Final Project Part 2: Stat 102

In [232]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import seaborn as sns
from matplotlib.widgets import Button, CheckButtons
from matplotlib import gridspec
import functools
from Bandit_env import BanditEnv, Interactive_UCB_Algorithm,Interactive_
TS_Algorithm
```

In [233]:
```
chi = pd.read_csv("chicago.csv")
ny = pd.read_csv("ny.csv")
dc = pd.read_csv("dc.csv")
daily = pd.read_csv("day.csv")
```

Question 1: Bandits

Question 1.1 Formalizing the problem as a Multi-Armed Bandits Problem

In this formalization, the arms would be intersections, the rewards would be the # of individuals who take fliers at that intersection, and the rewards would be bounded, as the max reward we can achieve is the # of fliers we can print out (in the simulation, however, I decide to choose sub-gaussian rewards). The time horizon is the number of intersections we need to visit until we have the optimal intersection. My modeling assumptions are the following:

1. Each day of the week will have the same distribution (i.e. Monday will not have better results than Tuesday)
2. Things like weather do not have an impact on the number of fliers we'll hand out
3. The "popularity" (# of fliers passed at) of one intersection does not have an effect on another intersection

Out of all of these assumptions, I only assume the 3rd the hold, while the first two will likely not hold. Weather definitely has an effect on how many people frequent an intersection, which inturn leads to the # of people who would take a flie. For the first assumption, it is likely to not hold as weekdays will have a larger # of frequenters for an intersection than weekends would. We can test this by joining the dataset with weather conditions, and we can test the day of the week assumption by using the data we currently have.

For regret, we think about it as the average reward achieved from the optimal path minus the average reward from path we took (path being the intersections we visit before finding the optimal intersection).

Question 1.2 Simulate UCB strategy using past data

In [234]:
```
def add_from_and_to(x):
    if np.isnan(x['from_count']):
        return x['to_count']
    elif np.isnan(x['to_count']):
        return x['from_count']
    else:
        return x['from_count'] + x['to_count']
```

In [235]:
```
chi['starttime'] = pd.to_datetime(chi['starttime'])
chi['Date'] = chi['starttime'].dt.date
chi_rides_from = chi.groupby(['Date', 'from_station_name']).count()[['tr
ip_id']].reset_index()
chi_rides_from.columns = ['Date', 'station_name', 'from_count']
chi_rides_to = chi.groupby(['Date', 'to_station_name']).count()[['trip_i
d']].reset_index()
chi_rides_to.columns = ['Date', 'station_name', 'to_count']
chi_rides = pd.merge(chi_rides_from, chi_rides_to, how='outer', on=['Dat
e', 'station_name'])
chi_rides['total'] = chi_rides.apply(add_from_and_to, axis=1)
chi_rides = chi_rides[['Date', 'station_name', 'total']].sort_values('to
tal', ascending=False)
```

In [236]:
```
dc['Start date'] = pd.to_datetime(dc['Start date'])
dc['Date'] = dc['Start date'].dt.date
dc_rides_from = dc.groupby(['Date', 'Start station']).count()[['Start st
ation number']].reset_index()
dc_rides_from.columns = ['Date', 'station_name', 'from_count']
dc_rides_to = dc.groupby(['Date', 'End station']).count()[['Start statio
n number']].reset_index()
dc_rides_to.columns = ['Date', 'station_name', 'to_count']
dc_rides = pd.merge(dc_rides_from, dc_rides_to, how='outer', on=['Date',
'station_name'])
dc_rides['total'] = dc_rides.apply(add_from_and_to, axis=1)
dc_rides = dc_rides[['Date', 'station_name', 'total']].sort_values('tota
l', ascending=False)
```

In [237]:
```
ny['starttime'] = pd.to_datetime(ny['starttime'])
ny['Date'] = ny['starttime'].dt.date
ny_rides_from = ny.groupby(['Date', 'start station name']).count()[['bik
eid']].reset_index()
ny_rides_from.columns = ['Date', 'station_name', 'from_count']
ny_rides_to = ny.groupby(['Date', 'end station name']).count()[['bikeid'
]].reset_index()
ny_rides_to.columns = ['Date', 'station_name', 'to_count']
ny_rides = pd.merge(ny_rides_from, ny_rides_to, how='outer', on=['Date',
'station_name'])
ny_rides['total'] = ny_rides.apply(add_from_and_to, axis=1)
ny_rides = ny_rides[['Date', 'station_name', 'total']].sort_values('tota
l', ascending=False)
```

In [238]:
```
chi_10 = chi_rides['station_name'].unique()[0:10]
dc_10 = dc_rides['station_name'].unique()[0:10]
ny_10 = ny_rides['station_name'].unique()[0:10]
```

In [239]:
```
chi_rides_10 = chi_rides[chi_rides['station_name'].isin(chi_10)]
dc_rides_10 = dc_rides[dc_rides['station_name'].isin(dc_10)]
ny_rides_10 = ny_rides[ny_rides['station_name'].isin(ny_10)]
```

Question 1.2.1 Implementation and Results

For my simluation procedure, I am using the UCB algorithm to achieve a logarithmic (or close to logarithmic) regret for choosing intersections. Regret, in this case, is the average reward achieved from the optimal path minus the average reward from path we took (path being the intersections we visit before finding the optimal intersection). In this process, we will assume the rewards are sub-gaussian, as you have an unlimited amount of fliers you can pass out. We are instantiating our parameters with the observed means from the data, and a general variance across the top 10 intersections. At each step, we look at our past rewards for each arm and take the mean of the rewards and store it, and from those means, we use them to calculate our confidence bounds with the following function if $T! = 0$: $\mu_a + \sqrt{\frac{2\sigma}{T_a \log(t)}}$. We choose the widths of the upper confidence bounds by the constant that is multiplied by our variance (σ).

Simulation

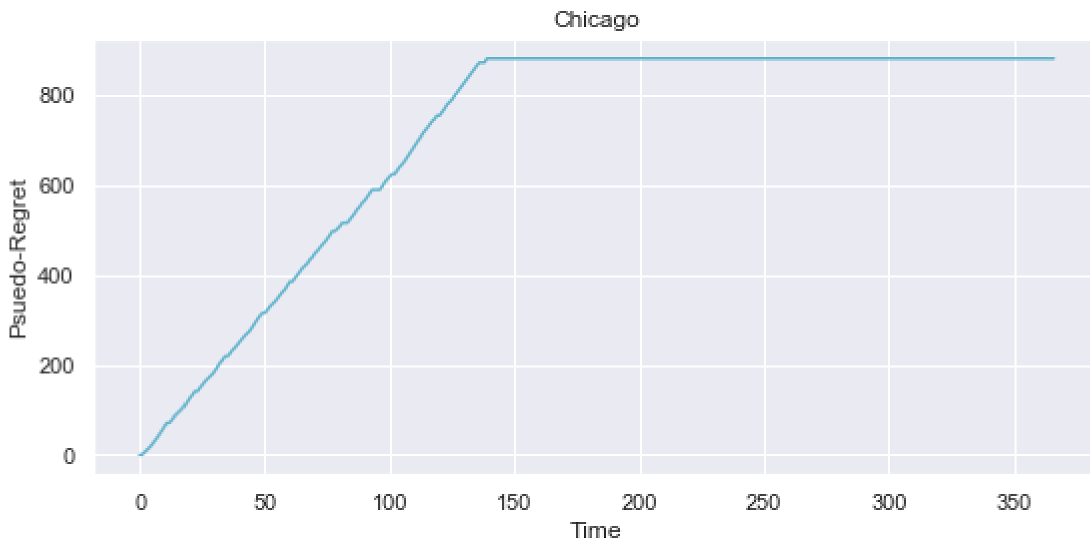
In [279]:
```
def UCB_pull_arm(t,variance,times_pulled,rewards):
    """
    Implement the choice of arm for the UCB algorithm

    Inputs:
    iteration           - iteration of the bandit algorithm
    times_pulled        - a list of length K (where K is the number of ar
ms) of the number of
                          times each arm has been pulled
    rewards             - a list of K lists. Each of the K lists holds th
e samples received from pulling each arm up
                          to iteration t.

    Returns:
    arm                 -  integer representing the arm that the UCB algo
rithm would choose.
    confidence bounds   -  a list of the upper confidence bounds for each
arm
    """

    K=len(times_pulled)
    delta=1.0/t**2

    confidence_bounds=[]
    means=[]
    for arm in np.arange(K):
        mu_a = np.mean(rewards[arm])
        T_a = times_pulled[arm]
        if T_a == 0:
            cb = np.inf
        else:
            cb = mu_a + np.sqrt((2*variance)/T_a*np.log(t))
        confidence_bounds.append(cb)
        means.append(mu_a)
    arm=np.argmax(confidence_bounds)

    return arm, confidence_bounds, means
```

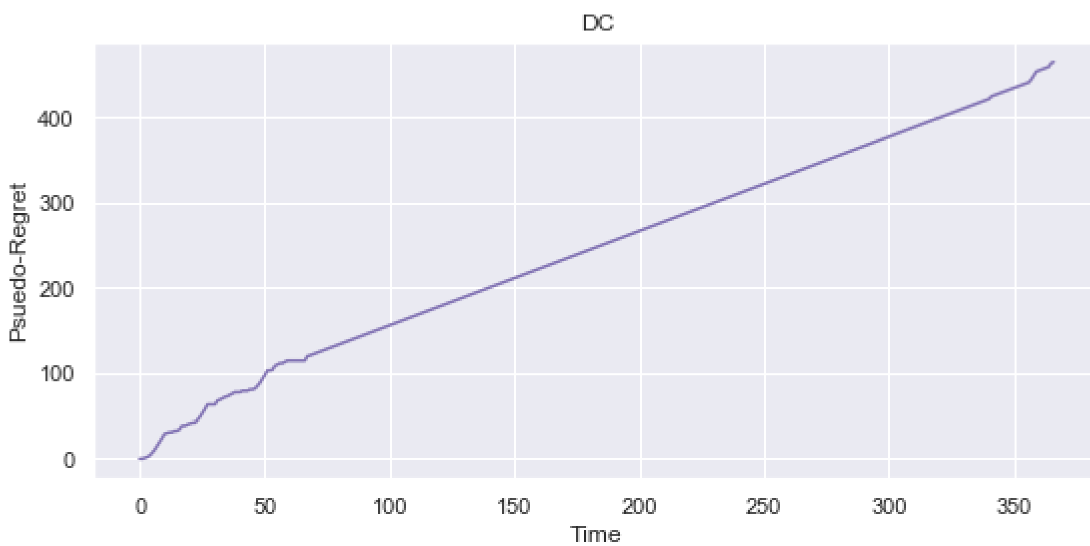
Regret over Time

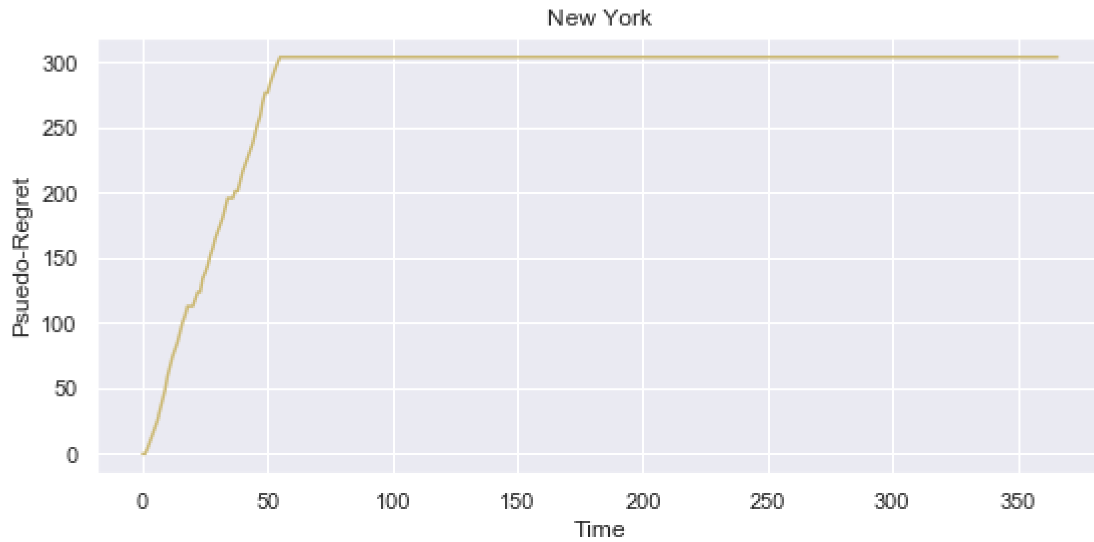
In [280]:
```
city_names = ['Chicago', 'DC', 'New York']
colors = ['c', 'm', 'y']

index = 0

times_pulled = [[], [], []]
arm_means = [[], [], []]
c_bounds = [[], [], []]

for df in [chi_rides_10, dc_rides_10, ny_rides_10]:

    grouped = df.groupby('station_name').mean().sort_values('total', asc
ending=False)['total']
    means=grouped.values
    variance=np.var(df['total'])
    standard_deviations=[np.sqrt(variance) for arm in range(len(means))]
    arm_to_station = {index: grouped.index.tolist()[index] for index in
np.arange(len(grouped))}
    bandit_env=BanditEnv(means,standard_deviations,df,arm_to_station)

    #Initialize Figure
    T=366
    num_runs=20

    plt.rcParams['figure.figsize']=[9,4]
    plt.figure()

    #Initialize pseudo-regret
    UCB_pseudo_regret=0
    for runs in range(num_runs):
        #Initialize Bandit_environment
        bandit_env.initialize(make_plot=0)
        for t in range(1,T+1):
            #Choose arm using UCB algorithm
            arm,confidence_bounds,means=UCB_pull_arm(t,variance,bandit_e
nv.times_pulled,bandit_env.rewards)

            #Pull Arm
            bandit_env.pull_arm(arm, t)
            if runs == 0:
                times_pulled[index].append(bandit_env.times_pulled.copy
())
                arm_means[index].append(means.copy())
                c_bounds[index].append(confidence_bounds.copy())


        #Keep track of pseudo-regret
        UCB_pseudo_regret+=np.array(bandit_env.regret)

    #Make plot
    ax = plt.plot(UCB_pseudo_regret/num_runs, color=colors[index])
    ax = plt.xlabel('Time')
    ax = plt.ylabel('Psuedo-Regret')
    ax = plt.title(city_names[index])

    index += 1
```

```
plt.show()
```

```
/Users/tannerarrizabalaga/anaconda3/lib/python3.7/site-packages/numpy/c
ore/fromnumeric.py:3118: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/Users/tannerarrizabalaga/anaconda3/lib/python3.7/site-packages/numpy/c
ore/fromnumeric.py:3118: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/Users/tannerarrizabalaga/anaconda3/lib/python3.7/site-packages/numpy/c
ore/fromnumeric.py:3118: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
```



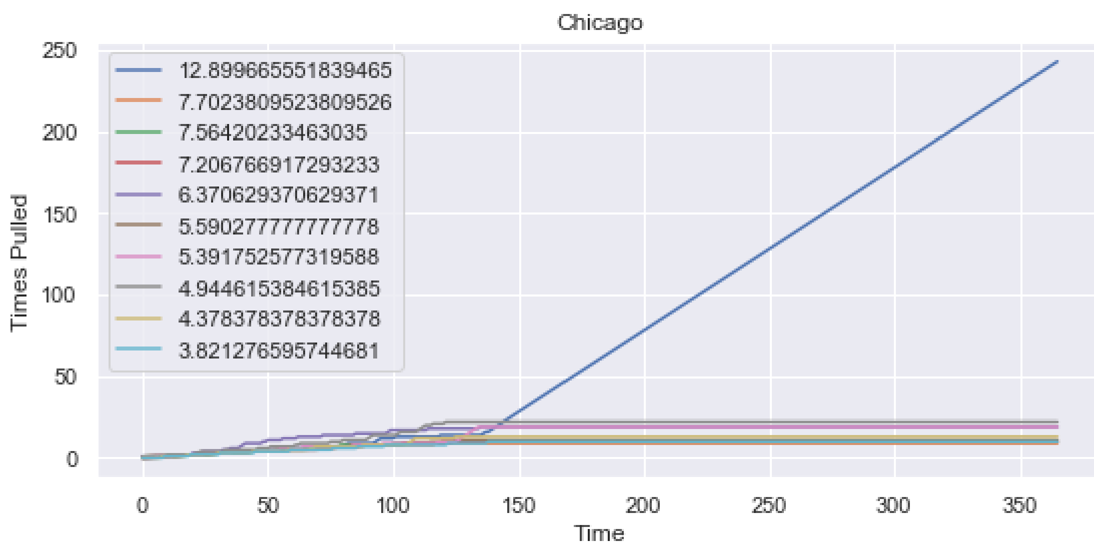




Times Pulled by Arm

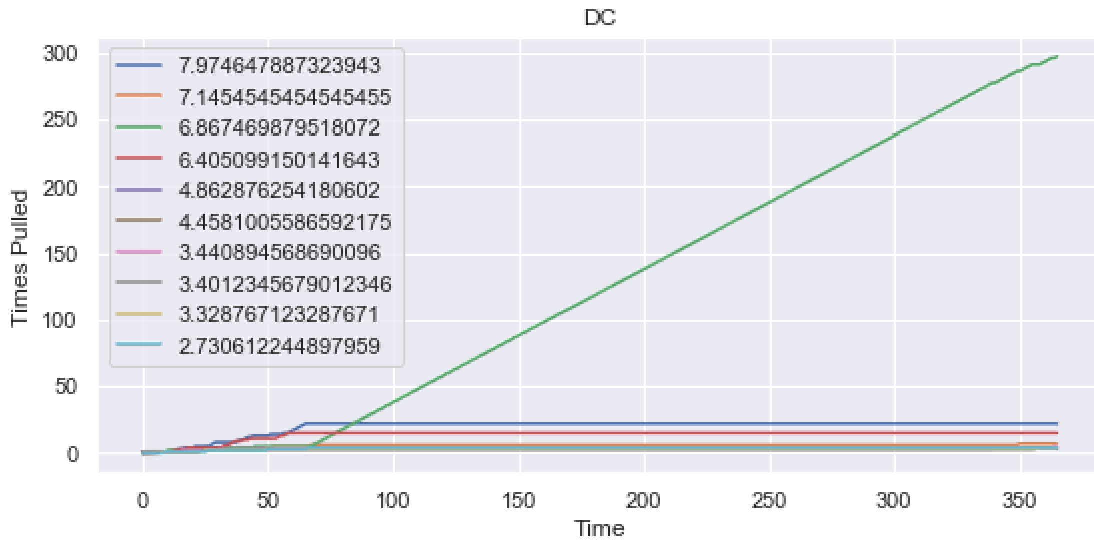
In [308]:
```
plt.plot(times_pulled[0])
plt.xlabel('Time')
plt.ylabel('Times Pulled')
plt.title('Chicago')
plt.legend(chi_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[308]: <matplotlib.legend.Legend at 0x1a287f2c88>



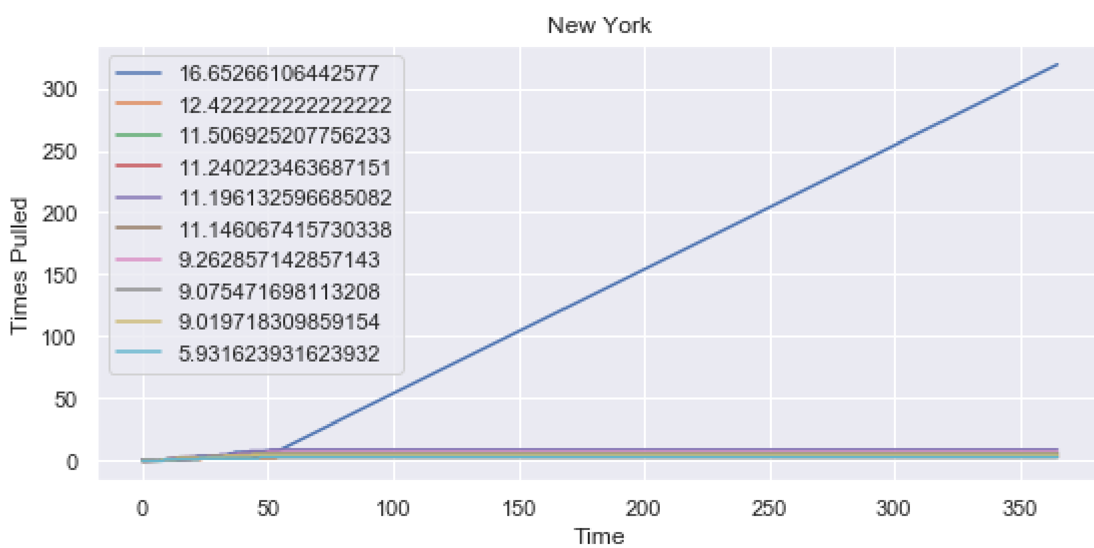
In [309]:
```python
plt.plot(times_pulled[1])
plt.xlabel('Time')
plt.ylabel('Times Pulled')
plt.title('DC')
plt.legend(dc_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[309]: <matplotlib.legend.Legend at 0x1a28db6a20>



In [310]:
```python
plt.plot(times_pulled[2])
plt.xlabel('Time')
plt.ylabel('Times Pulled')
plt.title('New York')
plt.legend(ny_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

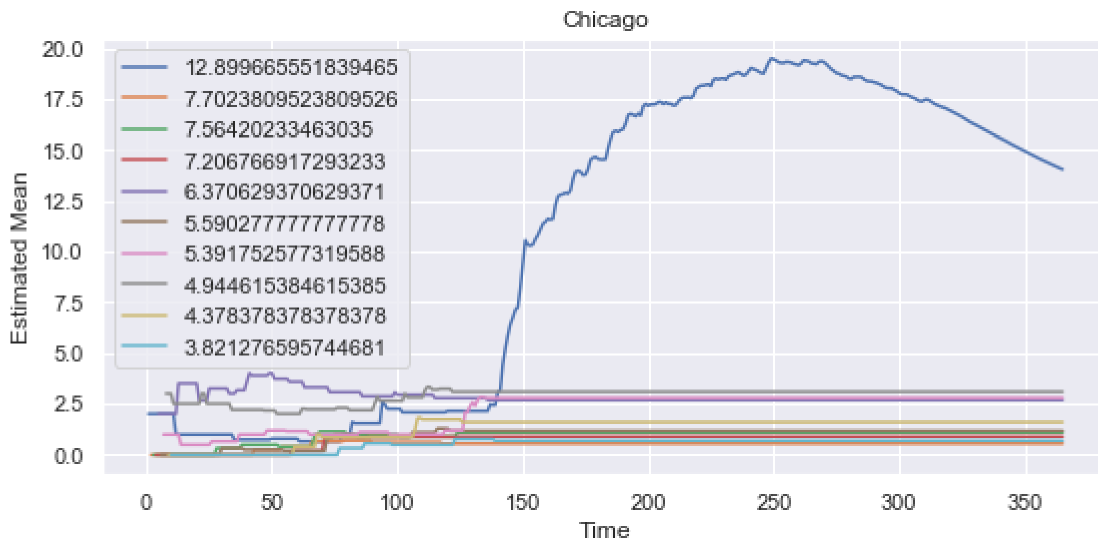
Out[310]: <matplotlib.legend.Legend at 0x1a2a8cfc50>



Estimated Means over Time

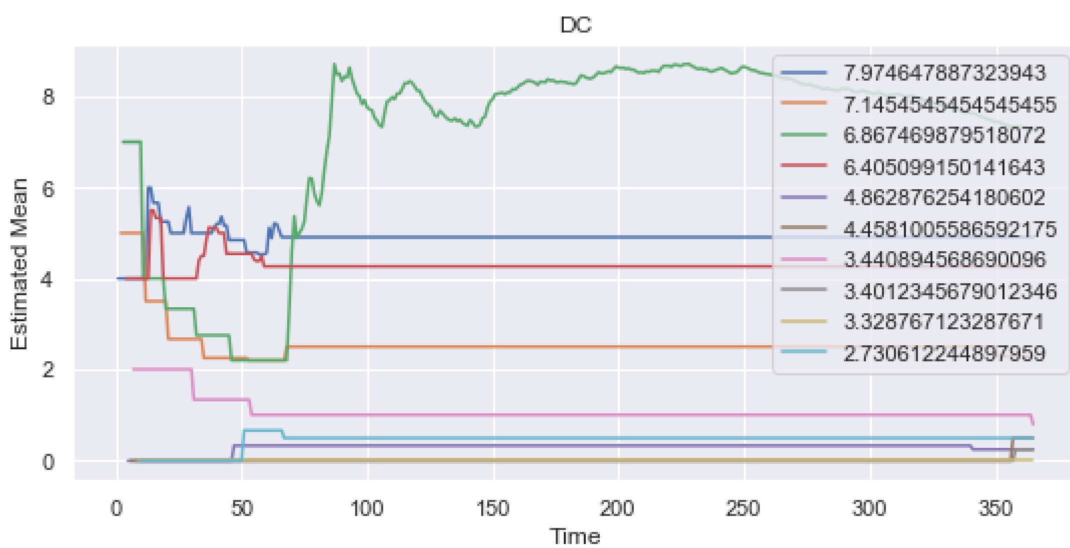
In [311]:
```
plt.plot(arm_means[0])
plt.xlabel('Time')
plt.ylabel('Estimated Mean')
plt.title('Chicago')
plt.legend(chi_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[311]: <matplotlib.legend.Legend at 0x1a2db72a90>



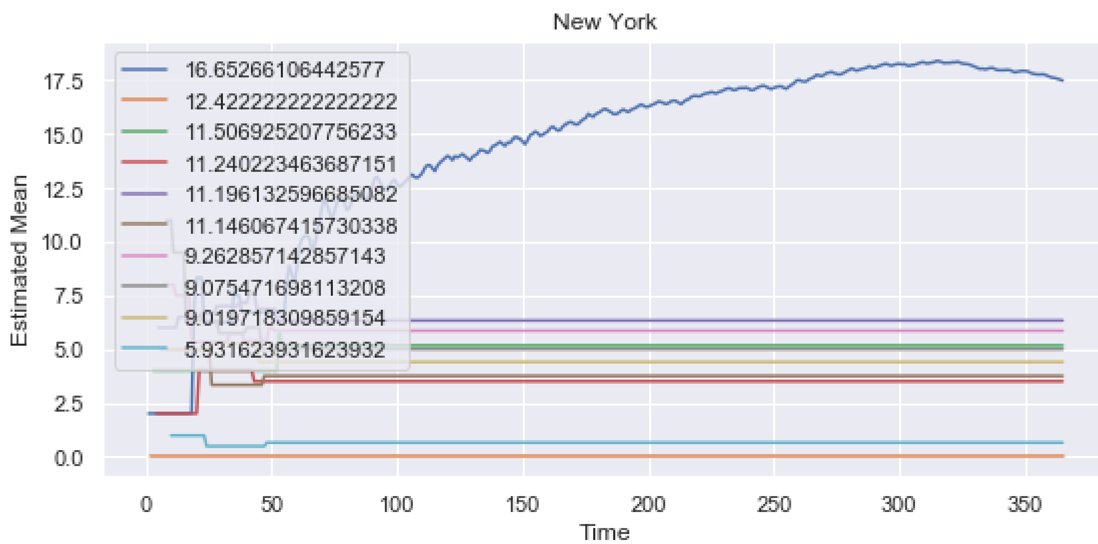
In [312]:
```
plt.plot(arm_means[1])
plt.xlabel('Time')
plt.ylabel('Estimated Mean')
plt.title('DC')
plt.legend(dc_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[312]: <matplotlib.legend.Legend at 0x1a2fb25e80>



In [313]:
```
plt.plot(arm_means[2])
plt.xlabel('Time')
plt.ylabel('Estimated Mean')
plt.title('New York')
plt.legend(ny_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

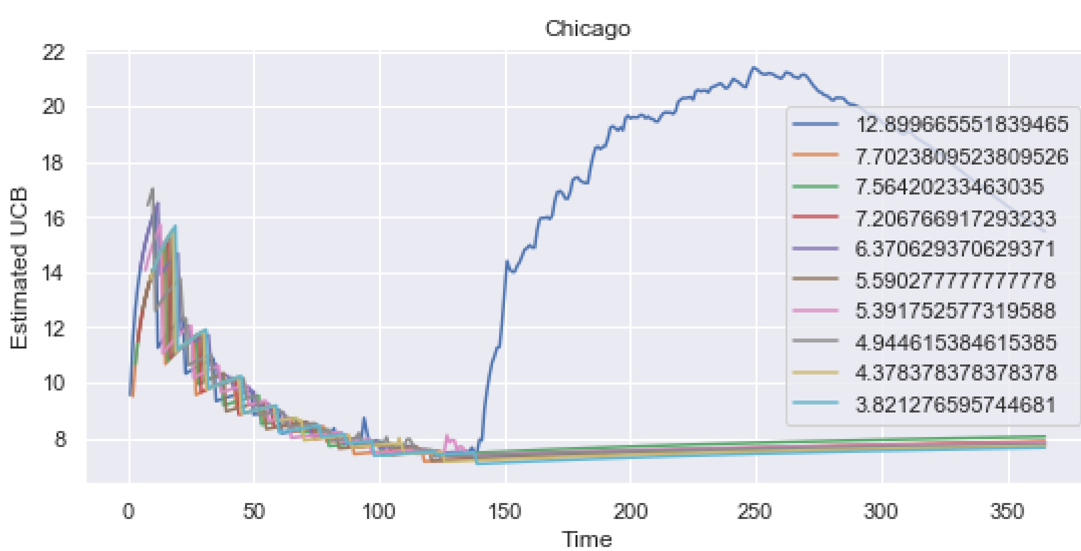
Out[313]: <matplotlib.legend.Legend at 0x1a30227710>



Estimated UCB over Time

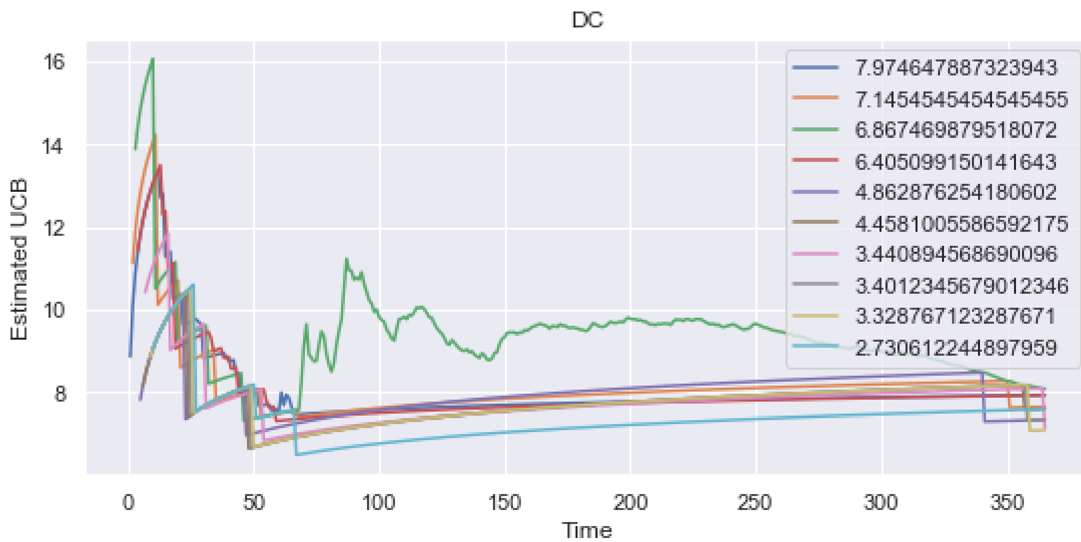
In [314]:
```
plt.plot(c_bounds[0])
plt.xlabel('Time')
plt.ylabel('Estimated UCB')
plt.title('Chicago')
plt.legend(chi_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[314]: <matplotlib.legend.Legend at 0x1a30ee8e10>



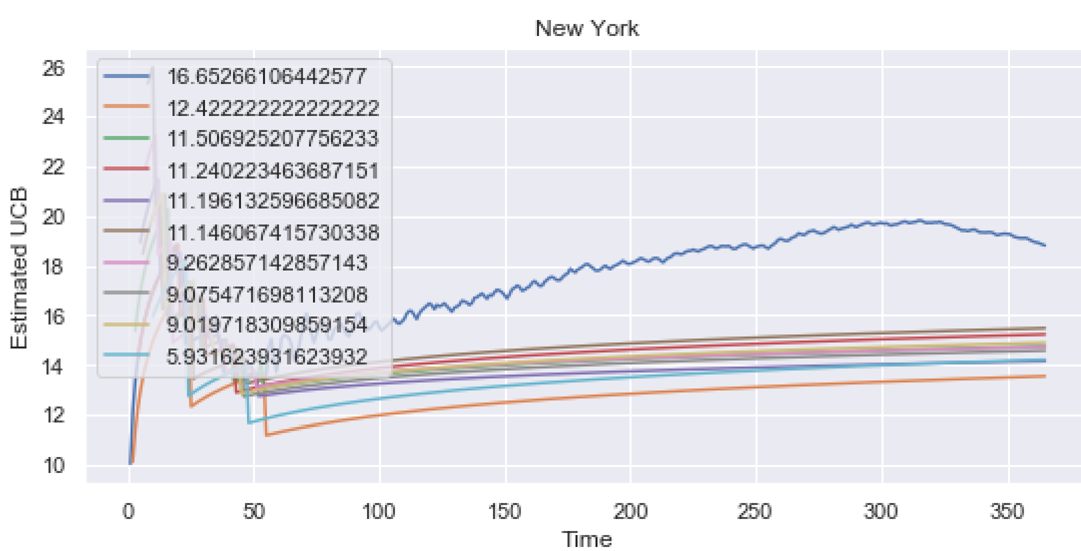
In [315]:
```
plt.plot(c_bounds[1])
plt.xlabel('Time')
plt.ylabel('Estimated UCB')
plt.title('DC')
plt.legend(dc_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[315]: <matplotlib.legend.Legend at 0x1a314415c0>



In [316]:
```
plt.plot(c_bounds[2])
plt.xlabel('Time')
plt.ylabel('Estimated UCB')
plt.title('New York')
plt.legend(ny_rides_10.groupby('station_name').mean().sort_values('tota
l', ascending=False)['total'].values)
```

Out[316]: <matplotlib.legend.Legend at 0x1a31da5eb8>



Question 1.2.2 Discussion

Yes, for some of the different orderings we see that the algorithm decides to go for another arm. This usually only happens when the arm we choose has a similar true mean. For an adaptive strategy, I believe accounting for the day of the week or season could really help us. For instance, if we made two different algorithms split by weekday and weekend, we may get different results, and one intersection that is not the optimal intersection on weekdays may do very well on weekends. This can be further improved by splitting on seasons as well, as all cities experience serious drop offs in visit rates during the Winter.

Question 1.3

The applicability of my simulation/algorithm to the problem I formalized is solid, but I believe there are *much better* methods to approaching this problem. The UCB algorithm was able to find the optimal intersection across all 3 locations, and I do believe we can invest in a promotional program founded on the results of the simulation. However, the simulation ran on **very naive assumptions that likely do not hold in the real world**. For instance, we never kept track of the type of day (weekday vs. weekend), weather, or season for the location, and we assumed the popularity of one location will not affect the traffic at another station. In additions, we had violations associated with the distributions of each intersection, as UCB assumes the same probability distribution for each intersection no matter the time step we are at, and by the nature of the data, this is false.

Ultimately, I would not recommend using UCB to place the person handing out promotional flyers. I would suggest traditional business development strategies to tackle this problem as this method is too complicated and time consuming to tackle a problem that is already well defined -- using visualizations/pure metrics about intersection and make a decision with more business knowledge as well the route I would choose. If I were to alter the algorithm, I would do the split I discussed in the problem above to account for things like weather and seasonality. I believe a solid way to promote the company as well instead of fliers is to move towards digital marketing and promote on social media, as peer-to-peer marketing is out-of-date.

Question 2

Question 2.1 EDA

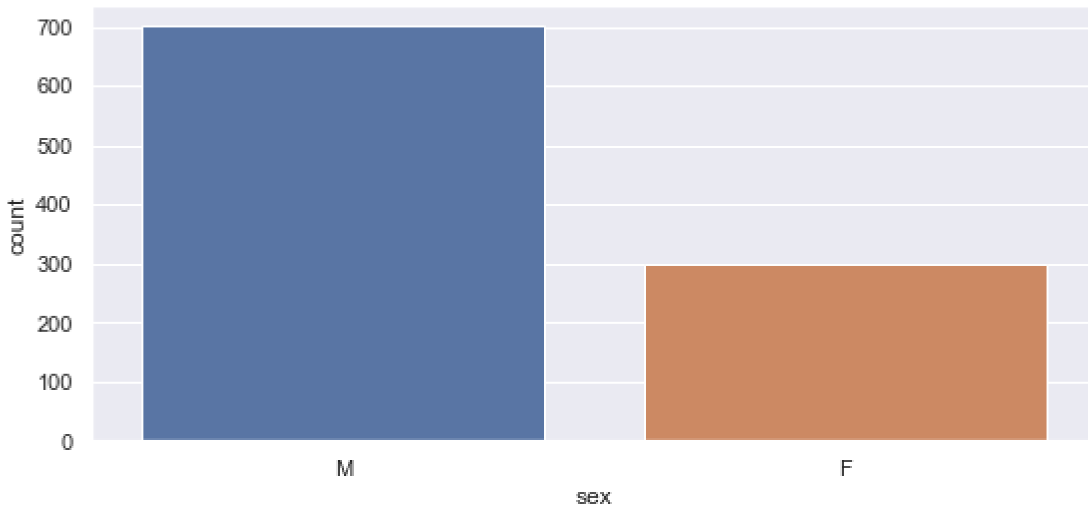
In [103]:
```
leaked = pd.read_csv('leaked.csv')
berk = pd.read_csv('berkeley.csv')
leaked.head()
```

Out[103]:

	name	sex	zip	month	year
0	Avery Phillips	M	94709	3	1993
1	Grayson Rodriguez	M	94705	6	1998
2	Ethan Baker	M	94712	1	1998
3	Carter Wright	M	94720	7	1995
4	Elijah Young	M	94706	2	1996

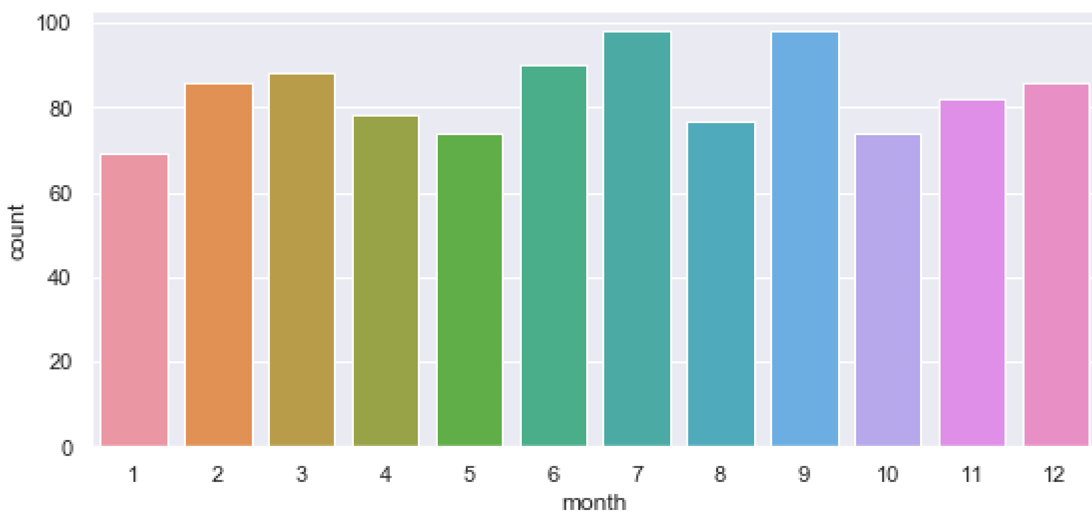
In [104]:
```
sns.countplot('sex', data=leaked)
```

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x1a29dee6a0>



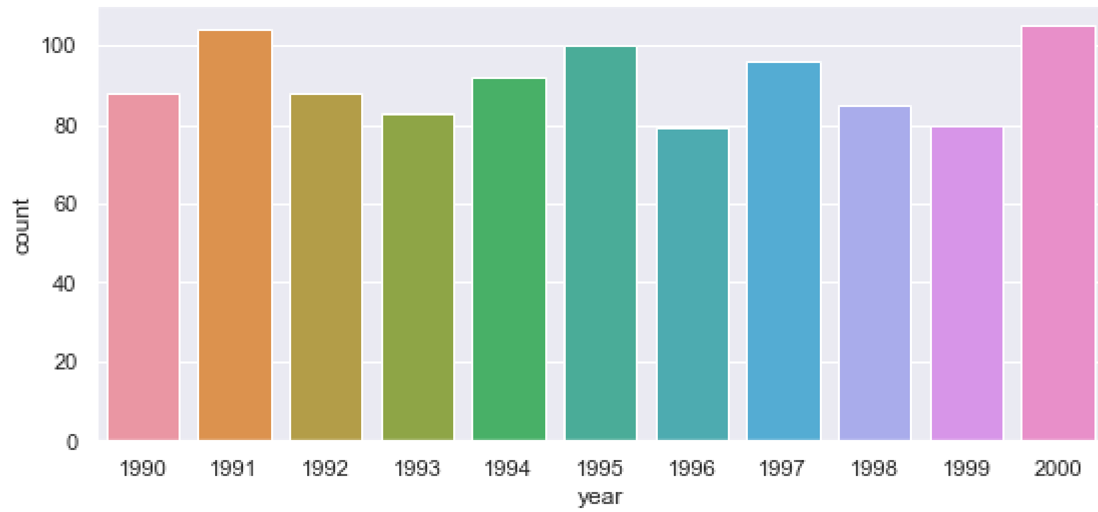
In [105]:
```
sns.countplot(leaked['month'])
```

Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2a562518>



In [106]:
```
sns.countplot(leaked['year'])
```

Out[106]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2d2c9710>
```



Of the three attributes, two of them are uniformally distributed (month and year) while the last one is not uniformally distributed (sex, there are a lot more males than females). These distributions are the same for the uniformally distributed attributes, but the sex attribute is not the same, as previous datasets had a more even ratio between guys and girls.

Question 2.2 Simple Proof of Concept

Create the identifiable dataset. From the code cell below, we see that 43 users (roughly 17%) can be identified in the leaked dataset with only the sex, month, and year columns.

In [209]:
```
grouped = leaked.groupby(['sex', 'month', 'year']).count()[['name']].res
et_index()
grouped = grouped[grouped['name'] == 1]

iden = pd.merge(leaked, grouped[['sex', 'month', 'year']], how='inner',
on=['sex', 'month', 'year'])
print(len(iden), 'users can be identified in the leaked dataset with jus
t these 3 attributes')
print(len(iden)/len(leaked.groupby(['sex', 'month', 'year']).count()[['n
ame']].reset_index()))
```

```
43 users can be identified in the leaked dataset with just these 3 attr
ibutes
0.16862745098039217
```

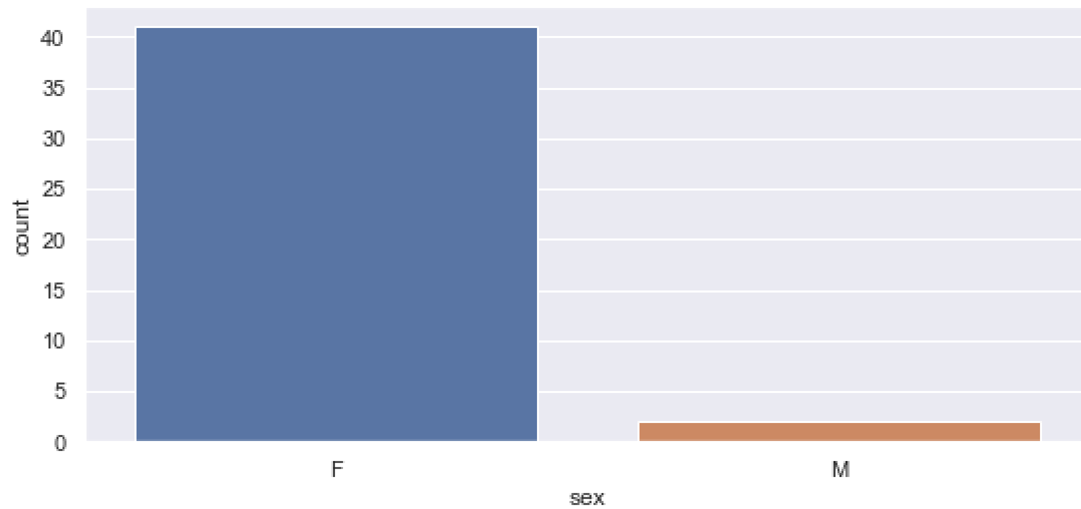
In [108]:
```
iden.head()
```

Out[108]:

	name	sex	zip	month	year
0	Emily Phillips	F	94702	1	1995
1	Evelyn Johnson	F	94701	7	1998
2	Abigail King	F	94703	1	1999
3	Sophia Brown	F	94705	12	1992
4	Olivia White	F	94706	5	2000

In [109]:
```
sns.countplot('sex', data=iden)
```

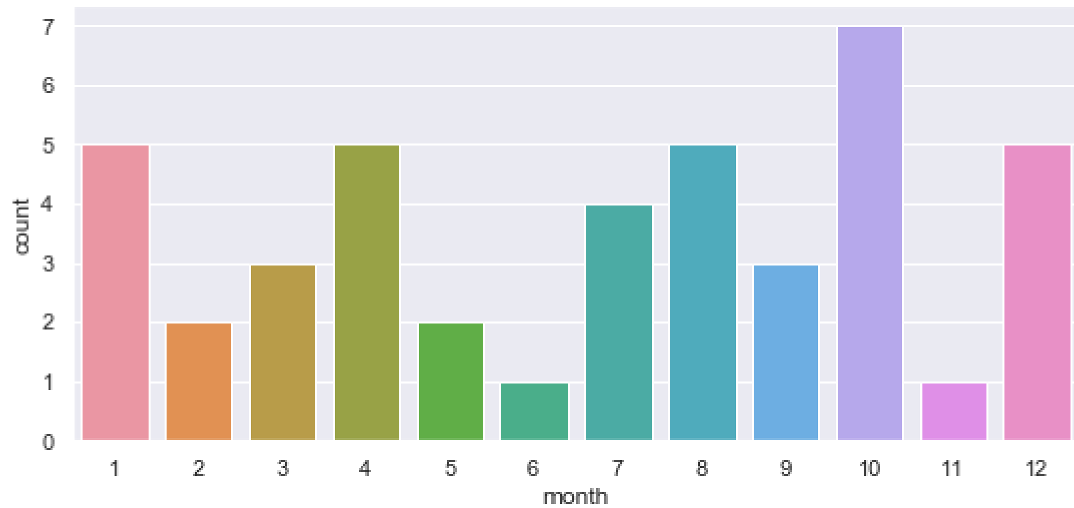
Out[109]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2a5a4828>



The distribution above clearly does not match our previous distribution plotted for all of 'leaked'. In 'leaked', we see that there are roughly twice the number of males than females in the dataset, whereas in the identifiable dataset, there are only two men who can be identified by the three attributes.

In [110]:
```
sns.countplot(iden['month'])
```

Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2d40d5f8>



In terms of the month, we see that the plot above is roughly similar our original plot which was uniformally distributed, the dips and peaks in the datset can be attributed to the fact that there were only 42 identifiable users.

In [113]:
```
iden_rides = pd.merge(berk, iden, how='inner', on=['sex', 'month', 'yea
r'])
```

In [116]:
```
len(iden_rides)
```

Out[116]: 2205

The script above does an inner join on the identifiable riders and the berkeley dataset, giving us all of their rides in Berkeley.

Question 2.3 A More Elaborate Attack

In order to obtain p_1 , p_2 in our dataset, we will be running a simple script to that does element-wise comparison between where a user starts/ends and where they live. From the below code snippet, we see that roughly 90% of our users start their rides from their registered zip and roughly 73% of users end their rides at their registered zip.

In [212]:
```
p1 = np.count_nonzero(iden_rides['start'] != iden_rides['zip']) / len(id
en_rides)
p2 = np.count_nonzero(iden_rides['end'] != iden_rides['zip']) / len(iden
_rides)
```

In [213]:
```
p1, p2
```

Out[213]:
```
(0.09659863945578231, 0.2780045351473923)
```

In order to produce a confidence interval, I have run a simulation 1000 times where we are resampling from our dataset with replacement (sample size = # of samples in iden_rides), recording p_1 and p_2, and then dumping them into two respective arrays.

After we run the simulation, I use the empirical rule ($\mu - 2\sigma$, $\mu + 2\sigma$) to create a confidence interval.

In [217]:
```
n = 1000
p1s = []
p2s = []

for trial in np.arange(n):
    sample_size = len(iden_rides)
    sample = iden_rides.sample(sample_size, replace=True)
    p1s.append(np.count_nonzero(sample['start'] != sample['zip']) / samp
le_size)
    p2s.append(np.count_nonzero(sample['end'] != sample['zip']) / sample
_size)
```

In [218]:
```
p1_mu, p1_std = np.mean(p1s), np.std(p1s)
p2_mu, p2_std = np.mean(p2s), np.std(p2s)

p1_conf = (p1_mu - 2 * p1_std, p1_mu + 2 * p1_std)
p2_conf = (p2_mu - 2 * p2_std, p2_mu + 2 * p2_std)
```

In [219]:
```
p1_conf, p2_conf
```

Out[219]:
```
((0.08381369936701752, 0.10950421446518205),
 (0.25826068933620383, 0.29699554649146054))
```

In order to get theoritically identifiable users, I have grouped by zip, sex, month, and year, found all users that occur once, and list them. From this method, 720 users (roughly 85%) can be identified.

In [220]:
```
theory = leaked.groupby(['zip', 'sex', 'month', 'year']).count()[['name'
]].reset_index()
theory = theory[theory['name'] == 1]
```

In [208]:
```
print(len(theory), 'users can be identified in the leaked dataset with j
ust these 3 attributes')
print(len(theory)/len(leaked.groupby(['zip', 'sex', 'month', 'year']).co
unt()[['name']].reset_index()))
```

```
720 users can be identified in the leaked dataset with just these 3 att
ributes
0.8450704225352113
```

In [206]:
```
def jaccard(leak, berk):
    intersection_count = np.count_nonzero(leak == berk)
    union_count = len(leak)
    stat = intersection_count / union_count
    return stat

name_predictions = []

#not_identifiable =
index = 0
for berk_row in berk.iterrows():
    index+=1
    if index % 1000 == 0:
        print(index)
    leaked_rows = leaked[leaked['zip'] == berk_row[1]['start']]
    sims = {}
    for leak_row in leaked_rows.iterrows():
        adjusted_leak_row = list(leak_row[1][1:].values)
        adjusted_berk_row = berk_row[1][0:len(berk_row[1])-1].values
        adjusted_berk_row = [adjusted_berk_row[0], adjusted_berk_row[3],
adjusted_berk_row[1], adjusted_berk_row[2]]
        sims[leak_row[1]['name']] = jaccard(np.array(adjusted_leak_row),
adjusted_berk_row)
    df = pd.DataFrame(list(sims.items()))
    df.columns = ['Name', 'Similarity']
    name_predictions.append(df.sort_values('Similarity', ascending=False
).iloc[0, 0])
```

```
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
29000
30000
31000
32000
33000
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000
50000
51000
52000
53000
54000
```

In [207]:
```
name_predictions
```

```
Out[207]: ['Joseph Williams',
 'Wyatt Smith',
 'Daniel Wilson',
 'Carter Robinson',
 'John Phillips',
 'Carter Robinson',
 'Benjamin Robinson',
 'Ella Wilson',
 'James Jackson',
 'Noah Moore',
 'Noah Johnson',
 'Wyatt Carter',
 'Grayson Collins',
 'Olivia Davis',
 'Abigail Harris',
 'James Lee',
 'Michael Jackson',
 'Harper Carter',
 'Luke Green',
 'Harper Carter',
 'Elizabeth Miller',
 'Luke Davis',
 'Daniel Baker',
 'Grayson White',
 'Sophia Rodriguez',
 'Luke Turner',
 'Samuel Lewis',
 'Abigail Hill',
 'Mason Campbell',
 'Oliver Smith',
 'Sophia Rodriguez',
 'Dylan Clark',
 'Jayden Moore',
 'James Jackson',
 'Joseph Anderson',
 'Ethan Adams',
 'Elizabeth Adams',
 'Mason Young',
 'Mason Lewis',
 'Ethan Walker',
 'Dylan Walker',
 'Jackson Martin',
 'Carter Gonzalez',
 'Andrew Thompson',
 'Alexander Lopez',
 'Andrew Gonzalez',
 'Benjamin Campbell',
 'Harper Jones',
 'Emma Lee',
 'Wyatt Clark',
 'Ella Perez',
 'Emma Martin',
 'Joseph Anderson',
 'Sophia Campbell',
 'Luke Nelson',
 'Noah Green',
 'Benjamin Lewis',
```

```
'Henry Brown',
'David Collins',
'Olivia Allen',
'Charlotte King',
'Ella Gonzalez',
'Henry Davis',
'Ella Evans',
'Noah Adams',
'Daniel Martinez',
'Matthew Robinson',
'Abigail Taylor',
'Sebastian Thompson',
'Luke Garcia',
'Abigail Hill',
'Abigail Miller',
'Daniel Miller',
'Jack White',
'Evelyn Evans',
'David Smith',
'Abigail Harris',
'Mia Davis',
'Benjamin Jackson',
'Dylan Garcia',
'Wyatt Lopez',
'Benjamin Lewis',
'Henry Walker',
'Benjamin Turner',
'Michael Parker',
'Lucas Thomas',
'Liam Allen',
'Andrew Lopez',
'Amelia Green',
'Lucas Edwards',
'Grayson Thomas',
'Mason Campbell',
'Evelyn Rodriguez',
'David Nelson',
'Mila Lewis',
'Sophia Brown',
'Dylan Lewis',
'Aiden Williams',
'Michael Brown',
'Carter Thompson',
'Elijah Lee',
'Ethan Carter',
'Matthew Garcia',
'Logan Rodriguez',
'Lucas Smith',
'Owen Edwards',
'Evelyn Rodriguez',
'Oliver King',
'Liam Adams',
'Jackson Wright',
'Abigail White',
'Ella Mitchell',
'Jackson Smith',
'John Jones',
```

```
               'Mason Nelson',
               'Logan Evans',
               'Grayson Collins',
               'William Johnson',
               'Carter Garcia',
               'Emily Baker',
               'James Martin',
               'Michael Mitchell',
               'Jackson Carter',
               'Sophia Rodriguez',
               'Abigail White',
               'Noah Johnson',
               'Lucas Clark',
               'Liam Brown',
               'Sebastian King',
               'Dylan Walker',
               'Samuel Moore',
               'Liam Campbell',
               'Grayson Baker',
               'Luke Wright',
               'Alexander Campbell',
               'Emily Campbell',
               'Emily Evans',
               'Oliver King',
               'Grayson Rodriguez',
               'Alexander Allen',
               'Dylan Nelson',
               'Ethan Walker',
               'Michael Hall',
               'Jacob Green',
               'Mason Campbell',
               'David Rodriguez',
               'William Mitchell',
               'David Adams',
               'Liam Gonzalez',
               'Harper Evans',
               'Jack Turner',
               'Elizabeth Lee',
               'Grayson Smith',
               'Avery Hill',
               'William Mitchell',
               'Dylan Davis',
               'Jayden Young',
               'Olivia Davis',
               'Alexander Miller',
               'Sebastian King',
               'Owen Brown',
               'Sebastian Hill',
               'James Rodriguez',
               'Abigail Thomas',
               'Grayson Miller',
               'Mia Jackson',
               'William Robinson',
               'Wyatt Garcia',
               'Oliver Parker',
               'Avery Roberts',
               'James Parker',
```

```
'Avery Jones',
'Daniel Carter',
'Carter Thomas',
'William Lopez',
'Alexander Campbell',
'Carter Robinson',
'William Mitchell',
'Amelia Mitchell',
'Aiden Smith',
'Amelia Gonzalez',
'Emma Jones',
'Liam Campbell',
'Jayden Lee',
'Isabella Moore',
'Elijah Young',
'Harper Lopez',
'Elizabeth Martinez',
'David Anderson',
'Dylan Hall',
'Wyatt Clark',
'Ethan Edwards',
'Daniel Jackson',
'Mia Perez',
'Lucas Roberts',
'Grayson Miller',
'Andrew Harris',
'Ava Miller',
'William Martin',
'Benjamin Phillips',
'Sebastian Scott',
'Isabella Martin',
'Lucas Martinez',
'Matthew Phillips',
'James Brown',
'Lucas Martinez',
'Emma Rodriguez',
'Amelia Mitchell',
'Michael Jackson',
'Amelia Davis',
'Alexander Walker',
'Ella White',
'Harper Harris',
'Emma Thompson',
'John King',
'Aiden Robinson',
'Carter Smith',
'Evelyn Young',
'Ethan Hernandez',
'Amelia Clark',
'Logan Evans',
'Jayden Scott',
'Andrew Johnson',
'Michael Miller',
'Liam Harris',
'Harper Taylor',
'Noah Martin',
'Emma Lee',
```

```
                'Dylan Garcia',
                'Ava Thomas',
                'Mason Jackson',
                'Oliver Jackson',
                'Liam Brown',
                'Henry Parker',
                'Carter Thompson',
                'Elizabeth Martinez',
                'Mila Baker',
                'Carter Scott',
                'Dylan Thompson',
                'Jacob Parker',
                'Logan Turner',
                'Emily Carter',
                'Carter Edwards',
                'Dylan Edwards',
                'Carter Gonzalez',
                'Matthew Martinez',
                'Oliver Phillips',
                'Ella Evans',
                'Mia Hernandez',
                'Ella Lopez',
                'John Hernandez',
                'Sebastian Thompson',
                'Luke Turner',
                'David Walker',
                'Logan Perez',
                'Dylan Davis',
                'Abigail Lee',
                'Ethan Carter',
                'Ethan White',
                'Abigail Thomas',
                'Grayson Mitchell',
                'Elijah Mitchell',
                'Grayson Baker',
                'Benjamin Campbell',
                'Sophia Edwards',
                'Wyatt Scott',
                'David Nelson',
                'Oliver King',
                'Aiden Williams',
                'William Martinez',
                'Emily Martin',
                'Jacob Baker',
                'Harper King',
                'Jayden Thompson',
                'David Taylor',
                'Dylan Thompson',
                'Harper Adams',
                'William Martinez',
                'William Phillips',
                'Mason King',
                'Carter Thomas',
                'Liam Evans',
                'Elizabeth Hall',
                'Jacob Perez',
                'Carter Hernandez',
```

```
'Daniel Jackson',
'Luke King',
'Emily Phillips',
'Sebastian Garcia',
'David Adams',
'Harper Lopez',
'Lucas Wright',
'Andrew Gonzalez',
'Evelyn Young',
'Henry Harris',
'Ava Edwards',
'Oliver Parker',
'Sebastian Garcia',
'Owen Mitchell',
'Emma Jones',
'Andrew Gonzalez',
'John Miller',
'Jackson Jackson',
'Emily Martin',
'Jack Lopez',
'Daniel Phillips',
'Evelyn Robinson',
'Elizabeth White',
'Luke Gonzalez',
'Aiden Green',
'Harper Wright',
'Sophia Anderson',
'Evelyn Roberts',
'Luke King',
'Jackson Martin',
'Sophia Rodriguez',
'Daniel Phillips',
'Andrew Thompson',
'Logan Moore',
'Matthew Martinez',
'Ava Allen',
'Joseph Rodriguez',
'Luke Parker',
'Evelyn Phillips',
'Alexander Davis',
'Matthew Parker',
'Avery Gonzalez',
'Oliver Perez',
'Mason Green',
'Samuel Carter',
'Joseph Robinson',
'John Wilson',
'David Turner',
'Olivia White',
'Carter Lee',
'Isabella Lee',
'John Evans',
'William Johnson',
'Ethan Walker',
'Avery Evans',
'Michael Hernandez',
'Benjamin King',
```

```
'Carter Walker',
'Carter White',
'Luke Adams',
'Noah Martinez',
'Elizabeth Nelson',
'Logan Hill',
'Dylan Martin',
'Ella Turner',
'Evelyn Parker',
'Jacob Adams',
'Owen Allen',
'Daniel King',
'Noah Lewis',
'Henry Gonzalez',
'James Martin',
'Sophia Evans',
'David Rodriguez',
'Jayden Scott',
'Isabella Harris',
'Amelia Lopez',
'Grayson Phillips',
'Mila Lewis',
'Abigail Rodriguez',
'Emma Lee',
'Elizabeth Lewis',
'Jackson Moore',
'Luke Wright',
'Ava Perez',
'Carter Phillips',
'Carter Lee',
'Ethan Davis',
'Oliver Adams',
'Sophia Evans',
'Harper Moore',
'Logan Carter',
'Amelia Edwards',
'Avery Evans',
'Avery Evans',
'David Phillips',
'Mason Campbell',
'Sophia Parker',
'Isabella Mitchell',
'John Hernandez',
'Ethan Davis',
'Jack Edwards',
'Elizabeth Lewis',
'Daniel Jones',
'Jack Anderson',
'Lucas Smith',
'Elijah Scott',
'Matthew Martinez',
'Ethan Walker',
'Oliver King',
'Benjamin Taylor',
'Andrew Lopez',
'Sebastian Taylor',
'Sophia Perez',
```

```
'Mia Clark',
'Grayson Rodriguez',
'Jacob Scott',
'Dylan Davis',
'Henry Wilson',
'Alexander Davis',
'Elizabeth Roberts',
'Alexander Davis',
'Charlotte Evans',
'Alexander Lee',
'Jack White',
'Olivia Rodriguez',
'Emily Campbell',
'Isabella Young',
'Ethan Baker',
'Henry Perez',
'Olivia Moore',
'Mia Davis',
'David Turner',
'Ethan Carter',
'Logan Walker',
'Harper Moore',
'Harper Jones',
'Olivia Anderson',
'Ethan Edwards',
'Henry Turner',
'Carter Lee',
'Michael Brown',
'Amelia Turner',
'Ella Miller',
'Charlotte Thomas',
'Lucas Martinez',
'David Adams',
'Daniel Robinson',
'Carter Perez',
'Jayden Robinson',
'Sophia Wilson',
'Elijah Young',
'Wyatt Young',
'Emma Hall',
'Logan Jackson',
'Owen Mitchell',
'Grayson Lewis',
'Dylan Edwards',
'Jack White',
'Mia Martinez',
'James Martinez',
'Aiden Edwards',
'Lucas Hall',
'Benjamin Turner',
'Jayden Jones',
'Noah White',
'Ava Brown',
'Henry Davis',
'Evelyn Phillips',
'Emma King',
'Jacob Perez',
```

```
'Oliver King',
'David Wilson',
'Daniel Gonzalez',
'Elizabeth Roberts',
'Wyatt Clark',
'Evelyn Johnson',
'Benjamin Nelson',
'Ethan Garcia',
'Ella Lewis',
'Charlotte Evans',
'Olivia Lopez',
'Carter Perez',
'Emily Nelson',
'Luke Wright',
'Lucas Thomas',
'Liam Collins',
'Charlotte Mitchell',
'Ella Wilson',
'Jacob Phillips',
'Daniel Phillips',
'Jayden Young',
'Emma Scott',
'Joseph Rodriguez',
'Grayson Thompson',
'Emma Thompson',
'Harper Moore',
'Jackson Jackson',
'Daniel Allen',
'David Young',
'Jack White',
'Jayden Green',
'Ava Clark',
'Owen Nelson',
'Liam Thompson',
'David Scott',
'Andrew Perez',
'Sophia Hill',
'Logan Allen',
'Jayden Campbell',
'Evelyn Evans',
'Henry Mitchell',
'Jacob Moore',
'Noah Moore',
'Liam Lewis',
'Carter Edwards',
'Harper White',
'Carter White',
'Amelia Adams',
'David Rodriguez',
'Sebastian Garcia',
'Sebastian Clark',
'Joseph Williams',
'Liam Brown',
'Abigail Williams',
'Daniel Carter',
'Avery White',
'Ella Phillips',
```

```
'Jacob Scott',
'Avery King',
'Elizabeth Martin',
'Dylan Walker',
'Andrew Lopez',
'Sebastian Clark',
'Liam Anderson',
'Jackson Jackson',
'Amelia Turner',
'Luke Thomas',
'Jack Williams',
'Mila Rodriguez',
'Avery Roberts',
'Lucas Robinson',
'Luke Green',
'Avery Campbell',
'Charlotte Walker',
'Isabella Jackson',
'Emily Carter',
'Joseph Walker',
'Olivia Moore',
'Oliver Allen',
'Ethan Lee',
'Joseph Harris',
'Daniel Miller',
'Mason Baker',
'Jackson Lopez',
'David Collins',
'Ava Miller',
'Lucas King',
'Alexander Scott',
'Henry Hall',
'Jack Gonzalez',
'Ella Phillips',
'Sophia Lopez',
'Elizabeth Hernandez',
'William Clark',
'Evelyn Evans',
'Harper King',
'Oliver Adams',
'Henry Gonzalez',
'Jacob Baker',
'Matthew Garcia',
'Emma Davis',
'Sebastian Garcia',
'Emma Hall',
'Oliver King',
'Dylan Garcia',
'Grayson Rodriguez',
'Olivia Roberts',
'Mila Baker',
'Carter Scott',
'Elizabeth Adams',
'Henry Perez',
'Samuel Evans',
'William Robinson',
'Henry Perez',
```

```
'Liam Adams',
'Olivia Anderson',
'Noah Adams',
'Mason Nelson',
'Evelyn Hill',
'Joseph Carter',
'Matthew Scott',
'Michael Anderson',
'Jack Thompson',
'David Turner',
'Elizabeth Nelson',
'Charlotte Rodriguez',
'Benjamin Jones',
'Jackson Clark',
'Benjamin Turner',
'Jack Anderson',
'Daniel Martin',
'Michael Miller',
'Mila King',
'Grayson Phillips',
'James Carter',
'Ella Carter',
'Ella Hernandez',
'Noah Moore',
'Noah Allen',
'Daniel Baker',
'Sophia Garcia',
'Matthew Lewis',
'Lucas Wright',
'Ava Smith',
'Luke King',
'Ella Evans',
'Henry Campbell',
'Oliver Perez',
'Ella Clark',
'Jackson Johnson',
'Ella Scott',
'Wyatt Carter',
'Elijah Thompson',
'Oliver Davis',
'Liam Collins',
'Grayson Miller',
'Grayson Collins',
'James Carter',
'Sophia Rodriguez',
'Samuel Adams',
'Dylan Hall',
'Henry Mitchell',
'James Rodriguez',
'Grayson Smith',
'Oliver Allen',
'Mila Thomas',
'Mila Johnson',
'Owen White',
'Sophia Walker',
'Harper Lopez',
'John Lopez',
```

```
                  'Dylan Lewis',
                  'Sebastian Garcia',
                  'Harper Wright',
                  'Carter Robinson',
                  'Ava Edwards',
                  'Aiden Miller',
                  'Logan Smith',
                  'Alexander White',
                  'John Adams',
                  'Samuel Carter',
                  'Isabella Wilson',
                  'Logan Allen',
                  'Charlotte Evans',
                  'James Carter',
                  'Noah Thomas',
                  'Oliver Jackson',
                  'Matthew Garcia',
                  'Benjamin Taylor',
                  'Ethan Wright',
                  'Matthew Hernandez',
                  'David Collins',
                  'Henry Turner',
                  'Liam Anderson',
                  'Jack Turner',
                  'Emily Martinez',
                  'Mia Martinez',
                  'James Evans',
                  'Olivia Robinson',
                  'Jack Turner',
                  'Lucas Edwards',
                  'Andrew Thompson',
                  'Emily Allen',
                  'Evelyn Phillips',
                  'Elijah Thompson',
                  'Grayson Miller',
                  'Oliver King',
                  'Liam Lewis',
                  'Elizabeth White',
                  'Charlotte Evans',
                  'Grayson Smith',
                  'Joseph Garcia',
                  'Mason Lewis',
                  'Harper Moore',
                  'Liam Lewis',
                  'Ava Thomas',
                  'Amelia Hernandez',
                  'Abigail Rodriguez',
                  'Jack Nelson',
                  'Benjamin King',
                  'Liam Campbell',
                  'William Martin',
                  'Ethan Nelson',
                  'Grayson Rodriguez',
                  'Luke Wright',
                  'Oliver Phillips',
                  'Avery Moore',
                  'Joseph Lee',
```

```
'William Hill',
'Amelia Lopez',
'Harper Clark',
'Logan Hall',
'Lucas Taylor',
'Logan Rodriguez',
'Oliver Thompson',
'Luke Green',
'Mila Jones',
'Mila King',
'Henry Harris',
'Luke Parker',
'Luke Wright',
'Andrew Jones',
'Olivia Anderson',
'Emma Garcia',
'Sebastian Clark',
'John King',
'Noah Johnson',
'Harper Jones',
'Daniel Carter',
'John Garcia',
'Aiden Robinson',
'Avery Thomas',
'Samuel Thompson',
'David Perez',
'Logan Smith',
'James Jackson',
'David Turner',
'Elizabeth Lewis',
'Owen Carter',
'Joseph Parker',
'Ava Clark',
'William Hill',
'John Garcia',
'Liam Anderson',
'Carter Gonzalez',
'Charlotte Wilson',
'William Moore',
'Michael Hall',
'John Jones',
'Evelyn Thompson',
'Grayson Jackson',
'Dylan Thomas',
'Joseph Harris',
'Andrew Gonzalez',
'Carter Scott',
'Mila Lewis',
'Henry Turner',
'Elizabeth Hall',
'Alexander Perez',
'Jackson Lopez',
'Jayden Young',
'Andrew Baker',
'William Lopez',
'Daniel Miller',
'Olivia Green',
```

```
'Mason Evans',
'Logan Turner',
'Ella Clark',
'Noah Thomas',
'Logan Hernandez',
'Liam Young',
'Olivia Rodriguez',
'Jayden Green',
'Liam Anderson',
'Dylan Martin',
'Jackson Clark',
'Jayden Jones',
'Sophia Parker',
'Dylan Garcia',
'Andrew Miller',
'William Davis',
'Abigail Jackson',
'Sebastian Turner',
'Mia Davis',
'Daniel Gonzalez',
'Michael Mitchell',
'Jackson Martin',
'Benjamin Jackson',
'Carter Perez',
'Alexander Brown',
'Jack Williams',
'Noah Garcia',
'Grayson Collins',
'Ethan Hernandez',
'Aiden Allen',
'Grayson Thompson',
'Harper Edwards',
'Evelyn Turner',
'Sebastian Scott',
'Logan Rodriguez',
'Joseph Parker',
'Benjamin Gonzalez',
'Sebastian Hill',
'Ethan Thomas',
'Olivia Robinson',
'Wyatt Lopez',
'Owen Perez',
'Benjamin Lewis',
'Jackson Lopez',
'Samuel Thompson',
'Luke Evans',
'Elijah Hill',
'Ethan Adams',
'Jackson King',
'Owen Edwards',
'Jacob Brown',
'Avery Gonzalez',
'Michael Carter',
'Jacob Baker',
'Matthew Garcia',
'Michael Anderson',
'Michael Parker',
```

```
                 'Benjamin Jackson',
                 'Ethan Carter',
                 'Dylan Garcia',
                 'James Martinez',
                 'Jayden Perez',
                 'Ethan Clark',
                 'Elijah Lee',
                 'Evelyn Phillips',
                 'Isabella Taylor',
                 'Dylan Hall',
                 'Elijah Martinez',
                 'Joseph Harris',
                 'Carter Robinson',
                 'Wyatt Garcia',
                 'Charlotte Mitchell',
                 'Samuel Jones',
                 'Harper Wright',
                 'Grayson King',
                 'Evelyn Thompson',
                 'Jack White',
                 'Dylan Garcia',
                 'Joseph White',
                 'Emma Garcia',
                 'Luke Taylor',
                 'Luke Williams',
                 'Jayden Scott',
                 'Sophia Evans',
                 'Lucas Clark',
                 'Jacob Martinez',
                 'Jack Edwards',
                 'Benjamin Baker',
                 'Amelia Green',
                 'William Davis',
                 'Lucas Smith',
                 'Mila Brown',
                 'Grayson Phillips',
                 'Benjamin Carter',
                 'James Martin',
                 'Oliver Perez',
                 'Daniel Evans',
                 'Elijah Thompson',
                 'Mila Lewis',
                 'Grayson Miller',
                 'Sophia Hall',
                 'Benjamin Jones',
                 'Jackson Baker',
                 'Olivia Moore',
                 'Harper Jackson',
                 'John Jones',
                 'Lucas Roberts',
                 'Amelia Robinson',
                 'Jayden Robinson',
                 'Lucas Thomas',
                 'Henry Mitchell',
                 'Avery Hill',
                 'Alexander Lopez',
                 'Ava Carter',
```

```
'Mia Miller',
'Carter Adams',
'William Martin',
'Mia Martinez',
'David Nelson',
'Evelyn Young',
'Daniel Robinson',
'Jackson King',
'Elizabeth Martinez',
'Ella Evans',
'Emma Brown',
'Andrew Thompson',
'Liam Gonzalez',
'Daniel Lewis',
'Emma Lee',
'Wyatt Phillips',
'Elijah Robinson',
'Charlotte Campbell',
'Ella Carter',
'Dylan Garcia',
'Oliver Parker',
'Matthew Martinez',
'James Jackson',
'Sophia Martin',
'Sebastian King',
'Alexander Anderson',
'James Hernandez',
'Noah Allen',
'Mia White',
'Carter Wright',
'Elizabeth White',
'Ava Clark',
'Ella Evans',
'Liam Collins',
'David Campbell',
'Mason Robinson',
'Aiden Walker',
'Abigail Jackson',
'Luke Wright',
'Harper Rodriguez',
'Matthew Lewis',
'Mason Nelson',
'Oliver Lee',
'Lucas Thomas',
'Sebastian Moore',
'Harper Lopez',
'Samuel Moore',
'Lucas Robinson',
'Mason Jackson',
'Logan Jones',
'Avery Jones',
'Jackson Carter',
'James Parker',
'Harper Taylor',
'Avery White',
'Emily Martinez',
'Michael Hernandez',
```

```
'James Allen',
'Benjamin King',
'Carter Adams',
'Carter Robinson',
'Ava Thomas',
'Luke Davis',
'Wyatt Lopez',
'Jackson Moore',
'Ella Evans',
'Oliver Davis',
'Daniel Gonzalez',
'Abigail Parker',
'Logan Wilson',
'Ella Lee',
'Luke Carter',
'Dylan Garcia',
'Alexander Campbell',
'Henry Walker',
'Evelyn Johnson',
'Lucas Taylor',
'Henry Hall',
'Emily Jones',
'Abigail Williams',
'Liam Collins',
'Jack Garcia',
'Sophia Edwards',
'Harper Moore',
'Luke Wright',
'Mila King',
'Henry Campbell',
'Emily Nelson',
'Avery Jones',
'Charlotte King',
'Harper Jackson',
'Elizabeth Martinez',
'Jack Moore',
'Logan Evans',
'William Hill',
'Olivia Lewis',
'Mila Thomas',
'Noah Johnson',
'Ava Young',
'Jayden Rodriguez',
'Ella Miller',
'Sophia Rodriguez',
'Dylan Lewis',
'Aiden Walker',
'John King',
'John Martinez',
'Jackson Clark',
'Michael Parker',
'Wyatt Scott',
'Benjamin Miller',
'Jackson King',
'Joseph Harris',
'Sebastian Edwards',
'Sebastian King',
```

```
        'Alexander Lewis',
        'Charlotte Perez',
        'Wyatt Garcia',
        'Emma Martin',
        'Elizabeth Baker',
        'Mila King',
        'Ella Hernandez',
        'Avery Roberts',
        'Ella Clark',
        'Samuel Adams',
        'Carter Thompson',
        'Samuel Wilson',
        'Benjamin Gonzalez',
        'Olivia Brown',
        'Grayson Miller',
        'Mia Davis',
        'Isabella Young',
        'Harper King',
        'Grayson Lewis',
        'Ethan Thomas',
        'Samuel Thompson',
        'Ethan Walker',
        'Mason King',
        'James Hernandez',
        'Aiden Clark',
        'Logan Garcia',
        'Isabella Mitchell',
        'Dylan Martin',
        'Grayson Edwards',
        'Emma Collins',
        'Matthew Miller',
        ...]
```

Question 2.4 Takeaways

My findings

Overall, we see that just with a few columns, we can identify roughly 85% of the individuals in the leaked dataset. In addition, the algorithm I used involved using our p_1 statistic to help us first identify all user with the same start zip code for a ride and home zip (this is useful because over 90% of our riders start their rides in the same zip code as their address) and used the Jaccard index (https://en.wikipedia.org/wiki/Jaccard_index (https://en.wikipedia.org/wiki/Jaccard_index)) to get similarity scores based on sex, month, and year. This algorithm was able to run through all 54K+ rides and make predictions (broke ties with randomness) for the user in less than 30 min., which is really unsettling. For those who create these datasets, I believe that a discussion that is extremely necessary is two-fold: 1. Do we really need to include certain bits of data (i.e. is month a crucial column in this dataset? In my opinion, it is not, and it actually led to users getting identified) and 2. Is the granularity of our data going to be an issue if the data is ever released.

For the already released data, we should immediately contact our customers and tell them about the leak. In addition, as a company, we should release a PR statement to our customers/the public about the leak and how we are going to increase our security in order to prevent this from happening again, and I would also speak with our data engineers and have them conduct analysis on which columns can be eliminated from our dataset (specifically looking at columns containing personal data that does not contribute much value to the business).

When we release future datsets, I think many things need to be considered.

1. What is purpose of this dataset? If its just for the public to see where users start and finish rides, why are we including personl data like sex, month, and year?
2. How large is our customer base? Since we are a stealth start-up, our customer base is likely very small, thus making individuals easier to identify (if this dataset was leaked by a larger company, i.e. Facebook, identifying users just on sex, month, and year would be impossible.
3. Is our data *truly* annonymized? Are we releasing columns that are proxies for other, deeply personal information?