In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

In [3]:

```
trips = pd.read_csv('C:\\Users\\CHETAN\\OneDrive\\Desktop\\final\\uber.csv')
#Time information is presented in two columns (key and pickup_datetime).
#data in the first column (Unnamed:0) will not be helpful to our analytics.
#removing the column 'Unnamed:0' and the column named 'key'.
del trips['Unnamed: 0']
del trips['key']
# final print
trips
```

Out[3]:

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

200000 rows × 7 columns

In [4]:

```
date_st = [dates.strip("UTC ") for dates in trips['pickup_datetime']]
trips['pickup_datetime'] = [datetime.strptime(dates, '%Y-%m-%d %H:%M:%S') for dates in date
trips['pickup_datetime'] = trips['pickup_datetime'].dt.date

# final print
trips
```

Out[4]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dro
0	7.5	2015-05-07	-73.999817	40.738354	-73.999512	
1	7.7	2009-07-17	-73.994355	40.728225	-73.994710	
2	12.9	2009-08-24	-74.005043	40.740770	-73.962565	
3	5.3	2009-06-26	-73.976124	40.790844	-73.965316	
4	16.0	2014-08-28	-73.925023	40.744085	-73.973082	
...	...	...	...	...	...	
199995	3.0	2012-10-28	-73.987042	40.739367	-73.986525	
199996	7.5	2014-03-14	-73.984722	40.736837	-74.006672	
199997	30.9	2009-06-29	-73.986017	40.756487	-73.858957	
199998	14.5	2015-05-20	-73.997124	40.725452	-73.983215	
199999	14.1	2010-05-15	-73.984395	40.720077	-73.985508	

200000 rows × 7 columns

In [5]:

```
# sorting the dataframe 'trips' based on the pickup_datetime date ascending and print the h
trips.sort_values(by='pickup_datetime', ascending=True, inplace=True, ignore_index=True)
trips.head(10)
```

Out[5]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_l
0	6.10	2009-01-01	-73.975759	40.749823	-73.982534	40.
1	10.10	2009-01-01	-73.982492	40.757212	-73.997370	40.
2	9.30	2009-01-01	-73.998807	40.713696	-73.993580	40.
3	7.80	2009-01-01	-73.980338	40.766303	-73.975158	40.
4	8.50	2009-01-01	-73.972600	40.749377	-73.981393	40.
5	8.50	2009-01-01	-73.981918	40.779456	-73.957685	40.
6	7.70	2009-01-01	-73.962266	40.779096	-73.975849	40.
7	32.65	2009-01-01	-73.872978	40.774098	-73.982055	40.
8	4.50	2009-01-01	-73.988440	40.740365	-73.986823	40.
9	4.60	2009-01-01	-73.965825	40.754429	-73.972814	40.

In [6]:

```
#Calculating distances between the pick-up and drop-off locations.
from math import sqrt

lat1 = trips['pickup_latitude']
lon1 = trips['pickup_longitude']
lat2 = trips['dropoff_latitude']
lon2 = trips['dropoff_longitude']
trips['distance'] = np.sqrt((lat1 - lat2)**2 + (lon1 - lon2)**2)
trips.tail(10)
```

Out[6]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dro
199990	7.00	2015-06-30	-73.964020	40.777203	-73.980034	
199991	20.50	2015-06-30	-73.967888	40.792416	-74.003708	
199992	75.54	2015-06-30	-73.703262	40.653118	-73.703285	
199993	18.50	2015-06-30	-73.991684	40.754646	-73.948006	
199994	6.00	2015-06-30	-73.978477	40.783051	-73.970085	
199995	11.50	2015-06-30	-73.961136	40.756756	-73.982857	
199996	8.50	2015-06-30	-73.955315	40.804562	-73.942322	
199997	4.50	2015-06-30	-73.963081	40.766251	-73.969421	
199998	9.00	2015-06-30	-74.005302	40.745792	-73.980911	
199999	9.50	2015-06-30	-73.982468	40.772266	-73.976784	

In [7]:

```
# Selecting all the records in January 2014 and store it in a variable called 'trip_jan14'.
start = pd.to_datetime("2014-01-01").date()
end = pd.to_datetime("2014-01-31").date()
trip_jan14 = trips.loc[(trips['pickup_datetime'] >= start) & (trips['pickup_datetime'] <= e
# your final print
trip_jan14
```

Out[7]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dro
156266	8.5	2014-01-01	-73.995383	40.720680	-73.995978	
156267	4.0	2014-01-01	-73.981798	40.765092	-73.976987	
156268	26.0	2014-01-01	-73.976075	40.759432	-74.007680	
156269	22.5	2014-01-01	-73.982268	40.745457	-74.004782	
156270	5.5	2014-01-01	-73.970427	40.752365	-73.981125	
...	...	...	...	...	...	
158755	14.0	2014-01-31	-73.962542	40.772987	-73.977230	
158756	8.0	2014-01-31	-73.982534	40.756929	-73.987059	
158757	6.0	2014-01-31	-73.982652	40.745070	-73.973608	
158758	15.0	2014-01-31	-74.006410	40.743883	-73.987823	
158759	4.5	2014-01-31	-73.971601	40.787764	-73.971601	

2494 rows × 8 columns

In [8]:

```
#excluding rows from the variable 'trip_jan14' that will be considered outliers. The outlie
fare_low = trip_jan14['fare_amount'].quantile(0.05)
fare_hi = trip_jan14['fare_amount'].quantile(0.95)
dist_low = trip_jan14['distance'].quantile(0.05)
dist_hi = trip_jan14['distance'].quantile(0.95)

trip_outliers = trip_jan14[(trip_jan14["distance"] < dist_hi) & (trip_jan14["distance"] > d
trip_filtered = trip_outliers
# your final print
trip_filtered
```

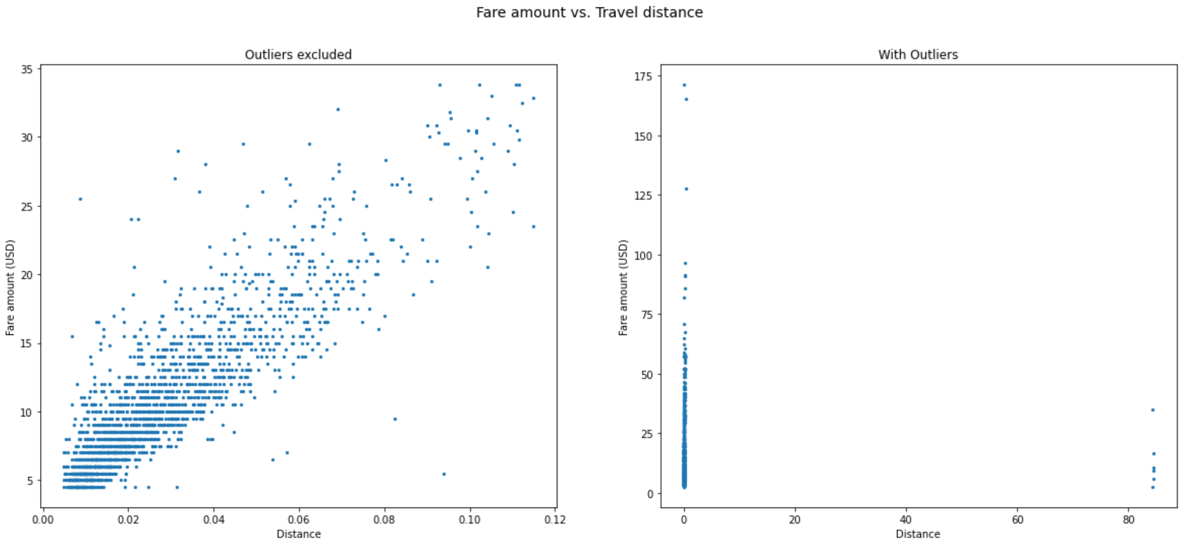
Out[8]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dro
156266	8.5	2014-01-01	-73.995383	40.720680	-73.995978	
156268	26.0	2014-01-01	-73.976075	40.759432	-74.007680	
156269	22.5	2014-01-01	-73.982268	40.745457	-74.004782	
156270	5.5	2014-01-01	-73.970427	40.752365	-73.981125	
156271	17.0	2014-01-01	-73.945873	40.801373	-73.973143	
...	...	...	...	...	...	
158754	8.5	2014-01-31	-73.967885	40.763388	-73.955567	
158755	14.0	2014-01-31	-73.962542	40.772987	-73.977230	
158756	8.0	2014-01-31	-73.982534	40.756929	-73.987059	
158757	6.0	2014-01-31	-73.982652	40.745070	-73.973608	
158758	15.0	2014-01-31	-74.006410	40.743883	-73.987823	

2130 rows × 8 columns

In [9]:

```
#Data Visualization
#Creating two scatter plots for the fare amount in y-axis and trip distances in x-axis usin
fig, ax1 = plt.subplots(1,2, figsize=(20,8))
trip_filtered.plot.scatter(x='distance', y='fare_amount', s=5, ax=ax1[0], title="Outliers e
trip_jan14.plot.scatter(x='distance', y='fare_amount', s=5, ax=ax1[1], title="With Outliers
plt.suptitle("Fare amount vs. Travel distance", fontsize=14)
plt.show()
```



In [10]:

```
corr = trip_filtered.corr()

corr.style.background_gradient(cmap='BuGn')
```

Out[10]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
fare_amount	1.000000	0.138584	-0.079318	0.241897	-0.118203
up_longitude	0.138584	1.000000	0.383908	0.510784	0.220253
ckup_latitude	-0.079318	0.383908	1.000000	0.156126	0.532864
off_longitude	0.241897	0.510784	0.156126	1.000000	0.404568
poff_latitude	-0.118203	0.220253	0.532864	0.404568	1.000000
senger_count	0.002507	0.010946	0.023809	0.012326	0.028554
distance	0.873414	0.177867	-0.075776	0.331917	-0.069195

In [13]:

```
X = trip_filtered['distance'].values.reshape(-1, 1)        #Independent Variable
y = trip_filtered['fare_amount'].values.reshape(-1, 1)     #Dependent Variable
```

In [15]:

```
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
y_std = std.fit_transform(y)
print(y_std)
```

```
[[-0.36607157]
 [ 2.88777184]
 [ 2.23700316]
 ...
 [-0.45903852]
 [-0.83090634]
 [ 0.84249884]]
```

In [16]:

```
x_std = std.fit_transform(X)
print(x_std)
```

```
[[-0.58699396]
 [ 0.48224865]
 [ 0.88140059]
 ...
 [-1.06960337]
 [-0.81087549]
 [-0.24328076]]
```

In [17]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_std, y_std, test_size=0.2, random_sta
```

In [18]:

```
from sklearn.linear_model import LinearRegression
l_reg = LinearRegression()
l_reg.fit(X_train, y_train)

print("Training set score: {:.2f}".format(l_reg.score(X_train, y_train)))
print("Test set score: {:.7f}".format(l_reg.score(X_test, y_test)))
```

```
Training set score: 0.76
Test set score: 0.7742213
```

In [20]:

```
y_pred = l_reg.predict(X_test)
df = {'Actual': y_test, 'Predicted': y_pred}
```

In [21]:

```
df
```

Out[21]:

```
{'Actual': array([[ 1.02843275],
        [-0.36607157],
        [-0.83090634],
        [ 0.47063102],
        [-0.9238733 ],
        [-0.55200548],
        [ 0.37766407],
        [-1.10980721],
        [-1.10980721],
        [-0.9238733 ],
        [-0.18013766],
        [-0.55200548],
        [ 0.47063102],
        [-0.27310461],
        [ 1.02843275],
        [-0.36607157],
        [-1.01684025],
        [-0.36607157],
```

In [22]:

```
!pip install tabulate
```

```
Collecting tabulate
  Downloading tabulate-0.8.10-py3-none-any.whl (29 kB)
Installing collected packages: tabulate
Successfully installed tabulate-0.8.10
```

In [23]:

```
from tabulate import tabulate
print(tabulate(df, headers = 'keys', tablefmt = 'psql'))
```

```
+-------------+-------------+
|      Actual |   Predicted |
|-------------+-------------|
|  1.02843    | -0.184749   |
| -0.366072   | -0.379188   |
| -0.830906   | -0.469004   |
|  0.470631   |  0.45027    |
| -0.923873   | -0.867893   |
| -0.552005   | -0.608699   |
|  0.377664   |  0.317061   |
| -1.10981    | -0.8609     |
| -1.10981    | -0.766916   |
| -0.923873   | -0.856356   |
| -0.180138   | -0.736829   |
| -0.552005   | -0.0609121  |
|  0.470631   |  0.801232   |
| -0.273105   |  0.131945   |
|  1.02843    |  0.0469107  |
| -0.366072   | -0.293227   |
```

In [24]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
#print('Mean Absolute % Error:', metrics.mean_absolute_percentage_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.33428235687706626
Mean Squared Error: 0.21328011425528853
Root Mean Squared Error: 0.4618226004163163
```

In [25]:

```
print(l_reg.intercept_)
print(l_reg.coef_)
```

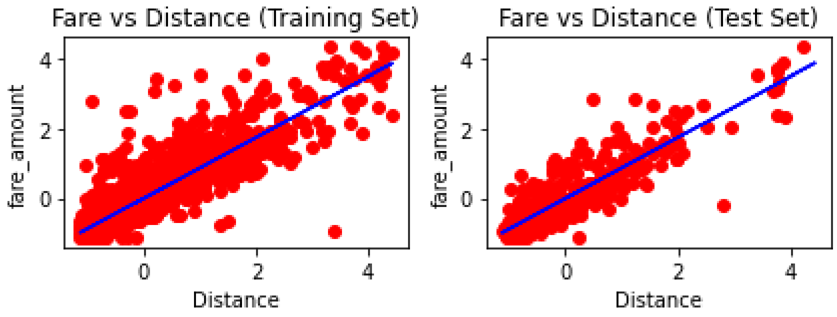
```
[0.0048171]
[[0.87448463]]
```

In [26]:

```
plt.subplot(2, 2, 1)
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.title("Fare vs Distance (Training Set)")
plt.ylabel("fare_amount")
plt.xlabel("Distance")

plt.subplot(2, 2, 2)
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.ylabel("fare_amount")
plt.xlabel("Distance")
plt.title("Fare vs Distance (Test Set)")


plt.tight_layout()
plt.rcParams["figure.figsize"] = (32,22)
plt.show()
```



In [ ]: