33]:
```
# Visual Comparison
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, label="Actual", color='blue', alpha=0.4)
plt.scatter(X_test, lr_preds, label="Linear Regression", color='red', alpha=0.5)
plt.scatter(X_test, rf_preds, label="Random Forest", color='green', alpha=0.5)
plt.xlabel("Distance (km)")
plt.ylabel("Fare ($)")
plt.title("Actual vs Predicted Fares")
plt.legend()
plt.grid(True)
plt.show()
```


Actual vs Predicted Fares
Fare ($)
Distance (km)
Actual
Linear Regression
Random Forest

In [ ]: