# Chicago Taxi Analysis

## For this project, we are analyzing the data of Chicago taxi service to draw conclusions to some qustions.

Below is the data and some visuals to answer each question.

In [1]:

```
# Import the CSV file here and show the table without any changes
import pandas as pd
import numpy as np

taxiData = pd.read_csv("chicago_taxi_trips_2016_01.csv")
taxiData.head()
```

Out[1]:

| | taxi_id | trip_start_timestamp | trip_end_timestamp | trip_seconds | trip_miles | pickup_census_tract | dropo |
|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 2016-1-13 06:15:00 | 2016-1-13 06:15:00 | 180.0 | 0.40 | NaN | |
| **1** | 2776.0 | 2016-1-22 09:30:00 | 2016-1-22 09:45:00 | 240.0 | 0.70 | NaN | |
| **2** | 3168.0 | 2016-1-31 21:30:00 | 2016-1-31 21:30:00 | 0.0 | 0.00 | NaN | |
| **3** | 4237.0 | 2016-1-23 17:30:00 | 2016-1-23 17:30:00 | 480.0 | 1.10 | NaN | |
| **4** | 5710.0 | 2016-1-14 05:45:00 | 2016-1-14 06:00:00 | 480.0 | 2.71 | NaN | |

In [2]:

```
value=taxiData['trip_miles'].mean()
print(value)
value1=taxiData['trip_seconds'].mean()
print(value1)
value2=taxiData['pickup_census_tract'].mean()
print(value2)
value3=taxiData['dropoff_census_tract'].mean()
print(value3)
value4=taxiData['trip_total'].mean()
print(value4)
value5=taxiData['fare'].mean()
print(value5)
value6=taxiData['tips'].mean()
print(value6)
value7=taxiData['tolls'].mean()
print(value7)
value8=taxiData['extras'].mean()
print(value8)
```

```
2.8727017026125337
653.442181752938
nan
516.8220157750194
15.621889226697302
13.153964152301748
1.5151068196686905
```

```
0.004308201799537096
0.9484849850976609
```

In [3]:

```
taxiData['trip_miles'].fillna(value=taxiData['trip_miles'].mean(), inplace=True)
value1=taxiData['trip_miles'].mean()
print(value1)
```

```
2.872701702612533
```

In [4]:

```
taxiData['fare'].fillna(value=taxiData['fare'].mean(), inplace=True)
value1=taxiData['fare'].mean()
print(value1)
```

```
13.153964152301747
```

In [5]:

```
taxiData['pickup_community_area'].fillna(value=taxiData['pickup_community_area'].median
value1=taxiData['pickup_community_area'].mean()
print(value1)
```

```
23.016217562968805
```

In [6]:

```
print(taxiData['trip_miles'].value_counts())
```

```
0.00      450257
0.10       55541
0.80       55338
1.00       54697
0.90       53136
           ...  
38.82          1
16.06          1
10.57          1
530.00         1
14.64          1
Name: trip_miles, Length: 2849, dtype: int64
```

In [7]:

```
import seaborn as sns
taxiData= taxiData[taxiData['trip_miles'] != 0]
value=taxiData['trip_miles'].mean()
print(value)
value1=taxiData['trip_seconds'].mean()
print(value1)
value2=taxiData['pickup_census_tract'].mean()
print(value2)
value3=taxiData['dropoff_census_tract'].mean()
print(value3)
value4=taxiData['trip_total'].mean()
print(value4)
value5=taxiData['fare'].mean()
print(value5)
value6=taxiData['tips'].mean()
print(value6)
value7=taxiData['tolls'].mean()
print(value7)
value8=taxiData['extras'].mean()
print(value8)
```

```
3.9028925439927207
762.7667393593381
nan
520.5788336618923
16.015848389513305
13.530489269224667
1.4992974783359871
0.0031798617305976917
0.9828412928507739
```

In [8]:

```
taxiData.pop('pickup_census_tract')
taxiData.pop('dropoff_census_tract')
taxiData.pop('pickup_latitude')
taxiData.pop('pickup_longitude')
taxiData.pop('dropoff_latitude')
taxiData.pop('dropoff_longitude')
taxiData.pop('pickup_community_area')
taxiData.pop('dropoff_community_area')
taxiData.head()
```

Out[8]:

| | taxi_id | trip_start_timestamp | trip_end_timestamp | trip_seconds | trip_miles | fare | tips | tolls | extras |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 2016-1-13 06:15:00 | 2016-1-13 06:15:00 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 |
| **1** | 2776.0 | 2016-1-22 09:30:00 | 2016-1-22 09:45:00 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 |
| **3** | 4237.0 | 2016-1-23 17:30:00 | 2016-1-23 17:30:00 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 |
| **4** | 5710.0 | 2016-1-14 05:45:00 | 2016-1-14 06:00:00 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 |
| **5** | 1987.0 | 2016-1-8 18:15:00 | 2016-1-8 18:45:00 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 |

In [9]:

```
taxiData['cash_pay'] = (taxiData["payment_type"] == 'Cash')
taxiData['cash_pay'] = taxiData['cash_pay'].astype(int)
taxiData.pop('payment_type')
taxiData.pop('company')
taxiData.head()
```

Out[9]:

| | taxi_id | trip_start_timestamp | trip_end_timestamp | trip_seconds | trip_miles | fare | tips | tolls | extras |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 2016-1-13 06:15:00 | 2016-1-13 06:15:00 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 |
| **1** | 2776.0 | 2016-1-22 09:30:00 | 2016-1-22 09:45:00 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 |
| **3** | 4237.0 | 2016-1-23 17:30:00 | 2016-1-23 17:30:00 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 |
| **4** | 5710.0 | 2016-1-14 05:45:00 | 2016-1-14 06:00:00 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 |
| **5** | 1987.0 | 2016-1-8 18:15:00 | 2016-1-8 18:45:00 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 |

In [10]:

```
taxiData[['trip_date','trip_start_time']] = taxiData['trip_start_timestamp'].str.split(
taxiData[['trip_date_end','trip_end_time']] = taxiData['trip_end_timestamp'].str.split(
taxiData.pop('trip_start_timestamp')
taxiData.pop('trip_end_timestamp')
taxiData.head()
```

Out[10]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | trip_date | trip_start_t |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 2016-1-13 | 06:1 |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 2016-1-22 | 09:3 |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 2016-1-23 | 17:3 |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 2016-1-14 | 05:4 |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 2016-1-8 | 18:1 |

In [11]:

```
taxiData[['year','month','day']] = taxiData['trip_date'].str.split('-', expand=True)
taxiData[['year1','month1','day1']] = taxiData['trip_date_end'].str.split('-', expand=T
taxiData.pop('trip_date_end')
taxiData.pop('trip_date')
taxiData.head()
```

Out[11]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | trip_start_time | trip_e |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 06:15:00 | |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 09:30:00 | |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 17:30:00 | |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 05:45:00 | |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 18:15:00 | |

In [12]:

```
print(taxiData['year'].value_counts())
print(taxiData['year1'].value_counts())
print(taxiData['month'].value_counts())
print(taxiData['month1'].value_counts())
print(taxiData['day'].value_counts())
print(taxiData['day1'].value_counts())
```

```
2016    1255548
Name: year, dtype: int64
2016    1255548
Name: year1, dtype: int64
1    1255548
Name: month, dtype: int64
1     1255363
2         180
7           2
12          1
10          1
8           1
Name: month1, dtype: int64
15    50568
22    50002
```

```
1     49304
29    48284
21    47278
8     47072
13    45986
28    45358
14    44910
20    44885
12    43499
19    43014
16    42859
27    42097
11    41554
26    40130
23    39801
7     39663
25    39001
9     38046
6     37904
30    36935
17    36823
5     36086
2     34910
4     34045
18    32986
10    31743
24    31664
31    29767
3     29374
Name: day, dtype: int64
15    50396
22    49817
1     49230
29    48127
21    47232
8     46843
13    45929
28    45248
14    44911
20    44871
12    43505
19    42961
16    42785
27    42076
11    41548
26    40151
23    39827
7     39606
25    38999
9     38169
6     37927
17    37141
30    36989
5     36038
2     34868
4     34171
18    33014
24    31913
10    31850
31    29981
```

```
3     29425
Name: day1, dtype: int64
```

In [13]:

```
taxiData.pop('year')
taxiData.pop('year1')
taxiData.pop('month')
taxiData.pop('month1')
taxiData.head()
```

Out[13]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | trip_start_time | trip_e |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 06:15:00 | |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 09:30:00 | |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 17:30:00 | |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 05:45:00 | |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 18:15:00 | |

In [14]:

```
taxiData[['start_hour','start_minute','start_second']] = taxiData['trip_start_time'].st
taxiData[['end_hour','end_minute','end_second']] = taxiData['trip_end_time'].str.split(
taxiData.pop('trip_start_time')
taxiData.pop('trip_end_time')
taxiData.head()
```

Out[14]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | day | day1 | start_hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 13 | 13 | 0 |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 22 | 22 | 0 |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 23 | 23 | 1 |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 14 | 14 | 0 |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 8 | 8 | 1 |

In [15]:

```
print(taxiData['start_hour'].value_counts())
print(taxiData['end_hour'].value_counts())
print(taxiData['start_minute'].value_counts())
print(taxiData['end_minute'].value_counts())
print(taxiData['start_second'].value_counts())
print(taxiData['end_second'].value_counts())
```

```
18    88260
19    87109
17    78908
20    73902
16    69505
21    63730
15    62147
13    61353
14    60508
12    60256
```

```
09    59960
22    57062
11    54988
10    54957
08    51672
23    47941
00    43584
01    40450
02    35850
07    31138
03    27162
04    17070
06    15708
05    12328
Name: start_hour, dtype: int64
19    90530
18    86938
20    78351
17    75386
16    66888
21    65871
13    61900
15    60372
14    59684
12    59407
09    59399
22    58895
10    57638
11    54621
23    50194
08    46827
00    44416
01    40911
02    36810
03    29008
07    27152
04    18219
06    13709
05    12422
Name: end_hour, dtype: int64
45    321288
00    315118
15    310479
30    308663
Name: start_minute, dtype: int64
00    322297
45    311679
30    310878
15    310694
Name: end_minute, dtype: int64
00    1255548
Name: start_second, dtype: int64
00    1255548
Name: end_second, dtype: int64
```

In [16]:

```
taxiData.pop('start_second')
taxiData.pop('end_second')
taxiData.head()
```

Out[16]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | day | day1 | start_hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 13 | 13 | 0 |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 22 | 22 | 0 |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 23 | 23 | 1 |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 14 | 14 | 0 |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 8 | 8 | 1 |

In [17]:

```
taxiData.rename(columns = {'day':'start_day', 'day1':'end_day'}, inplace = True)
taxiData.head()
```

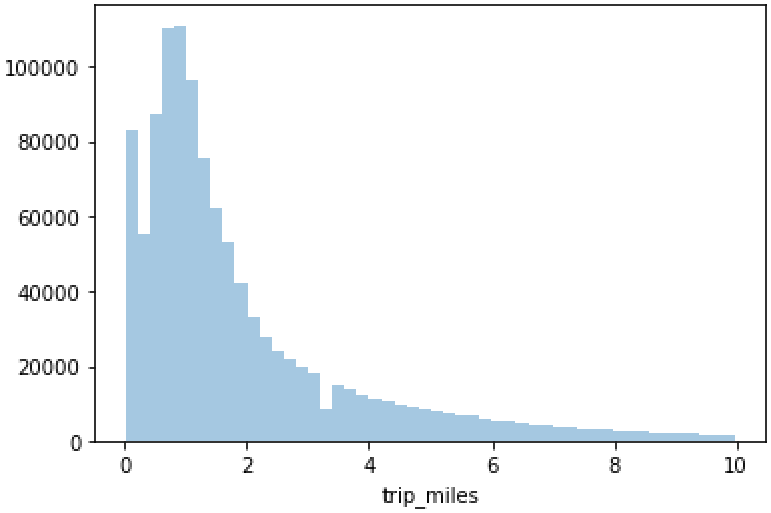
Out[17]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | start_day | end_day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 13 | 13 |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 22 | 22 |
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 23 | 23 |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 14 | 14 |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 8 | 8 |

In [18]:

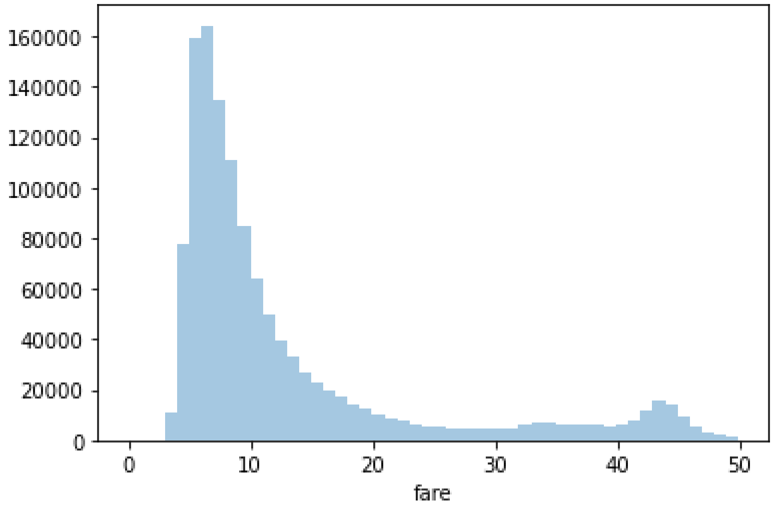
```
sns.distplot(taxiData[taxiData["trip_miles"]<10]["trip_miles"], kde=False);
```

```
C:\Users\neelp\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [19]:

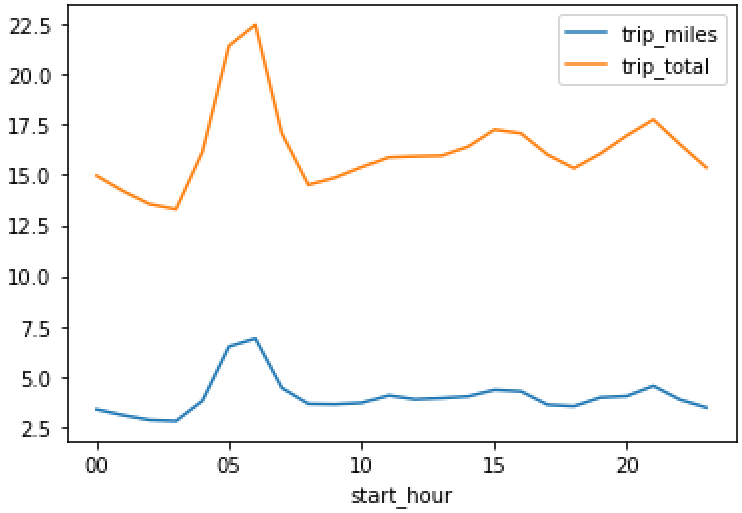
```
sns.distplot(taxiData[taxiData["fare"]<50]["fare"], kde=False);
```



In [20]:

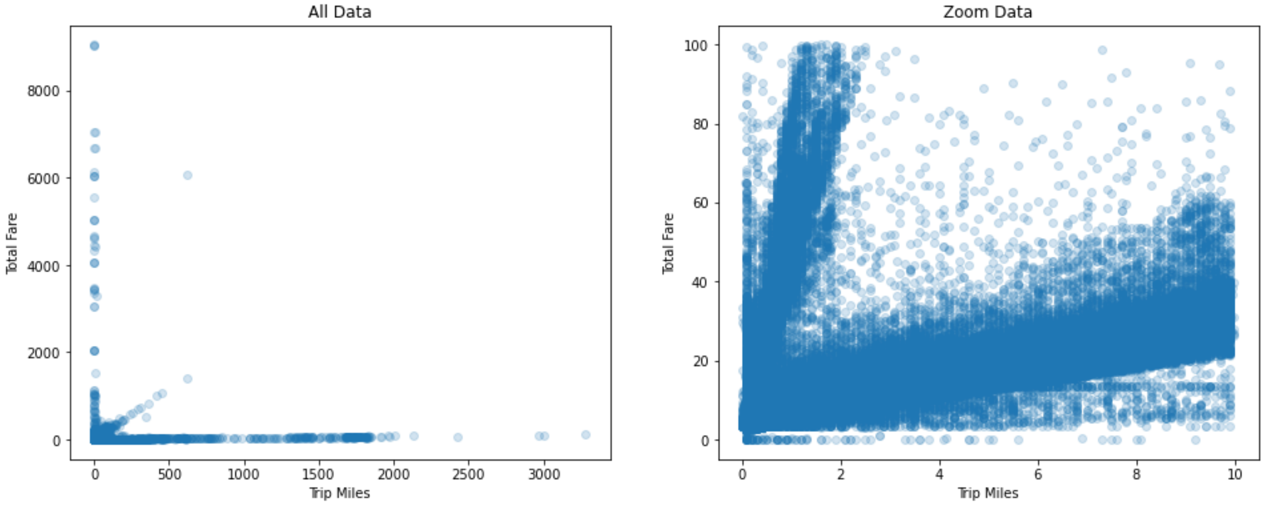
```
taxiData.groupby('start_hour')['trip_miles','trip_total'].mean().plot();
```

```
C:\Users\neelp\AppData\Local\Temp/ipykernel_14948/3580490451.py:1: FutureWarning: Indexi
ng with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use
a list instead.
  taxiData.groupby('start_hour')['trip_miles','trip_total'].mean().plot();
```



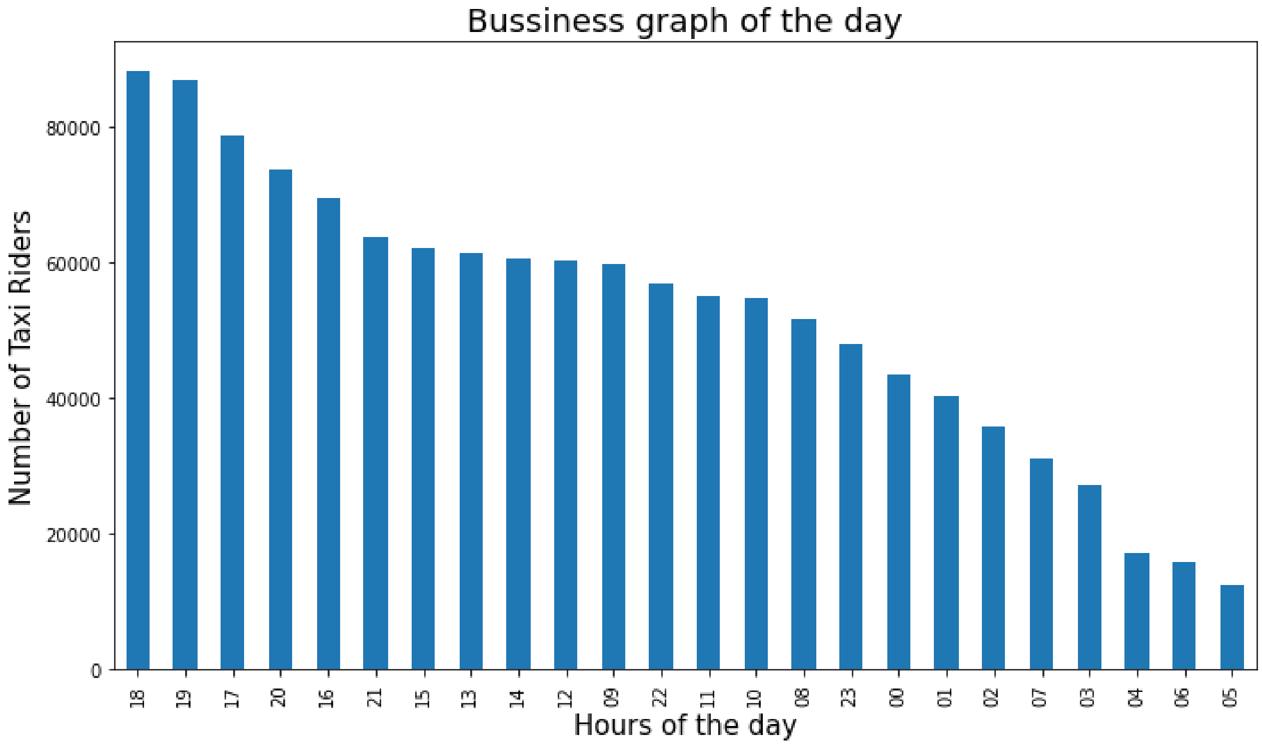
In [21]:

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1,2, figsize=(16,6))
axs[0].scatter(taxiData["trip_miles"],taxiData["trip_total"],alpha=0.2)
axs[0].set_title("All Data")
axs[0].set_xlabel("Trip Miles")
axs[0].set_ylabel("Total Fare");

zoom = ((taxiData["trip_miles"]<10)&(taxiData["trip_total"]<100))
axs[1].scatter(taxiData[zoom]["trip_miles"],taxiData[zoom]["trip_total"],alpha=0.2)
axs[1].set_title("Zoom Data")
axs[1].set_xlabel("Trip Miles")
axs[1].set_ylabel("Total Fare");
```



In [22]:

```
taxiData1 = taxiData
plt.figure(figsize=(10,6))
graph = taxiData1['start_hour'].value_counts().plot.bar()
plt.xlabel("Hours of the day", size=15)
plt.ylabel("Number of Taxi Riders", size=15)
plt.title("Bussiness graph of the day", size=18)
plt.tight_layout()
```



In [23]:

```
taxiData['M&F'] = ((taxiData["trip_miles"]<10)&(taxiData["trip_total"]<100))
taxiData['M&F'] = taxiData['M&F'].astype(int)
taxiData.head()
```

Out[23]:

| | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | start_day | end_day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 85.0 | 180.0 | 0.40 | 4.50 | 0.00 | 0.0 | 0.0 | 4.50 | 1 | 13 | 13 |
| **1** | 2776.0 | 240.0 | 0.70 | 4.45 | 4.45 | 0.0 | 0.0 | 8.90 | 0 | 22 | 22 |

|   | taxi_id | trip_seconds | trip_miles | fare | tips | tolls | extras | trip_total | cash_pay | start_day | end_day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 4237.0 | 480.0 | 1.10 | 7.00 | 0.00 | 0.0 | 0.0 | 7.00 | 1 | 23 | 23 |
| **4** | 5710.0 | 480.0 | 2.71 | 10.25 | 0.00 | 0.0 | 0.0 | 10.25 | 1 | 14 | 14 |
| **5** | 1987.0 | 1080.0 | 6.20 | 17.75 | 0.00 | 0.0 | 0.0 | 17.75 | 1 | 8 | 8 |

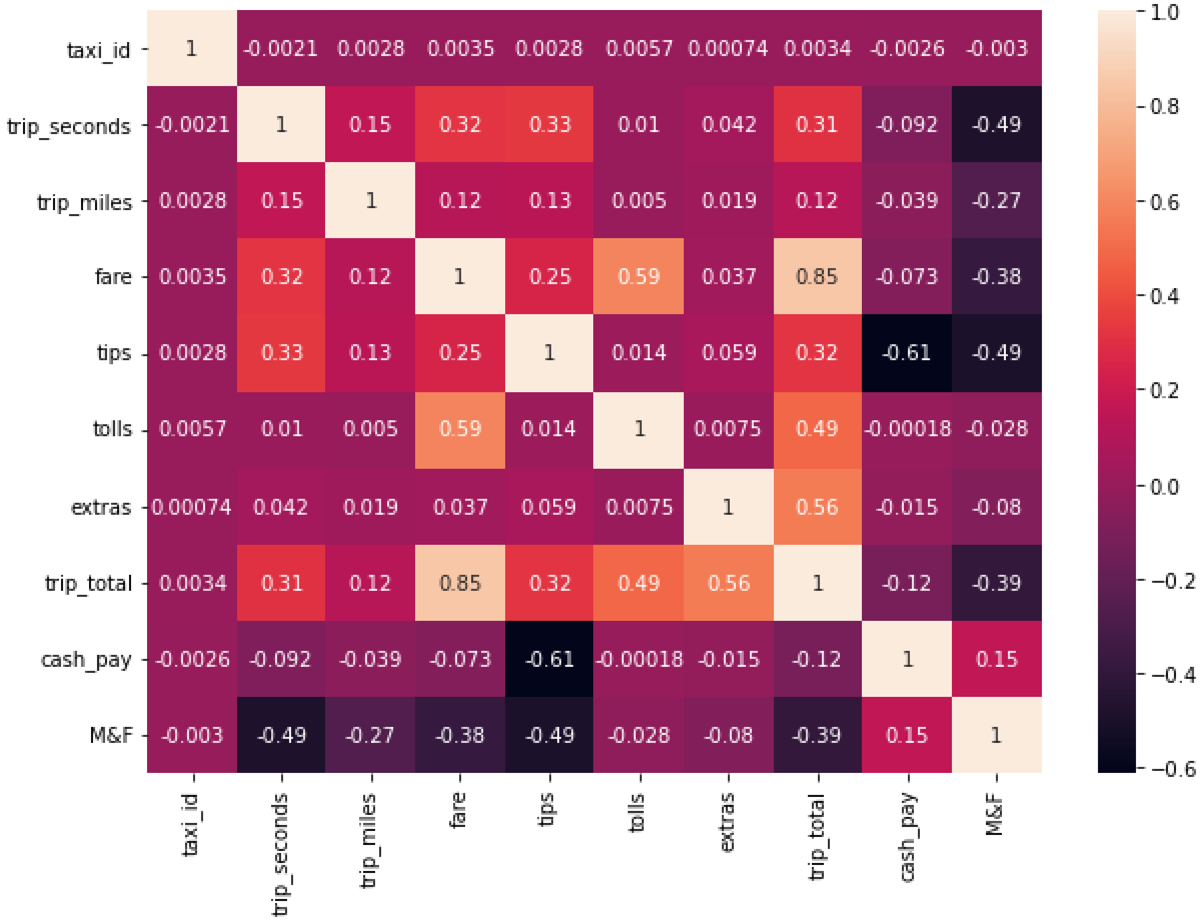
In [24]:

```
print(taxiData['M&F'].value_counts())
```

```
1    1117762
0     137786
Name: M&F, dtype: int64
```

In [25]:

```
plt.figure(figsize=(10,7))
sns.heatmap(taxiData.corr(), annot=True)
plt.show()
```



In [26]:

```
from sklearn.model_selection import train_test_split
taxiData2 = taxiData
```

In [27]:

```
taxiData2['trip_miles'].fillna(value=taxiData2['trip_miles'].mean(), inplace=True)
taxiData2['trip_total'].fillna(value=taxiData2['trip_total'].mean(), inplace=True)
```

In [28]:

```
taxiData2 = taxiData2.fillna(0)
```

In [29]:

```
# Xdata = taxiData2[['trip_miles', 'trip_total']]
# Xdata['trip_miles'].fillna(value=Xdata['trip_miles'].mean(), inplace=True)
# Xdata['trip_total'].fillna(value=Xdata['trip_total'].mean(), inplace=True)
# ytarget = taxiData2['M&F']
taxiData0 = taxiData2.to_numpy()
taxiData0 = taxiData0.astype(int)
print(taxiData0)
```

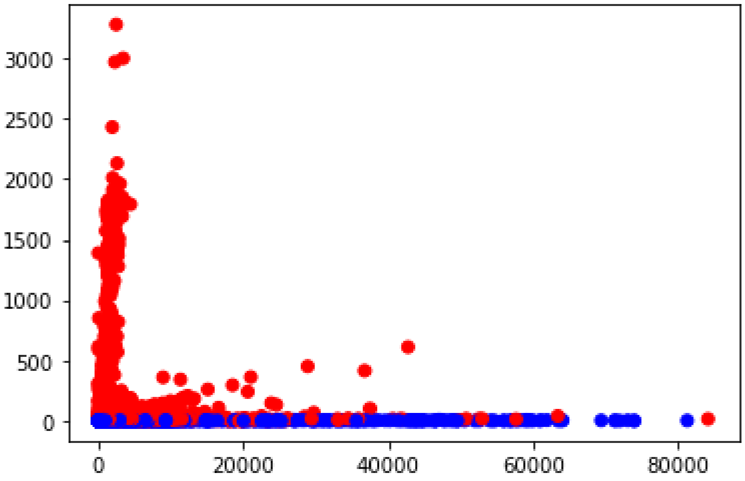
```
[[  85  180    0 ...    6   15    1]
 [2776  240    0 ...    9   45    1]
 [4237  480    1 ...   17   30    1]
 ...
 [1213 1380   17 ...    6   45    0]
 [1911  960    2 ...   12   45    1]
 [8206  360    2 ...    3   15    1]]
```

In [30]:

```
# Xtd = taxiData0[:, [1, 2]]
Xtd = taxiData0[:, 1:3]
ytt = taxiData['M&F']
X_train,X_test,y_train,y_test = train_test_split(Xtd, ytt, test_size = 0.20)
```

In [32]:

```
from sklearn import svm
from matplotlib.colors import ListedColormap
from sklearn import metrics

colors = np.array(['r' , 'b'])
plt.scatter(Xtd[:,0] , Xtd[:,1]  ,c = colors[ytt])
plt.show()
```



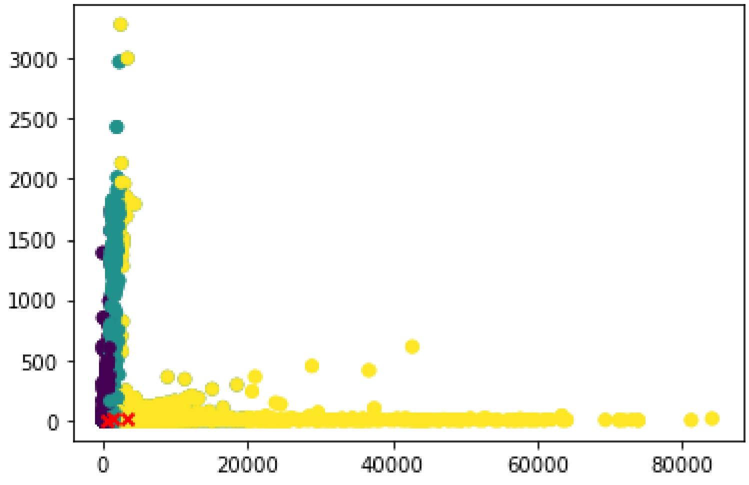
In [33]:

```
def meshGrid (x , y , h):
    '''x is data for x-axis meshgrid
       y is data for y-axis meshgrid
       h is stepsize
    '''
    x_min, x_max = x.min() - 1 , x.max() + 1
    y_min, y_max = y.min() - 1 , y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    return xx , yy
```

In [34]:

```
from sklearn.cluster import KMeans
# from sklearn.metrics.cluster import completeness_score
# from sklearn.metrics.cluster import homogeneity_score

cmap_light = ListedColormap(['#FBBBB9', '#82CAFF', '#5EFB6E'])
cmap_bold = ListedColormap(['#CA226B', '#2B65EC', '#387C44'])
cmap_test = ListedColormap(['#8E35EF', '#659EC7', '#FFFF00'])
cmap_predict1 = ListedColormap(['#8105ED', '#ED05CA', '#FA0505'])

y1_predict = KMeans(n_clusters = 3, random_state = 200).fit_predict(Xtd)
kmeans = KMeans(n_clusters = 3, init ='random', random_state = 200, verbose=True).fit(X

plt.scatter(Xtd[:,0], Xtd[:,1], c =y1_predict)
plt.scatter(Xtd[:,0], Xtd[:,1], c= kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],c = 'r',marker ='
plt.show()

xx6 , yy6 = meshGrid(Xtd[:,0], Xtd[:,1], 0.01)

Z6 = kmeans.predict(np.c_[xx6.ravel(), yy6.ravel()])
Z6 = Z6.reshape(xx6.shape)

plt.figure()
plt.contourf(xx6, yy6, Z6, cmap=cmap_light ,levels=[-1, 0, 1] ,alpha = 0.5)

# For plotting train and test and prediction separatley
plt.scatter(Xtd[:, 0], Xtd[:, 1], alpha=1.0,c = y1_predict, cmap=cmap_predict1 ,linewid
plt.xlim(xx6.min(), xx6.max())
plt.ylim(yy6.min(), yy6.max())

plt.show()
```

```
Initialization complete
Iteration 0, inertia 906536551384.0
Iteration 1, inertia 619871916941.754
Iteration 2, inertia 546211606461.69525
Iteration 3, inertia 500485165354.1886
Iteration 4, inertia 481268996636.3737
Iteration 5, inertia 470323557541.524
Iteration 6, inertia 460879197378.67706
Iteration 7, inertia 452765111151.3736
Iteration 8, inertia 445871314422.1643
Iteration 9, inertia 444006423135.20013
Iteration 10, inertia 440441861484.5365
Iteration 11, inertia 439208354953.2287
Iteration 12, inertia 438182653280.95135
Iteration 13, inertia 435512175173.4532
Iteration 14, inertia 434935132756.7486
Iteration 15, inertia 434548221386.2146
Iteration 16, inertia 433046778214.5932
Iteration 17, inertia 432535302622.65326
Iteration 18, inertia 432189284347.72736
Converged at iteration 18: strict convergence.
Initialization complete
Iteration 0, inertia 529357326201.0
Iteration 1, inertia 438010360244.4178
Iteration 2, inertia 433323126703.94385
Iteration 3, inertia 431531286990.70087
Iteration 4, inertia 431091814654.7114
Iteration 5, inertia 430773412473.3765
Converged at iteration 5: center shift 0.01734577434380537 within tolerance 34.813282539
41602.
Initialization complete
Iteration 0, inertia 698585404425.0
Iteration 1, inertia 585295451795.0052
Iteration 2, inertia 521349471005.04346
Iteration 3, inertia 494749067665.4828
Iteration 4, inertia 477517890999.3849
Iteration 5, inertia 467163220762.6799
Iteration 6, inertia 460879197378.67706
Iteration 7, inertia 452765111151.3736
Iteration 8, inertia 445871314422.16425
Iteration 9, inertia 444006423135.20026
Iteration 10, inertia 440441861484.5365
Iteration 11, inertia 439208354953.2287
Iteration 12, inertia 438182653280.95135
Iteration 13, inertia 435512175173.4532
Iteration 14, inertia 434935132756.7486
Iteration 15, inertia 434548221386.2146
Iteration 16, inertia 433046778214.5932
Iteration 17, inertia 432535302622.6534
Iteration 18, inertia 432189284347.72736
Converged at iteration 18: strict convergence.
Initialization complete
Iteration 0, inertia 781393370466.0
Iteration 1, inertia 519722636204.88544
Iteration 2, inertia 474019376080.3805
Iteration 3, inertia 461419318864.9844
Iteration 4, inertia 455156181711.0625
Iteration 5, inertia 450095771234.02014
Iteration 6, inertia 445862740942.65686
Iteration 7, inertia 444006423135.20013
Iteration 8, inertia 440441861484.5365
Iteration 9, inertia 439208354953.2287
Iteration 10, inertia 438182653280.95135
Iteration 11, inertia 435512175173.4532
Iteration 12, inertia 434935132756.7486
Iteration 13, inertia 434548221386.2146
Iteration 14, inertia 433046778214.5932
Iteration 15, inertia 432535302622.65326
Iteration 16, inertia 432189284347.72736
Converged at iteration 16: strict convergence.
Initialization complete
Iteration 0, inertia 502990009831.0
Iteration 1, inertia 464481459414.67596
Iteration 2, inertia 455156181711.0625
Iteration 3, inertia 450095771234.02014
Iteration 4, inertia 445862740942.65686
Iteration 5, inertia 444006423135.20026
Iteration 6, inertia 440441861484.5365
Iteration 7, inertia 439208354953.2287
Iteration 8, inertia 438182653280.95135
Iteration 9, inertia 435512175173.4532
Iteration 10, inertia 434935132756.7486
Iteration 11, inertia 434548221386.2146
Iteration 12, inertia 433046778214.5932
Iteration 13, inertia 432535302622.65326
Iteration 14, inertia 432189284347.72736
Converged at iteration 14: strict convergence.
Initialization complete
Iteration 0, inertia 909658706664.0
Iteration 1, inertia 624789785451.1768
Iteration 2, inertia 546211606461.69525
Iteration 3, inertia 500485165354.1886
Iteration 4, inertia 481268996636.3737
Iteration 5, inertia 470323557541.524
Iteration 6, inertia 460879197378.67706
Iteration 7, inertia 452765111151.3736
Iteration 8, inertia 445871314422.1643
Iteration 9, inertia 444006423135.20026
Iteration 10, inertia 440441861484.5365
Iteration 11, inertia 439208354953.2287
Iteration 12, inertia 438182653280.95135
Iteration 13, inertia 435512175173.4532
Iteration 14, inertia 434935132756.7486
Iteration 15, inertia 434548221386.2146
Iteration 16, inertia 433046778214.5932
Iteration 17, inertia 432535302622.65326
Iteration 18, inertia 432189284347.72736
Converged at iteration 18: strict convergence.
Initialization complete
Iteration 0, inertia 694146355981.0
Iteration 1, inertia 566850810421.5238
Iteration 2, inertia 512406608294.51556
Iteration 3, inertia 489821219667.0774
Iteration 4, inertia 477516757739.91516
Iteration 5, inertia 467163220762.6799
Iteration 6, inertia 460879197378.67706
Iteration 7, inertia 452765111151.3736
Iteration 8, inertia 445871314422.16425
Iteration 9, inertia 444006423135.20013
Iteration 10, inertia 440441861484.5365
Iteration 11, inertia 439208354953.2287
Iteration 12, inertia 438182653280.95135
Iteration 13, inertia 435512175173.4532
Iteration 14, inertia 434935132756.7486
Iteration 15, inertia 434548221386.2146
Iteration 16, inertia 433046778214.5932
Iteration 17, inertia 432535302622.65326
Iteration 18, inertia 432189284347.72736
Converged at iteration 18: strict convergence.
Initialization complete
Iteration 0, inertia 890675305353.0
Iteration 1, inertia 575619111298.624
Iteration 2, inertia 477997258658.8968
Iteration 3, inertia 461419318864.9844
Iteration 4, inertia 455156181711.0625
Iteration 5, inertia 450095771234.02014
Iteration 6, inertia 445862740942.65686
Iteration 7, inertia 444006423135.20026
Iteration 8, inertia 440441861484.5365
Iteration 9, inertia 439208354953.2287
Iteration 10, inertia 438182653280.95135
Iteration 11, inertia 435512175173.4532
Iteration 12, inertia 434935132756.7486
Iteration 13, inertia 434548221386.2146
Iteration 14, inertia 433046778214.5932
Iteration 15, inertia 432535302622.65326
Iteration 16, inertia 432189284347.72736
Converged at iteration 16: strict convergence.
Initialization complete
Iteration 0, inertia 557598350839.0
Iteration 1, inertia 479927206852.5112
Iteration 2, inertia 464108059542.9469
Iteration 3, inertia 457844092645.5693
Iteration 4, inertia 452763729276.1678
Iteration 5, inertia 445867731193.71216
Iteration 6, inertia 444006423135.20026
Iteration 7, inertia 440441861484.5365
Iteration 8, inertia 439208354953.2287
Iteration 9, inertia 438182653280.95135
Iteration 10, inertia 435512175173.4532
Iteration 11, inertia 434935132756.7486
Iteration 12, inertia 434548221386.2146
Iteration 13, inertia 433046778214.5932
Iteration 14, inertia 432535302622.65326
Iteration 15, inertia 432189284347.72736
Converged at iteration 15: strict convergence.
Initialization complete
Iteration 0, inertia 786043732248.0
Iteration 1, inertia 604015148360.9747
Iteration 2, inertia 532702683673.2222
Iteration 3, inertia 500484902073.17395
Iteration 4, inertia 481268996636.3737
Iteration 5, inertia 470323557541.524
Iteration 6, inertia 460879197378.67706
Iteration 7, inertia 452765111151.3736
Iteration 8, inertia 445871314422.1643
Iteration 9, inertia 444006423135.20026
Iteration 10, inertia 440441861484.5365
Iteration 11, inertia 439208354953.2287
Iteration 12, inertia 438182653280.95135
Iteration 13, inertia 435512175173.4532
Iteration 14, inertia 434935132756.7486
Iteration 15, inertia 434548221386.2146
Iteration 16, inertia 433046778214.5932
Iteration 17, inertia 432535302622.65326
Iteration 18, inertia 432189284347.72736
Converged at iteration 18: strict convergence.
```



```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_14948/3087625254.py in <module>
     16 plt.show()
     17 
---> 18 xx6 , yy6 = meshGrid(Xtd[:,0], Xtd[:,1], 0.01)
     19 
     20 Z6 = kmeans.predict(np.c_[xx6.ravel(), yy6.ravel()])

~\AppData\Local\Temp/ipykernel_14948/3462481497.py in meshGrid(x, y, h)
      6     x_min, x_max = x.min() - 1 , x.max() + 1
      7     y_min, y_max = y.min() - 1 , y.max() + 1
----> 8     xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h)
)
      9 
     10     return xx , yy

<__array_function__ internals> in meshgrid(*args, **kwargs)

~\anaconda3\lib\site-packages\numpy\lib\function_base.py in meshgrid(copy, sparse, index
ing, *xi)
   4299 
   4300     if copy:
-> 4301         output = [x.copy() for x in output]
   4302 
   4303     return output

~\anaconda3\lib\site-packages\numpy\lib\function_base.py in <listcomp>(.0)
   4299 
   4300     if copy:
-> 4301         output = [x.copy() for x in output]
   4302 
   4303     return output

MemoryError: Unable to allocate 20.1 TiB for an array with shape (328200, 8418200) and d
ata type float64
```

In [39]:

```
Xtd = taxiData[['trip_miles', 'trip_total']]
ytt = taxiData['M&F']
X_train,X_test,y_train,y_test = train_test_split(Xtd, ytt, test_size = 0.20)
```

In [40]:

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

Out[40]:

```
LinearRegression()
```

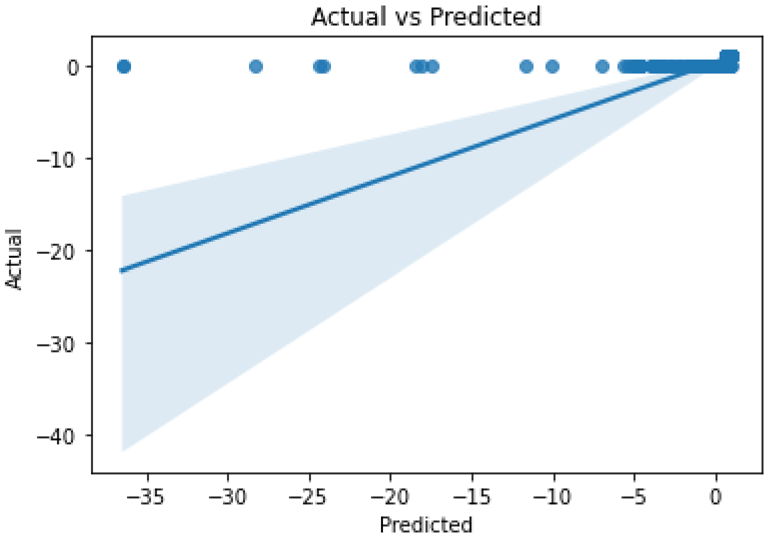
In [41]:

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
pred = regressor.predict(X_test)
print("MSE: ", mean_squared_error(pred, y_test))
print("MAE: ", mean_absolute_error(pred, y_test))
print("r2E: ", r2_score(pred, y_test))
```

```
MSE:  0.08843070276215886
MAE:  0.1542395842702467
r2E:  -1.4774490135570408
```

In [42]:

```
sns.regplot(x=pred, y=y_test)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Actual vs Predicted")
plt.show()
```



In [43]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_new = scaler.fit_transform(X_train)
X_test_new = scaler.transform(X_test)

regressor.fit(X_train_new, y_train)
pred1 = regressor.predict(X_test_new)
print("MSE: ", mean_squared_error(pred1, y_test))
print("MAE: ", mean_absolute_error(pred1, y_test))
print("r2E: ", r2_score(pred1, y_test))
```

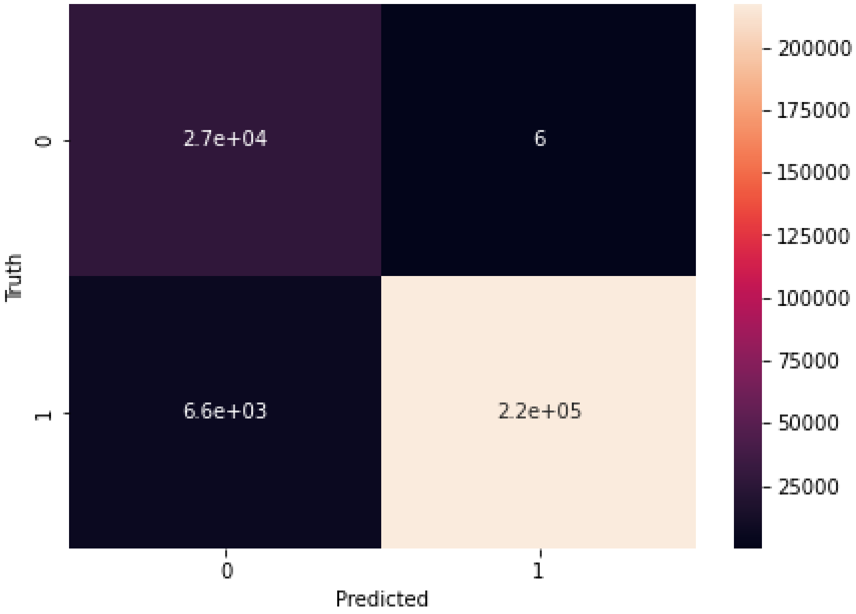
```
MSE:  0.0884307027621588
MAE:  0.15423958427024712
r2E:  -1.4774490135570812
```

In [44]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import seaborn as sn

model = GaussianNB()
model.fit(X_train, y_train)
model_predict = model.predict(X_test)

#Display the outcome of classification
print(metrics.classification_report(y_test, model_predict))
print(metrics.confusion_matrix(y_test, model_predict))
plt.figure(figsize=(7,5))
sn.heatmap(metrics.confusion_matrix(y_test, model_predict), annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()
```

```
              precision    recall  f1-score   support
```

```
           0       0.81      1.00      0.89     27292
           1       1.00      0.97      0.99    223818

    accuracy                           0.97    251110
   macro avg       0.90      0.99      0.94    251110
weighted avg       0.98      0.97      0.98    251110

[[ 27286      6]
 [  6551 217267]]
```



In [ ]:

```python
X_train,X_test,y_train,y_test = train_test_split(Xdata, ytarget, test_size = 0.20)
```

In [ ]:

```python
# from sklearn.ensemble import RandomForestClassifier
# clf = RandomForestClassifier(n_estimators= 20, random_state = 0)
# clf.fit(X_train, y_train)
# rf_rmse=np.sqrt(mean_squared_error(clf.predict(X_test), y_test))
# print("RMSE for Random Forest is ",rf_rmse)
```

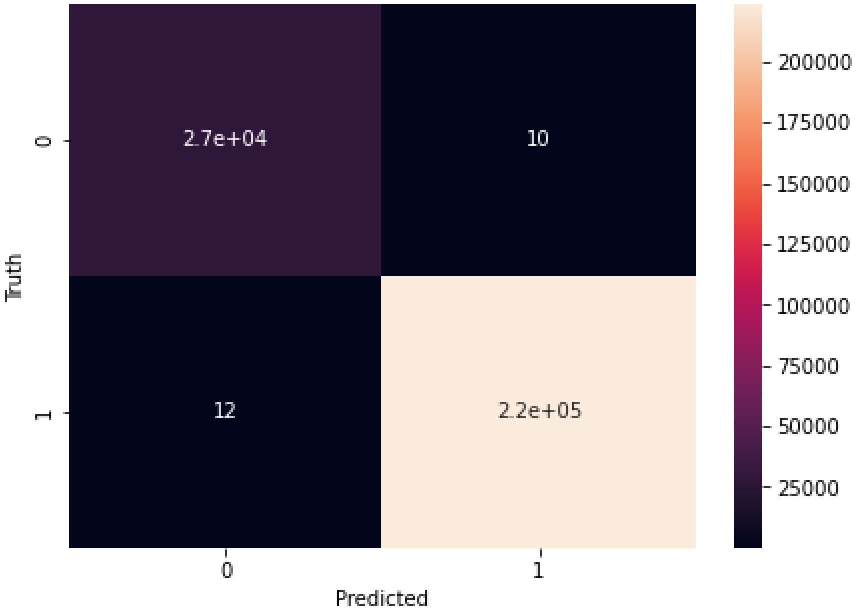
In [46]:

```python
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt1
from matplotlib.colors import ListedColormap
import seaborn as sn

cmap_light = ListedColormap(['#FBBBB9', '#5EFB6E', '#82CAFF'])
cmap_bold = ListedColormap(['#CA226B', '#387C44', '#2B65EC'])
cmap_test = ListedColormap(['#8E35EF', '#FFFF00', '#659EC7'])

#meshstep size parameter
h = 0.2

#KNN learner
model = KNeighborsClassifier(1)
```

```
#Fitting the data
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [50]:

```
cm = confusion_matrix(y_test, y_pred)
plt1.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt1.xlabel('Predicted')
plt1.ylabel('Truth')
plt1.show()

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     27292
           1       1.00      1.00      1.00    223818

    accuracy                           1.00    251110
   macro avg       1.00      1.00      1.00    251110
weighted avg       1.00      1.00      1.00    251110
```

In [69]:

```
taxiData9 = taxiData
taxiData9 = taxiData9.fillna(0)
```

In [70]:

```
X = taxiData9.drop('M&F', axis=1)
y = taxiData9['M&F']
```

In [73]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2)
```

In [74]:

```
from sklearn.linear_model import Ridge
ridgereg = Ridge(alpha=0.1, normalize=True)
```

```
ridgereg.fit(X_train, y_train)
y_pred = ridgereg.predict(X_test)
```

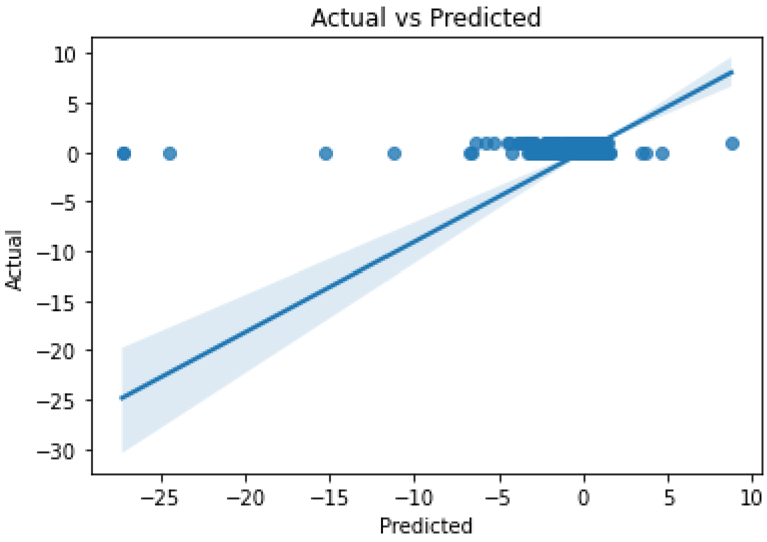
In [75]:

```
print("R-Square Value",r2_score(y_test,y_pred),"\n")
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_pred),"\n")
print ("mean_squared_error : ",metrics.mean_squared_error(y_test, y_pred),"\n")
print ("root_mean_squared_error : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

```
R-Square Value 0.3891365179564942

mean_absolute_error : 0.11989868849672725

mean_squared_error :  0.059372119981769673

root_mean_squared_error :  0.2436639488758435
```

In [76]:

```
sns.regplot(x=y_pred, y=y_test)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Actual vs Predicted")
plt.show()
```



In [66]:

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Out[66]:

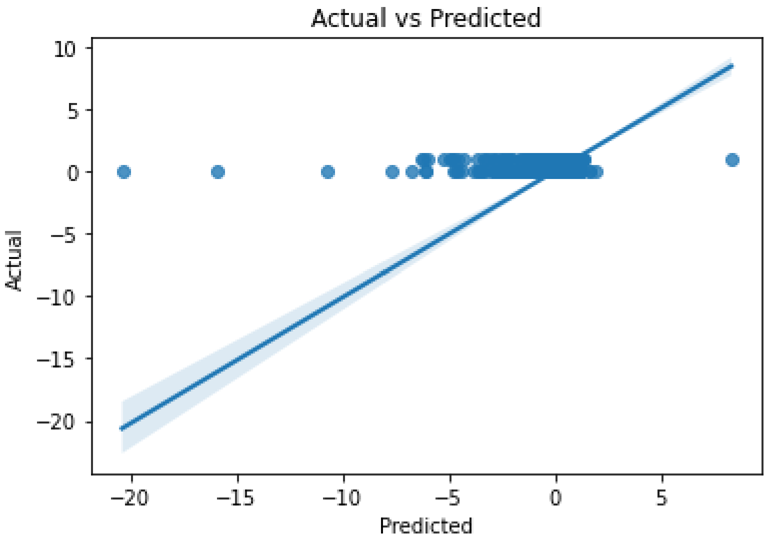
```
LinearRegression()
```

In [67]:

```
from sklearn.metrics import r2_score
from sklearn import metrics
y_pred1 = linreg.predict(X_test)
print("R-Square Value",r2_score(y_test,y_pred))
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_pred1))
print ("mean_squared_error : ",metrics.mean_squared_error(y_test, y_pred1))
print ("root_mean_squared_error : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred1)
R-Square Value 0.4458265256917868
mean_absolute_error : 0.11924469894160818
mean_squared_error :  0.05362709233607836
root_mean_squared_error :  0.23157524119835945
```

In [68]:

```
sns.regplot(x=y_pred1, y=y_test)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Actual vs Predicted")
plt.show()
```



In [ ]: