In [1]:
```
import mysql.connector as sql

import pandas as pd
import datetime
```

In [2]:
```
db_connection = sql.connect(host='Dannys-MacBook-Pro.local', database='F
inalProject', user='root', password='Aguilard123')
```

In [104]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjan1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;


    return queryList
```

In [105]:
```
listJan1 = []
listJan1 = test_fetchmany()
jan1 = pd.DataFrame(listJan1)
jan1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime1']
jan1
```

Out[105]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	4.147784	6.945637	4.374111	6.710931	4.356727	2
1	7.594026	14.270650	8.374191	14.925587	7.681661	2
2	1.163832	1.379135	1.376192	1.587731	1.788145	1
3	3.019481	3.331026	2.471341	2.499381	2.644888	4
4	9.733277	16.658934	9.143169	16.138129	9.300967	4
...	...	...	...	...	...	
49996	4.127908	5.054376	4.317536	5.345597	5.970606	3
49997	9.080258	13.226641	8.159088	12.356414	10.301319	3
49998	4.160313	4.444944	4.084746	4.188293	5.593965	4
49999	3.410557	4.137114	3.622108	4.352451	5.748773	1
50000	4.004006	5.117827	4.007819	5.094910	6.343775	1

50001 rows × 6 columns

In [114]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jan1.itertuples():
    if(jan1.at[i, 'TotalDistanceP2P'] > jan1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jan1.at[i, 'TotalDistanceP2P'] > jan1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jan1.at[i, 'TotalDistanceP2P'] > jan1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jan1.at[i, 'TotalDistanceP2P'] > jan1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jan1DF1 = pd.DataFrame(myList11)
jan1DF2 = pd.DataFrame(myList12)
jan1DF3 = pd.DataFrame(myList13)
jan1DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in jan1DF1.itertuples():
    jan1DF1["Savings1"] = jan1DF1["TotalDistanceP2P"] - jan1DF1["Sequenc
e1Dist"]

#for row in jan1DF2.itertuples():
    #jan1DF2["Savings2"] = jan1DF2["TotalDistanceP2P"] - jan1DF2["Sequen
ce2Dist"]

#for row in jan1DF3.itertuples():
    #jan1DF3["Savings3"] = jan1DF3["TotalDistanceP2P"] - jan1DF3["Sequen
ce3Dist"]

#for row in jan1DF4.itertuples():
    #jan1DF4["Savings4"] = jan1DF4["TotalDistanceP2P"] - jan1DF4["Sequen
ce4Dist"]

#Display the dataframes
jan1DF1
#jan1DF2
#jan1DF3
#jan1DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan1DF4['Savings4'].sum()
```

Out[114]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	4.147784	6.945637	4.374111	6.710931	4.356727
1	1	7.594026	14.270650	8.374191	14.925587	7.681661
2	2	1.163832	1.379135	1.376192	1.587731	1.788145
3	6	1.420138	1.556477	1.712223	1.653551	1.629456
4	7	1.236090	1.373982	1.565834	1.755657	1.593350
...	...	...	...	...	...	...
41958	49996	4.127908	5.054376	4.317536	5.345597	5.970606
41959	49997	9.080258	13.226641	8.159088	12.356414	10.301319
41960	49998	4.160313	4.444944	4.084746	4.188293	5.593965
41961	49999	3.410557	4.137114	3.622108	4.352451	5.748773
41962	50000	4.004006	5.117827	4.007819	5.094910	6.343775

41963 rows × 8 columns

In [115]:
```
jan1DF1['Savings1'].sum()
```

Out[115]:
```
35860.72350446017
```

In [4]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjan2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [5]:
```
listJan2 = []
listJan2 = test_fetchmany()
jan2 = pd.DataFrame(listJan2)
jan2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
jan2
```

Out[5]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	1.163832	1.379135	1.376192	1.587731	1.788145	2
1	1.420138	1.556477	1.712223	1.653551	1.629456	2
2	1.236090	1.373982	1.565834	1.755657	1.593350	2
3	1.354496	1.580440	1.981019	1.891115	1.384623	3
4	1.613763	1.686340	1.698797	1.802467	1.772745	3
...	...	...	...	...	...	
49996	8.592588	5.915843	11.222305	8.721670	6.093884	0
49997	6.719161	5.784099	11.127548	10.235013	6.056897	0
49998	6.149329	5.738473	10.933777	10.537754	6.205042	0
49999	8.007096	5.713223	10.756526	8.455164	6.357042	0
50000	8.845313	6.043675	11.243596	8.294098	6.200424	1

50001 rows × 6 columns

In [5]:

```
---------------------------------------------------------------------
----
ValueError                                Traceback (most recent call l
ast)
<ipython-input-5-086d4802a337> in <module>
      1 jan2 = pd.DataFrame(listJan2)
----> 2 jan2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dis
t', 'Sequence4Dist', 'TotalDistanceP2P', 'comparedTime2']
      3 jan2

~/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py in _
_setattr__(self, name, value)
   5285         try:
   5286             object.__getattribute__(self, name)
-> 5287             return object.__setattr__(self, name, value)
   5288         except AttributeError:
   5289             pass

pandas/_libs/properties.pyx in pandas._libs.properties.AxisProperty.__s
et__()

~/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py in _
set_axis(self, axis, labels)
    659
    660     def _set_axis(self, axis, labels) -> None:
--> 661         self._data.set_axis(axis, labels)
    662         self._clear_item_cache()
    663

~/opt/anaconda3/lib/python3.8/site-packages/pandas/core/internals/manag
ers.py in set_axis(self, axis, new_labels)
    175
    176         if new_len != old_len:
--> 177             raise ValueError(
    178                 f"Length mismatch: Expected axis has {old_len}
 elements, new "
    179                 f"values have {new_len} elements"

ValueError: Length mismatch: Expected axis has 0 elements, new values h
ave 6 elements
```

In [6]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jan2.itertuples():
    if(jan2.at[i, 'TotalDistanceP2P'] > jan2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jan2.at[i, 'TotalDistanceP2P'] > jan2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jan2.at[i, 'TotalDistanceP2P'] > jan2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jan2.at[i, 'TotalDistanceP2P'] > jan2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jan2DF1 = pd.DataFrame(myList11)
jan2DF2 = pd.DataFrame(myList12)
jan2DF3 = pd.DataFrame(myList13)
jan2DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
#for row in jan1DF1.itertuples():
    #jan1DF1["Savings1"] = jan1DF1["TotalDistanceP2P"] - jan1DF1["Sequen
ce1Dist"]

for row in jan2DF2.itertuples():
    jan2DF2["Savings2"] = jan2DF2["TotalDistanceP2P"] - jan2DF2["Sequenc
e2Dist"]

#for row in jan1DF3.itertuples():
    #jan1DF3["Savings3"] = jan1DF3["TotalDistanceP2P"] - jan1DF3["Sequen
ce3Dist"]

#for row in jan1DF4.itertuples():
    #jan1DF4["Savings4"] = jan1DF4["TotalDistanceP2P"] - jan1DF4["Sequen
ce4Dist"]

#Display the dataframes
#jan1DF1
jan2DF2
#jan1DF3
#jan1DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan1DF4['Savings4'].sum()
```

Out[6]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	14	1.682297	1.577336	1.716807	1.705080	1.645731
1	19	1.557452	1.361307	1.779113	1.419339	1.367396
2	20	1.566773	1.280893	1.720733	1.380213	1.345361
3	24	1.723800	1.435989	1.715842	1.606817	1.505349
4	31	1.419904	0.892713	1.741091	1.213898	0.936825
...	...	...	...	...	...	...
30021	49995	10.528201	6.115196	11.082012	6.588419	6.433530
30022	49996	8.592588	5.915843	11.222305	8.721670	6.093884
30023	49997	6.719161	5.784099	11.127548	10.235013	6.056897
30024	49999	8.007096	5.713223	10.756526	8.455164	6.357042
30025	50000	8.845313	6.043675	11.243596	8.294098	6.200424

30026 rows × 8 columns

In [7]:
```
jan2DF2['Savings2'].sum()
```

Out[7]:
```
17781.34176140048
```

In [8]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjan3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [9]:
```
listJan3 = []
listJan3 = test_fetchmany()
jan3 = pd.DataFrame(listJan3)
jan3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
jan3
```

Out[9]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	1.045008	1.197393	1.437158	1.589538	1.545437	3
1	1.302420	1.171366	1.167656	1.036688	1.788912	2
2	1.859482	2.220794	2.125018	2.347953	1.880978	4
3	2.035062	3.152297	2.352412	3.461861	2.585087	3
4	4.173236	4.364565	3.065607	3.506003	3.084160	2
...	...	...	...	...	...	
49996	4.064814	4.469834	2.525426	3.046874	2.641385	1
49997	5.797629	8.179911	4.329795	6.725982	4.547093	0
49998	0.794072	1.078198	0.851606	1.046075	0.923567	2
49999	1.866280	1.980854	1.268560	1.287550	1.409269	1
50000	3.198960	5.596268	3.077706	5.515593	3.215537	1

50001 rows × 6 columns

In [10]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jan3.itertuples():
    if(jan3.at[i, 'TotalDistanceP2P'] > jan3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jan3.at[i, 'TotalDistanceP2P'] > jan3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jan3.at[i, 'TotalDistanceP2P'] > jan3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jan3.at[i, 'TotalDistanceP2P'] > jan3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#jan3DF1 = pd.DataFrame(myList11)
#jan3DF2 = pd.DataFrame(myList12)
jan3DF3 = pd.DataFrame(myList13)
#jan3DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
#for row in jan1DF1.itertuples():
    #jan1DF1["Savings1"] = jan1DF1["TotalDistanceP2P"] - jan1DF1["Sequen
ce1Dist"]

#for row in jan2DF2.itertuples():
    #jan2DF2["Savings2"] = jan2DF2["TotalDistanceP2P"] - jan2DF2["Sequen
ce2Dist"]

for row in jan3DF3.itertuples():
    jan3DF3["Savings3"] = jan3DF3["TotalDistanceP2P"] - jan3DF3["Sequenc
e3Dist"]

#for row in jan1DF4.itertuples():
    #jan1DF4["Savings4"] = jan1DF4["TotalDistanceP2P"] - jan1DF4["Sequen
ce4Dist"]

#Display the dataframes
#jan1DF1
#jan2DF2
jan3DF3
#jan1DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan1DF4['Savings4'].sum()
```

Out[10]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	4	4.173236	4.364565	3.065607	3.506003	3.084160
1	5	12.249756	16.327885	8.881914	13.057702	9.231173
2	6	16.929316	20.598093	11.143022	14.119167	11.240273
3	8	17.113718	22.629978	12.123588	18.137657	12.291593
4	9	12.339853	12.918500	6.958111	7.581017	7.745591
...	...	...	...	...	...	...
24395	49994	4.584619	4.586220	2.637827	2.826836	2.645369
24396	49995	3.974138	6.953757	3.781594	6.865511	3.869138
24397	49996	4.064814	4.469834	2.525426	3.046874	2.641385
24398	49997	5.797629	8.179911	4.329795	6.725982	4.547093
24399	49999	1.866280	1.980854	1.268560	1.287550	1.409269

24400 rows × 8 columns

In [11]:
```
jan3DF3['Savings3'].sum()
```

Out[11]:
```
13736.735041348884
```

In [2]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjan4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [3]:
```
listJan4 = []
listJan4 = test_fetchmany()
jan4 = pd.DataFrame(listJan4)
jan4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
jan4
```

Out[3]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	1.045008	1.197393	1.437158	1.589538	1.545437	3
1	1.302420	1.171366	1.167656	1.036688	1.788912	1
2	1.140589	1.142942	1.714821	1.722555	1.213323	4
3	1.626853	1.836530	1.956712	1.849960	1.665019	4
4	1.257092	1.353798	1.539318	1.549409	1.599681	2
...	...	...	...	...	...	
49996	6.361108	4.338910	6.837716	4.786916	5.882370	1
49997	9.877740	6.255176	8.887940	5.468124	5.748412	4
49998	7.584469	4.918641	7.520394	4.873720	5.779423	2
49999	5.576073	4.251949	6.346079	5.009146	6.287046	2
50000	5.615725	5.001394	5.265725	4.655637	8.116845	1

50001 rows × 6 columns

In [4]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jan4.itertuples():
    if(jan4.at[i, 'TotalDistanceP2P'] > jan4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jan4.at[i, 'TotalDistanceP2P'] > jan4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jan4.at[i, 'TotalDistanceP2P'] > jan4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jan4.at[i, 'TotalDistanceP2P'] > jan4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#jan3DF1 = pd.DataFrame(myList11)
#jan3DF2 = pd.DataFrame(myList12)
#jan3DF3 = pd.DataFrame(myList13)
jan4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
#for row in jan1DF1.itertuples():
    #jan1DF1["Savings1"] = jan1DF1["TotalDistanceP2P"] - jan1DF1["Sequen
ce1Dist"]

#for row in jan2DF2.itertuples():
    #jan2DF2["Savings2"] = jan2DF2["TotalDistanceP2P"] - jan2DF2["Sequen
ce2Dist"]

#for row in jan3DF3.itertuples():
    #jan3DF3["Savings3"] = jan3DF3["TotalDistanceP2P"] - jan3DF3["Sequen
ce3Dist"]

for row in jan4DF4.itertuples():
    jan4DF4["Savings4"] = jan4DF4["TotalDistanceP2P"] - jan4DF4["Sequenc
e4Dist"]

#Display the dataframes
#jan1DF1
#jan2DF2
#jan3DF3
jan4DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan4DF4['Savings4'].sum()
```

Out[4]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	c
0	8	5.039473	3.444587	4.321510	2.776235	3.098004	
1	9	4.295767	2.708966	4.053238	2.579564	2.630655	
2	10	4.159613	3.553996	4.287635	3.036677	3.241289	
3	11	4.290391	3.702756	4.189132	3.091304	3.488551	
4	12	4.520077	3.040507	4.058896	2.677409	2.956539	
...	...	...	...	...	...	...	
7958	49972	8.706855	5.138097	8.503247	4.992047	5.016026	
7959	49979	9.775873	6.511151	8.669972	5.422462	6.222355	
7960	49985	10.198528	6.547992	9.043989	5.497867	5.885178	
7961	49986	9.298902	5.674385	8.621094	5.005794	5.434466	
7962	49997	9.877740	6.255176	8.887940	5.468124	5.748412	

7963 rows × 8 columns

In [5]:
```
jan4DF4['Savings4'].sum()
```

Out[5]: 1690.9239010225585

In [2]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewfeb1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [3]:
```
listFeb1 = []
listFeb1 = test_fetchmany()
feb1 = pd.DataFrame(listFeb1)
feb1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
feb1
```

Out[3]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	3.442580	3.174541	3.412724	3.172757	5.018159	4
1	10.026140	16.168116	9.959783	16.067817	11.464674	2
2	3.472001	3.062238	3.333905	2.942883	4.984675	4
3	3.608906	3.423828	3.240125	3.058026	5.440045	2
4	3.142988	2.735684	3.331688	2.923432	4.660338	4
...	...	...	...	...	...	
49996	2.796853	3.213284	2.813603	3.182269	3.222398	4
49997	2.383353	3.304770	2.385840	3.306893	3.741648	0
49998	5.610074	9.087352	6.130814	8.997404	5.779256	4
49999	2.512610	3.291267	3.029487	3.743554	3.084497	4
50000	5.004531	7.389300	4.645906	6.821327	5.566112	3

50001 rows × 6 columns

In [4]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in feb1.itertuples():
    if(feb1.at[i, 'TotalDistanceP2P'] > feb1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(feb1.at[i, 'TotalDistanceP2P'] > feb1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (feb1.at[i, 'TotalDistanceP2P'] > feb1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (feb1.at[i, 'TotalDistanceP2P'] > feb1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
feb1DF1 = pd.DataFrame(myList11)
#jan3DF2 = pd.DataFrame(myList12)
#jan3DF3 = pd.DataFrame(myList13)
#jan4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in feb1DF1.itertuples():
    feb1DF1["Savings1"] = feb1DF1["TotalDistanceP2P"] - feb1DF1["Sequenc
e1Dist"]

#for row in jan2DF2.itertuples():
    #jan2DF2["Savings2"] = jan2DF2["TotalDistanceP2P"] - jan2DF2["Sequen
ce2Dist"]

#for row in jan3DF3.itertuples():
    #jan3DF3["Savings3"] = jan3DF3["TotalDistanceP2P"] - jan3DF3["Sequen
ce3Dist"]

#for row in jan4DF4.itertuples():
    #jan4DF4["Savings4"] = jan4DF4["TotalDistanceP2P"] - jan4DF4["Sequen
ce4Dist"]

#Display the dataframes
feb1DF1
#jan2DF2
#jan3DF3
#jan4DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan4DF4['Savings4'].sum()
```

Out[4]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	3.442580	3.174541	3.412724	3.172757	5.018159
1	1	10.026140	16.168116	9.959783	16.067817	11.464674
2	2	3.472001	3.062238	3.333905	2.942883	4.984675
3	3	3.608906	3.423828	3.240125	3.058026	5.440045
4	4	3.142988	2.735684	3.331688	2.923432	4.660338
...	...	...	...	...	...	...
42548	49996	2.796853	3.213284	2.813603	3.182269	3.222398
42549	49997	2.383353	3.304770	2.385840	3.306893	3.741648
42550	49998	5.610074	9.087352	6.130814	8.997404	5.779256
42551	49999	2.512610	3.291267	3.029487	3.743554	3.084497
42552	50000	5.004531	7.389300	4.645906	6.821327	5.566112

42553 rows × 8 columns

In [5]:
```
feb1DF1['Savings1'].sum()
```

Out[5]:
```
40807.343900144624
```

In [7]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewfeb2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [8]:
```
listFeb2 = []
listFeb2 = test_fetchmany()
feb2 = pd.DataFrame(listFeb2)
feb2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
feb2
```

Out[8]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	4.783107	3.990676	4.134569	3.366676	5.112449	3
1	3.442580	3.174541	3.412724	3.172757	5.018159	2
2	4.319584	3.039856	4.371567	3.024434	3.924631	1
3	4.704923	3.331332	4.760546	3.180755	3.827127	3
4	5.828050	3.448900	5.477396	3.225946	3.227846	3
...	...	...	...	...	...	
49996	3.972387	3.169853	5.177195	4.580944	3.408848	2
49997	4.143702	3.715489	5.170885	5.149737	3.960794	4
49998	5.363404	3.648835	5.131363	3.329085	3.933661	4
49999	5.109738	3.260811	5.406306	3.232779	3.270694	2
50000	4.141675	3.666234	4.359671	3.700623	4.722753	4

50001 rows × 6 columns

In [10]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in feb2.itertuples():
    if(feb2.at[i, 'TotalDistanceP2P'] > feb2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(feb2.at[i, 'TotalDistanceP2P'] > feb2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (feb2.at[i, 'TotalDistanceP2P'] > feb2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (feb2.at[i, 'TotalDistanceP2P'] > feb2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#feb1DF1 = pd.DataFrame(myList11)
feb2DF2 = pd.DataFrame(myList12)
#jan3DF3 = pd.DataFrame(myList13)
#jan4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
#for row in feb1DF1.itertuples():
    #feb1DF1["Savings1"] = feb1DF1["TotalDistanceP2P"] - feb1DF1["Sequen
ce1Dist"]

for row in feb2DF2.itertuples():
    feb2DF2["Savings2"] = feb2DF2["TotalDistanceP2P"] - feb2DF2["Sequenc
e2Dist"]

#for row in jan3DF3.itertuples():
    #jan3DF3["Savings3"] = jan3DF3["TotalDistanceP2P"] - jan3DF3["Sequen
ce3Dist"]

#for row in jan4DF4.itertuples():
    #jan4DF4["Savings4"] = jan4DF4["TotalDistanceP2P"] - jan4DF4["Sequen
ce4Dist"]

#Display the dataframes
#feb2DF1
feb2DF2
#jan3DF3
#jan4DF4

#Calculate how much is saved for the month
#jan1DF1['Savings1'].sum()
#jan1DF2['Savings2'].sum()
#jan1DF3['Savings3'].sum()
#jan4DF4['Savings4'].sum()
```

Out[10]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	2	4.319584	3.039856	4.371567	3.024434	3.924631
1	3	4.704923	3.331332	4.760546	3.180755	3.827127
2	5	5.023715	3.693893	4.495862	3.169667	4.454373
3	9	4.268789	3.669195	5.179065	4.117508	3.746472
4	10	4.574772	3.469800	4.781929	3.462447	3.944213
...	...	...	...	...	...	...
31262	49995	5.587844	3.816037	5.356130	4.001884	3.876097
31263	49996	3.972387	3.169853	5.177195	4.580944	3.408848
31264	49997	4.143702	3.715489	5.170885	5.149737	3.960794
31265	49998	5.363404	3.648835	5.131363	3.329085	3.933661
31266	49999	5.109738	3.260811	5.406306	3.232779	3.270694

31267 rows × 8 columns

In [12]:
```
feb2DF2['Savings2'].sum()
```

Out[12]:
```
20468.5774749735
```

In [13]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewfeb3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [14]:
```
listFeb3 = []
listFeb3 = test_fetchmany()
feb3 = pd.DataFrame(listFeb3)
feb3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
feb3
```

Out[14]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	9.079720	9.706751	8.367840	9.009265	14.705535	1
1	13.751115	19.493954	10.280060	15.897345	10.999986	0
2	1.386668	2.237595	1.470492	2.223133	1.577446	2
3	7.744599	11.736373	7.249818	11.284335	9.678988	0
4	8.045559	12.471477	7.612262	11.871043	8.507410	3
...	...	...	...	...	...	
49996	12.239755	12.483086	7.145939	6.607167	6.615465	3
49997	8.400593	16.146586	8.402228	16.148506	9.022676	0
49998	4.692654	8.406793	4.691906	8.412948	4.993206	0
49999	16.774651	16.750700	9.063990	8.888426	8.965029	1
50000	1.365196	1.202671	1.266800	1.197609	1.214190	3

50001 rows × 6 columns

In [15]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in feb3.itertuples():
    if(feb3.at[i, 'TotalDistanceP2P'] > feb3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(feb3.at[i, 'TotalDistanceP2P'] > feb3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (feb3.at[i, 'TotalDistanceP2P'] > feb3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (feb3.at[i, 'TotalDistanceP2P'] > feb3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#feb1DF1 = pd.DataFrame(myList11)
#feb2DF2 = pd.DataFrame(myList12)
feb3DF3 = pd.DataFrame(myList13)
#jan4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in feb3DF3.itertuples():
    feb3DF3["Savings3"] = feb3DF3["TotalDistanceP2P"] - feb3DF3["Sequenc
e3Dist"]

#Display the dataframes
feb3DF3

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[15]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	1	13.751115	19.493954	10.280060	15.897345	10.999986
1	5	16.412247	18.687800	10.404957	12.572605	11.931038
2	12	18.967658	20.185747	11.426980	13.594422	11.455114
3	14	8.771515	9.646263	6.151044	7.751485	6.191567
4	18	1.406301	1.627604	1.181893	1.277923	1.334236
...	...	...	...	...	...	...
23939	49974	1.765377	2.091206	1.327158	1.622450	1.579150
23940	49975	13.719043	16.467926	9.877905	12.336689	12.115038
23941	49978	11.301857	14.645726	9.271531	13.092562	10.899212
23942	49979	14.813463	15.317436	9.477893	9.677256	11.364560
23943	49993	2.504072	2.886209	1.821697	2.214506	2.342831

23944 rows × 8 columns

In [16]:
```
feb3DF3['Savings3'].sum()
```

Out[16]:
```
13472.776225921232
```

In [17]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewfeb4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [18]:
```
listFeb4 = []
listFeb4 = test_fetchmany()
feb4 = pd.DataFrame(listFeb4)
feb4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime4']
feb4
```

Out[18]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	8.834231	6.931230	9.905995	8.062957	10.391860	4
1	14.161020	8.101125	13.558770	7.846935	7.908981	4
2	13.772544	8.479212	13.030503	7.925905	8.652274	3
3	3.230952	2.512283	3.497348	2.496762	2.663131	4
4	3.105181	2.621279	3.131575	2.622897	2.795441	4
...	...	...	...	...	...	
49996	6.307947	4.503151	6.892820	5.104381	6.437644	1
49997	7.316337	5.101772	7.415492	5.216233	6.513593	2
49998	7.322126	4.618123	7.666124	4.980052	5.779312	1
49999	8.148565	6.609873	7.339691	5.659414	8.097494	2
50000	7.641167	6.077917	6.905551	5.344598	7.999678	2

50001 rows × 6 columns

In [21]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in feb4.itertuples():
    if(feb4.at[i, 'TotalDistanceP2P'] > feb4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(feb4.at[i, 'TotalDistanceP2P'] > feb4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (feb4.at[i, 'TotalDistanceP2P'] > feb4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (feb4.at[i, 'TotalDistanceP2P'] > feb4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#feb1DF1 = pd.DataFrame(myList11)
#feb2DF2 = pd.DataFrame(myList12)
#feb3DF3 = pd.DataFrame(myList13)
feb4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in feb4DF4.itertuples():
    feb4DF4["Savings4"] = feb4DF4["TotalDistanceP2P"] - feb4DF4["Sequenc
e4Dist"]

#Display the dataframes
feb4DF4

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[21]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	c
0	1	14.161020	8.101125	13.558770	7.846935	7.908981	
1	5	6.683275	4.841367	6.939081	4.085261	4.122146	
2	19	14.630179	8.537520	13.489473	7.361337	7.403290	
3	44	9.238956	6.131822	8.318522	5.361782	6.044704	
4	47	10.634424	6.355283	9.973007	5.817717	6.069061	
...	...	...	...	...	...	...	
8217	49966	10.623190	6.551777	9.562103	5.495499	5.816986	
8218	49973	8.739155	5.017049	8.881933	4.886543	4.962428	
8219	49979	10.696237	6.532527	9.676491	5.478750	5.683349	
8220	49988	9.681281	5.551271	9.133057	5.010099	5.245526	
8221	49993	9.252420	5.612381	8.838478	5.464724	5.601215	

8222 rows × 8 columns

In [22]:
```
feb4DF4['Savings4'].sum()
```

Out[22]:
```
1680.8044166577415
```

In [2]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmar1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [3]:
```
listMar1 = []
listMar1 = test_fetchmany()
mar1 = pd.DataFrame(listMar1)
mar1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime1']
mar1
```

Out[3]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	8.185607	12.140433	7.256353	11.316639	9.101365	3
1	5.584278	8.261276	5.214892	7.902922	7.263670	1
2	6.343978	9.244332	6.540266	9.850030	6.921352	4
3	6.756769	10.215809	6.741585	10.474164	7.691510	4
4	5.830131	7.288191	5.220295	6.988619	6.285181	4
...	...	...	...	...	...	
49996	3.834116	5.620736	4.638015	6.211011	4.401409	3
49997	1.854155	1.897673	2.304827	2.355560	3.011532	2
49998	5.296781	8.774119	6.105785	9.425227	6.087021	3
49999	3.135847	3.309503	2.751935	2.866346	3.976255	4
50000	7.885286	13.297358	7.900634	13.432116	8.815412	2

50001 rows × 6 columns

In [4]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in mar1.itertuples():
    if(mar1.at[i, 'TotalDistanceP2P'] > mar1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(mar1.at[i, 'TotalDistanceP2P'] > mar1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (mar1.at[i, 'TotalDistanceP2P'] > mar1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (mar1.at[i, 'TotalDistanceP2P'] > mar1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
mar1DF1 = pd.DataFrame(myList11)
#feb2DF2 = pd.DataFrame(myList12)
#feb3DF3 = pd.DataFrame(myList13)
#feb4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in mar1DF1.itertuples():
    mar1DF1["Savings1"] = mar1DF1["TotalDistanceP2P"] - mar1DF1["Sequenc
e1Dist"]

#Display the dataframes
mar1DF1

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[4]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	8.185607	12.140433	7.256353	11.316639	9.101365
1	1	5.584278	8.261276	5.214892	7.902922	7.263670
2	2	6.343978	9.244332	6.540266	9.850030	6.921352
3	3	6.756769	10.215809	6.741585	10.474164	7.691510
4	4	5.830131	7.288191	5.220295	6.988619	6.285181
...	...	...	...	...	...	...
43495	49996	3.834116	5.620736	4.638015	6.211011	4.401409
43496	49997	1.854155	1.897673	2.304827	2.355560	3.011532
43497	49998	5.296781	8.774119	6.105785	9.425227	6.087021
43498	49999	3.135847	3.309503	2.751935	2.866346	3.976255
43499	50000	7.885286	13.297358	7.900634	13.432116	8.815412

43500 rows × 8 columns

In [5]:
```
mar1DF1['Savings1'].sum()
```

Out[5]:
```
36618.84727859423
```

In [6]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmar2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [7]:
```
listMar2 = []
listMar2 = test_fetchmany()
mar2 = pd.DataFrame(listMar2)
mar2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
mar2
```

Out[7]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	3.862654	3.037713	3.524169	2.636645	3.730830	3
1	3.098073	2.909562	2.774820	2.605729	4.352028	2
2	3.975266	3.498246	3.920552	3.243960	3.794980	4
3	4.503530	2.679881	4.439136	2.422402	2.458031	2
4	3.033625	3.482506	2.828497	3.286019	4.871295	4
...	...	...	...	...	...	
49996	7.626762	4.722981	8.026562	4.748279	4.798385	2
49997	6.840290	4.928703	7.317955	5.730300	5.712714	3
49998	7.216254	4.111566	7.933157	4.862161	4.280375	0
49999	7.505185	4.868285	7.732737	5.514681	5.237514	3
50000	7.290547	4.345252	7.745143	4.617794	4.702075	1

50001 rows × 6 columns

In [8]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in mar2.itertuples():
    if(mar2.at[i, 'TotalDistanceP2P'] > mar2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(mar2.at[i, 'TotalDistanceP2P'] > mar2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (mar2.at[i, 'TotalDistanceP2P'] > mar2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (mar2.at[i, 'TotalDistanceP2P'] > mar2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#mar1DF1 = pd.DataFrame(myList11)
mar2DF2 = pd.DataFrame(myList12)
#feb3DF3 = pd.DataFrame(myList13)
#feb4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in mar2DF2.itertuples():
    mar2DF2["Savings2"] = mar2DF2["TotalDistanceP2P"] - mar2DF2["Sequenc
e2Dist"]

#Display the dataframes
mar2DF2

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[8]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	3.862654	3.037713	3.524169	2.636645	3.730830
1	2	3.975266	3.498246	3.920552	3.243960	3.794980
2	5	4.644059	4.025290	3.706889	3.107034	4.535686
3	6	3.925796	2.815328	4.046904	2.906666	2.985710
4	9	3.675085	3.095192	4.093245	3.296301	3.219232
...	...	...	...	...	...	...
31252	49996	7.626762	4.722981	8.026562	4.748279	4.798385
31253	49997	6.840290	4.928703	7.317955	5.730300	5.712714
31254	49998	7.216254	4.111566	7.933157	4.862161	4.280375
31255	49999	7.505185	4.868285	7.732737	5.514681	5.237514
31256	50000	7.290547	4.345252	7.745143	4.617794	4.702075

31257 rows × 8 columns

In [9]:
```
mar2DF2['Savings2'].sum()
```

Out[9]:
```
18411.69652092278
```

In [10]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmar3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [11]:
```
listMar3 = []
listMar3 = test_fetchmany()
mar3 = pd.DataFrame(listMar3)
mar3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
mar3
```

Out[11]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	6.542151	8.136106	5.415132	7.236460	6.938259	1
1	8.265536	9.943986	6.593067	8.435726	7.568205	4
2	13.802551	23.065664	13.727862	22.886969	15.394701	3
3	7.949599	10.786058	6.940332	10.137415	8.237625	2
4	25.548230	24.589648	14.419345	13.439931	14.562201	4
...	...	...	...	...	...	
49996	13.527849	15.423057	8.570066	10.498518	10.181416	0
49997	9.249301	10.110562	6.500464	7.220787	6.938523	4
49998	4.269864	6.786904	4.683151	7.224126	5.432177	3
49999	7.656026	7.411488	5.069976	5.107786	5.669936	3
50000	3.757568	3.727048	3.146141	3.094273	3.909332	3

50001 rows × 6 columns

In [12]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in mar3.itertuples():
    if(mar3.at[i, 'TotalDistanceP2P'] > mar3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(mar3.at[i, 'TotalDistanceP2P'] > mar3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (mar3.at[i, 'TotalDistanceP2P'] > mar3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (mar3.at[i, 'TotalDistanceP2P'] > mar3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
#mar1DF1 = pd.DataFrame(myList11)
#mar2DF2 = pd.DataFrame(myList12)
mar3DF3 = pd.DataFrame(myList13)
#feb4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in mar3DF3.itertuples():
    mar3DF3["Savings3"] = mar3DF3["TotalDistanceP2P"] - mar3DF3["Sequenc
e3Dist"]

#Display the dataframes
mar3DF3

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[12]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	1	8.265536	9.943986	6.593067	8.435726	7.568205
1	4	25.548230	24.589648	14.419345	13.439931	14.562201
2	5	8.172451	8.128185	5.851264	6.097292	6.668820
3	8	22.952447	23.195809	12.625232	12.903220	14.161927
4	9	6.415030	7.570325	4.893130	6.080929	6.268545
...	...	...	...	...	...	...
23305	49992	7.381213	6.790619	4.964297	4.318435	5.154747
23306	49995	12.782995	12.341930	7.670298	7.465995	8.000057
23307	49996	13.527849	15.423057	8.570066	10.498518	10.181416
23308	49997	9.249301	10.110562	6.500464	7.220787	6.938523
23309	49999	7.656026	7.411488	5.069976	5.107786	5.669936

23310 rows × 8 columns

In [13]:
```
mar3DF3['Savings3'].sum()
```

Out[13]:
```
12333.317719406341
```

In [4]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmar4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [7]:
```
listMar4 = []
listMar4 = test_fetchmany()
mar4 = pd.DataFrame(listMar4)
mar4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime4']
mar4
```

Out[7]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	3.683364	3.153803	4.086699	3.123693	3.284390	3
1	4.579104	3.491954	4.172264	2.923984	3.536976	2
2	4.907805	3.287586	5.418038	3.785658	3.926446	3
3	3.943696	3.379828	4.597993	3.967291	4.838733	3
4	4.268979	3.074603	3.802401	2.564624	2.863551	2
...	...	...	...	...	...	
49996	4.620036	3.177599	3.968424	2.404750	2.439515	4
49997	2.724177	2.033294	2.779682	2.069793	2.483952	3
49998	6.695044	9.359441	6.230257	9.035869	6.359524	-9
49999	2.837798	1.916616	2.846819	1.891863	2.300137	2
50000	2.785984	1.897725	2.993095	2.085558	2.134971	4

50001 rows × 6 columns

In [8]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in mar4.itertuples():
    if(mar4.at[i, 'TotalDistanceP2P'] > mar4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(mar4.at[i, 'TotalDistanceP2P'] > mar4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (mar4.at[i, 'TotalDistanceP2P'] > mar4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (mar4.at[i, 'TotalDistanceP2P'] > mar4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
mar4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in mar4DF4.itertuples():
    mar4DF4["Savings4"] = mar4DF4["TotalDistanceP2P"] - mar4DF4["Sequenc
e4Dist"]

#Display the dataframes
mar4DF4

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[8]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	c
0	4	4.268979	3.074603	3.802401	2.564624	2.863551	
1	7	6.597426	4.244671	6.502841	3.627682	3.654103	
2	19	14.761484	9.046442	15.200408	8.698219	9.008465	
3	25	15.042381	8.783319	14.226177	8.262121	8.542587	
4	39	3.034536	2.479244	2.432951	1.742541	1.921578	
...	...	...	...	...	...	...	
8032	49969	3.234569	2.387450	2.569847	1.720813	1.919960	
8033	49974	3.123704	2.083920	2.615298	1.567947	1.570979	
8034	49985	3.113619	2.425063	2.610553	1.799559	1.930690	
8035	49987	3.827784	2.702225	3.434594	2.210516	2.497970	
8036	49996	4.620036	3.177599	3.968424	2.404750	2.439515	

8037 rows × 8 columns

In [9]:
```
mar4DF4['Savings4'].sum()
```

Out[9]:
```
1557.4994079381813
```

In [13]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewapr1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [15]:
```
listApr1 = []
listApr1 = test_fetchmany()
apr1 = pd.DataFrame(listApr1)
apr1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime1']
apr1
```

Out[15]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	5.712930	9.130988	5.622793	9.043261	7.339277	
1	2.717241	3.505244	3.189191	3.971705	4.147135	
2	3.393951	3.920879	3.858459	4.399812	3.893501	
3	7.408868	10.420773	6.385351	9.300500	7.866503	
4	3.823194	4.658068	3.669103	4.671964	4.820047	
...	...	...	...	...	...	
49996	2.665058	2.663921	1.988030	1.972146	3.016349	
49997	3.046418	3.093724	2.207641	2.279090	3.226541	
49998	26.239048	49.504417	25.610620	48.850125	26.234254	
49999	2.743906	2.639541	2.080440	1.949682	2.899559	
50000	5384.108132	10765.111410	5383.344604	10764.488376	5384.107264	

50001 rows × 6 columns

In [16]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in apr1.itertuples():
    if(apr1.at[i, 'TotalDistanceP2P'] > apr1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(apr1.at[i, 'TotalDistanceP2P'] > apr1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (apr1.at[i, 'TotalDistanceP2P'] > apr1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (apr1.at[i, 'TotalDistanceP2P'] > apr1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
apr1DF1 = pd.DataFrame(myList11)

#Calculate the savings in a new column
for row in apr1DF1.itertuples():
    apr1DF1["Savings1"] = apr1DF1["TotalDistanceP2P"] - apr1DF1["Sequenc
e1Dist"]

#Display the dataframes
apr1DF1

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[16]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	5.712930	9.130988	5.622793	9.043261	7.339277
1	1	2.717241	3.505244	3.189191	3.971705	4.147135
2	2	3.393951	3.920879	3.858459	4.399812	3.893501
3	3	7.408868	10.420773	6.385351	9.300500	7.866503
4	4	3.823194	4.658068	3.669103	4.671964	4.820047
...	...	...	...	...	...	...
41476	49994	2.929721	3.008603	2.257318	2.384987	3.091742
41477	49995	1.864255	2.380454	2.276438	2.689510	2.444474
41478	49996	2.665058	2.663921	1.988030	1.972146	3.016349
41479	49997	3.046418	3.093724	2.207641	2.279090	3.226541
41480	49999	2.743906	2.639541	2.080440	1.949682	2.899559

41481 rows × 8 columns

In [17]:
```
apr1DF1['Savings1'].sum()
```

Out[17]:
```
77658.55155963424
```

In [31]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewapr2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [32]:
```
listApr2 = []
listApr2 = test_fetchmany()
apr2 = pd.DataFrame(listApr2)
apr2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
apr2
```

Out[32]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	2.717241	3.505244	3.189191	3.971705	4.147135	3
1	3.232045	1.937472	3.338909	2.040631	2.429645	0
2	3.152577	2.556737	3.023461	2.477409	3.364357	1
3	4.922660	3.320705	4.262568	2.733759	2.889219	4
4	3.497571	3.163949	3.799178	3.283440	3.195852	3
...	...	...	...	...	...	
49996	1.060536	0.838254	1.101212	0.876544	1.343847	0
49997	1.658120	1.260980	1.564674	1.202050	1.303111	2
49998	1.828927	1.543956	1.817529	1.251534	1.333233	4
49999	1.077248	0.931374	1.064305	0.920601	1.473874	0
50000	1.279572	1.075061	1.483990	1.300636	1.197876	1

50001 rows × 6 columns

In [33]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in apr2.itertuples():
    if(apr2.at[i, 'TotalDistanceP2P'] > apr2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(apr2.at[i, 'TotalDistanceP2P'] > apr2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (apr2.at[i, 'TotalDistanceP2P'] > apr2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (apr2.at[i, 'TotalDistanceP2P'] > apr2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
apr2DF2 = pd.DataFrame(myList12)

#Calculate the savings in a new column
for row in apr2DF2.itertuples():
    apr2DF2["Savings2"] = apr2DF2["TotalDistanceP2P"] - apr2DF2["Sequenc
e2Dist"]

#Display the dataframes
apr2DF2

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[33]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	1	3.232045	1.937472	3.338909	2.040631	2.429645
1	4	3.497571	3.163949	3.799178	3.283440	3.195852
2	5	3.752885	3.151575	3.765621	3.181936	3.217036
3	9	3.714462	3.407271	3.632947	3.289120	3.605406
4	10	3.594705	2.255913	3.553166	2.141754	2.533829
...	...	...	...	...	...	...
30943	49988	1.506985	1.008688	1.493925	1.066045	1.121569
30944	49989	1.608291	1.437750	1.493895	1.445971	1.550661
30945	49995	1.507714	1.186724	1.331960	1.005164	1.461570
30946	49997	1.658120	1.260980	1.564674	1.202050	1.303111
30947	50000	1.279572	1.075061	1.483990	1.300636	1.197876

30948 rows × 8 columns

In [34]:
```
apr2DF2['Savings2'].sum()
```

Out[34]:
```
16438.55742791945
```

In [35]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewapr3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [36]:
```
listApr3 = []
listApr3 = test_fetchmany()
apr3 = pd.DataFrame(listApr3)
apr3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
apr3
```

Out[36]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	8.323161	10.415000	5.981028	8.070116	6.413500	1
1	5.706173	7.705327	5.672582	7.638042	5.817077	3
2	5.488832	9.904933	5.809591	10.121116	5.761506	2
3	6.666126	7.020965	4.247043	4.511842	4.980246	1
4	2.093681	2.756618	2.298551	3.041457	2.664391	3
...	...	...	...	...	...	
49996	7.112945	11.809684	6.139537	10.833114	6.586648	0
49997	5.571539	9.819353	5.141468	9.389177	5.594387	0
49998	5.492086	9.252011	4.917606	8.743426	5.250907	0
49999	1.453073	1.922030	1.392959	1.997764	1.445573	2
50000	1.830150	2.046749	1.280146	1.516642	1.683105	0

50001 rows × 6 columns

In [38]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in apr3.itertuples():
    if(apr3.at[i, 'TotalDistanceP2P'] > apr3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(apr3.at[i, 'TotalDistanceP2P'] > apr3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (apr3.at[i, 'TotalDistanceP2P'] > apr3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (apr3.at[i, 'TotalDistanceP2P'] > apr3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
apr3DF3 = pd.DataFrame(myList13)

#Calculate the savings in a new column
for row in apr3DF3.itertuples():
    apr3DF3["Savings3"] = apr3DF3["TotalDistanceP2P"] - apr3DF3["Sequenc
e3Dist"]

#Display the dataframes
apr3DF3

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[38]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	8.323161	10.415000	5.981028	8.070116	6.413500
1	3	6.666126	7.020965	4.247043	4.511842	4.980246
2	7	3.845648	4.928600	2.936754	4.121819	3.363897
3	8	7.131650	11.518746	6.253810	10.482598	6.636988
4	29	3.342326	3.189354	2.241432	2.195672	2.550599
...	...	...	...	...	...	...
22959	49995	9.637637	11.146085	5.882579	7.414590	6.180008
22960	49996	7.112945	11.809684	6.139537	10.833114	6.586648
22961	49998	5.492086	9.252011	4.917606	8.743426	5.250907
22962	49999	1.453073	1.922030	1.392959	1.997764	1.445573
22963	50000	1.830150	2.046749	1.280146	1.516642	1.683105

22964 rows × 8 columns

In [39]:
```
apr3DF3['Savings3'].sum()
```

Out[39]:
```
11826.678413595593
```

In [55]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewapr4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [56]:
```
listApr4 = []
listApr4 = test_fetchmany()
apr4 = pd.DataFrame(listApr4)
apr4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime4']
apr4
```

Out[56]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	6.198769	4.961297	6.460379	5.210906	8.033577	1
1	7.917723	4.767570	8.504969	5.354171	5.795261	1
2	9.043261	5.622793	9.130988	5.712930	7.339277	0
3	10.260186	5.539494	10.449126	5.729498	5.937840	0
4	9.506580	5.529416	9.504336	5.534771	6.872552	0
...	...	...	...	...	...	
49996	3.148952	2.397895	3.470721	2.733846	3.620313	1
49997	3.027506	2.350091	3.202921	2.526002	3.840308	0
49998	7.061026	11.671014	7.643581	12.263976	8.720571	4
49999	3.950644	2.718949	4.350678	2.951697	3.061409	3
50000	3.702830	2.412931	3.911233	2.645126	3.194837	1

50001 rows × 6 columns

In [57]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in apr4.itertuples():
    if(apr4.at[i, 'TotalDistanceP2P'] > apr4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(apr4.at[i, 'TotalDistanceP2P'] > apr4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (apr4.at[i, 'TotalDistanceP2P'] > apr4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (apr4.at[i, 'TotalDistanceP2P'] > apr4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
apr4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in apr4DF4.itertuples():
    apr4DF4["Savings4"] = apr4DF4["TotalDistanceP2P"] - apr4DF4["Sequenc
e4Dist"]

#Display the dataframes
apr4DF4

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[57]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	c
0	5	2.244139	2.052970	2.087803	1.610552	1.631332	
1	6	3.021297	2.273178	2.560359	1.732213	1.919143	
2	10	5.056174	2.978654	4.696144	2.707118	2.745699	
3	11	5.115370	3.733278	4.516056	3.209403	3.680411	
4	14	5.890592	4.226784	4.715879	2.990370	3.007682	
...	...	...	...	...	...	...	
8052	49980	6.384501	4.759679	5.307621	3.683151	4.663338	
8053	49981	6.075358	3.608438	5.609969	3.116050	3.209749	
8054	49985	6.340416	3.944104	5.814362	3.281763	3.341023	
8055	49986	5.563969	3.405628	5.217062	3.136569	3.399846	
8056	49989	5.495886	3.466840	5.481527	3.120398	3.196593	

8057 rows × 8 columns

In [58]:
```
apr4DF4['Savings4'].sum()
```

Out[58]:
```
1590.37480608537
```

In [59]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmay1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [60]:
```
listMay1 = []
listMay1 = test_fetchmany()
may1 = pd.DataFrame(listMay1)
may1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime1']
may1
```

Out[60]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	1.660281	2.105791	1.643405	2.087366	2.572969	
1	2.989946	4.352645	2.879176	4.122436	3.584051	
2	4.583219	6.855216	4.427034	6.120677	4.538765	
3	3.388007	5.561758	3.889260	6.045851	3.783080	
4	4.461432	7.554335	5.101005	8.214544	4.563912	
...	...	...	...	...	...	
49996	5.234366	7.457673	5.082512	7.290843	7.527018	
49997	4.914937	6.392583	5.190828	6.587641	6.353613	
49998	7.703270	11.611650	7.285841	11.174266	9.477667	
49999	5.251114	7.101689	5.370420	7.214683	6.883127	
50000	5.711513	7.965053	5.970041	8.392189	7.146870	

50001 rows × 6 columns

In [61]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in may1.itertuples():
    if(may1.at[i, 'TotalDistanceP2P'] > may1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(may1.at[i, 'TotalDistanceP2P'] > may1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (may1.at[i, 'TotalDistanceP2P'] > may1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (may1.at[i, 'TotalDistanceP2P'] > may1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
may1DF1 = pd.DataFrame(myList11)

#Calculate the savings in a new column
for row in may1DF1.itertuples():
    may1DF1["Savings1"] = may1DF1["TotalDistanceP2P"] - may1DF1["Sequenc
e1Dist"]

#Display the dataframes
may1DF1

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[61]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	0	1.660281	2.105791	1.643405	2.087366	2.572969
1	1	2.989946	4.352645	2.879176	4.122436	3.584051
2	3	3.388007	5.561758	3.889260	6.045851	3.783080
3	4	4.461432	7.554335	5.101005	8.214544	4.563912
4	5	1.153679	1.227597	1.744165	1.823612	1.594015
...	...	...	...	...	...	...
42435	49996	5.234366	7.457673	5.082512	7.290843	7.527018
42436	49997	4.914937	6.392583	5.190828	6.587641	6.353613
42437	49998	7.703270	11.611650	7.285841	11.174266	9.477667
42438	49999	5.251114	7.101689	5.370420	7.214683	6.883127
42439	50000	5.711513	7.965053	5.970041	8.392189	7.146870

42440 rows × 8 columns

In [62]:
```
may1DF1['Savings1'].sum()
```

Out[62]:
```
31672.331438127316
```

In [63]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmay2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [64]:
```
listMay2 = []
listMay2 = test_fetchmany()
may2 = pd.DataFrame(listMay2)
may2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
may2
```

Out[64]:

	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P	compar
0	1.660281	2.105791	1.643405	2.087366	2.572969	4
1	1.153679	1.227597	1.744165	1.823612	1.594015	0
2	1.417272	1.491627	1.387691	1.478604	2.214518	1
3	2.002725	1.249421	1.967207	1.181681	1.392797	1
4	1.879368	1.352668	1.842669	1.260224	1.620581	1
...	...	...	...	...	...	
49996	2.259153	1.675294	2.226314	1.580004	1.676303	2
49997	1.566137	1.474420	1.520869	1.463582	2.180874	1
49998	1.710434	2.025491	1.657875	1.953259	2.594939	4
49999	1.730021	1.705895	1.672764	1.589801	2.260454	3
50000	1.578345	1.121888	1.726842	1.269452	1.622369	0

50001 rows × 6 columns

In [65]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in may2.itertuples():
    if(may2.at[i, 'TotalDistanceP2P'] > may2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(may2.at[i, 'TotalDistanceP2P'] > may2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (may2.at[i, 'TotalDistanceP2P'] > may2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (may2.at[i, 'TotalDistanceP2P'] > may2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
may2DF2 = pd.DataFrame(myList12)

#Calculate the savings in a new column
for row in may2DF2.itertuples():
    may2DF2["Savings2"] = may2DF2["TotalDistanceP2P"] - may2DF2["Sequenc
e2Dist"]

#Display the dataframes
may2DF2

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

Out[65]:

	Index	Sequence1Dist	Sequence2Dist	Sequence3Dist	Sequence4Dist	TotalDistanceP2P
0	3	2.002725	1.249421	1.967207	1.181681	1.392797
1	4	1.879368	1.352668	1.842669	1.260224	1.620581
2	12	1.820005	1.154664	1.817598	1.152022	1.447649
3	13	2.442145	2.086543	1.945314	1.585312	2.251812
4	18	1.718356	1.387862	1.842308	1.398385	1.656136
...	...	...	...	...	...	...
29054	49984	1.769971	1.244709	1.717504	1.197304	1.754527
29055	49989	1.710747	1.311075	2.065478	1.766744	1.472920
29056	49990	2.070963	1.117680	2.219707	1.263304	1.125296
29057	49994	1.857449	1.418059	1.798175	1.346166	1.847207
29058	49996	2.259153	1.675294	2.226314	1.580004	1.676303

29059 rows × 8 columns

In [66]:
```
may2DF2['Savings2'].sum()
```

Out[66]:
```
16632.30692090492
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmay3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listMay3 = []
listMay3 = test_fetchmany()
may3 = pd.DataFrame(listMay3)
may3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
may3
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in may3.itertuples():
    if(may3.at[i, 'TotalDistanceP2P'] > may3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(may3.at[i, 'TotalDistanceP2P'] > may3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (may3.at[i, 'TotalDistanceP2P'] > may3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (may3.at[i, 'TotalDistanceP2P'] > may3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
may3DF3 = pd.DataFrame(myList13)

#Calculate the savings in a new column
for row in may3DF3.itertuples():
    may3DF3["Savings3"] = may3DF3["TotalDistanceP2P"] - may3DF3["Sequenc
e3Dist"]

#Display the dataframes
may3DF3

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
may3DF3['Savings3'].sum()
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewmay4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listMay4 = []
listMay4 = test_fetchmany()
may4 = pd.DataFrame(listMay4)
may4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime4']
may4
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in may4.itertuples():
    if(may4.at[i, 'TotalDistanceP2P'] > may4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(may4.at[i, 'TotalDistanceP2P'] > may4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (may4.at[i, 'TotalDistanceP2P'] > may4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (may4.at[i, 'TotalDistanceP2P'] > may4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
may4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in may4DF4.itertuples():
    may4DF4["Savings4"] = may4DF4["TotalDistanceP2P"] - may4DF4["Sequenc
e4Dist"]

#Display the dataframes
may4DF4

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
may4DF4['Savings4'].sum()
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjun1")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listJun1 = []
listJun1 = test_fetchmany()
jun1 = pd.DataFrame(listJun1)
jun1.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime1']
jun1
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jun1.itertuples():
    if(jun1.at[i, 'TotalDistanceP2P'] > jun1.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jun1.at[i, 'TotalDistanceP2P'] > jun1.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jun1.at[i, 'TotalDistanceP2P'] > jun1.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jun1.at[i, 'TotalDistanceP2P'] > jun1.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jun1DF1 = pd.DataFrame(myList11)

#Calculate the savings in a new column
for row in jun1DF1.itertuples():
    jun1DF1["Savings1"] = jun1DF1["TotalDistanceP2P"] - jun1DF1["Sequenc
e1Dist"]

#Display the dataframes
jun1DF1

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
jun1DF1['Savings1'].sum()
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjun2")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listJun2 = []
listJun2 = test_fetchmany()
jun2 = pd.DataFrame(listJun2)
jun2.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime2']
jun2
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jun2.itertuples():
    if(jun2.at[i, 'TotalDistanceP2P'] > jun2.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jun2.at[i, 'TotalDistanceP2P'] > jun2.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jun2.at[i, 'TotalDistanceP2P'] > jun2.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jun2.at[i, 'TotalDistanceP2P'] > jun2.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jun2DF2 = pd.DataFrame(myList12)

#Calculate the savings in a new column
for row in jun2DF2.itertuples():
    jun2DF2["Savings2"] = jun2DF2["TotalDistanceP2P"] - jun2DF2["Sequenc
e2Dist"]

#Display the dataframes
jun2DF2

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
jun2DF2['Savings2'].sum()
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjun3")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listJun3 = []
listJun3 = test_fetchmany()
jun3 = pd.DataFrame(listJun3)
jun3.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime3']
jun3
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jun3.itertuples():
    if(jun3.at[i, 'TotalDistanceP2P'] > jun3.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jun3.at[i, 'TotalDistanceP2P'] > jun3.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jun3.at[i, 'TotalDistanceP2P'] > jun3.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jun3.at[i, 'TotalDistanceP2P'] > jun3.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jun3DF3 = pd.DataFrame(myList13)

#Calculate the savings in a new column
for row in jun3DF3.itertuples():
    jun3DF3["Savings3"] = jun3DF3["TotalDistanceP2P"] - jun3DF3["Sequenc
e3Dist"]

#Display the dataframes
jun3DF3

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
jan3DF3['Savings3'].sum()
```

In [ ]:
```
from mysql.connector import MySQLConnection, Error

def iter_row(cursor, size=50000):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

#queryList = []

def test_fetchmany():
    #queryList = []

    conn = MySQLConnection(host='Dannys-MacBook-Pro.local', database='Fi
nalProject', user='root', password='Aguilard123')
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM finalgreentaxiviewjun4")
    queryList = []
    counter = 0
    for row in iter_row(cursor, 50000):
        queryList.append(row)
        if counter >= 50000:
               break
        counter = counter + 1
        #print(row)
        #print(queryList)
        #df1 = pd.DataFrame(queryList)
        #df1
        #break;

    return queryList
```

In [ ]:
```
listJun4 = []
listJun4 = test_fetchmany()
jun4 = pd.DataFrame(listJun3)
jun4.columns = ['Sequence1Dist', 'Sequence2Dist', 'Sequence3Dist', 'Sequ
ence4Dist', 'TotalDistanceP2P', 'comparedTime4']
jun4
```

In [ ]:
```
#Creates Empty lists to store sequences
myList11 = []
myList12 = []
myList13 = []
myList14 = []

#Loops through to check whoch sequence gives the best saving
i = 0
for row in jun4.itertuples():
    if(jun4.at[i, 'TotalDistanceP2P'] > jun4.at[i, 'Sequence1Dist']):
        myList11.append(row)
    elif(jun4.at[i, 'TotalDistanceP2P'] > jun4.at[i, 'Sequence2Dist']):
        myList12.append(row)
    elif (jun4.at[i, 'TotalDistanceP2P'] > jun4.at[i, 'Sequence3Dist']):
        myList13.append(row)
    elif (jun4.at[i, 'TotalDistanceP2P'] > jun4.at[i, 'Sequence4Dist']):
        myList14.append(row)

    i = i + 1


#Convert the lists into DataFrame
jun4DF4 = pd.DataFrame(myList14)

#Calculate the savings in a new column
for row in jun4DF4.itertuples():
    jun4DF4["Savings4"] = jun4DF4["TotalDistanceP2P"] - jun4DF4["Sequenc
e4Dist"]

#Display the dataframes
jun4DF4

#Calculate how much is saved for the month
#jan4DF4['Savings4'].sum()
```

In [ ]:
```
jun4DF4['Savings4'].sum()
```

In [ ]:
```

```