Analysis

July 9, 2021

0.0.1 Imports

[1]:
```
# Import Modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
```

0.0.2 Load Data and Preprocessing

[2]:
```
# load data in dataframe
data = pd.read_csv('bq-results-20210629-101917-ms96hv1008ic.csv', dtype =␣
 ↪{"lock_postal_code": str})
```

[3]:
```
print('Shape: ' ,data.shape)
```

```
Shape:  (114852, 21)
```

[4]:
```
data.head()
```

[4]:

	action	key_id	operate_ms	timestamp	lock_id	…
0	locked	48fc13b5f3	4145.0	2021-05-26 15:30:07.127 UTC	8e5a08a742	
1	unlocked	48fc13b5f3	3848.0	2021-05-26 15:11:16.911 UTC	8e5a08a742	
2	unlocked	5f12e06341	3602.0	2021-05-27 14:51:41.206 UTC	878dc4b8af	
3	locked	5f12e06341	4605.0	2021-05-28 12:21:03.818 UTC	878dc4b8af	
4	locked	c6aab7e2b7	4965.0	2021-05-28 13:35:06.711 UTC	878dc4b8af	

	vendor_lock_name	vendor_lock_type	lock_postal_code	lock_country_iso	…
0	danalock	danalock	NaN	NaN	
1	danalock	danalock	NaN	NaN	
2	danalock	danalock	NaN	NaN	
3	danalock	danalock	NaN	NaN	
4	danalock	danalock	NaN	NaN	

	user_id	…	key_creator_id	key_creator_country_iso	device_id	…
0	b967f27244	…	NaN	NaN	1db3db1d4c	
1	b967f27244	…	NaN	NaN	1db3db1d4c	

```
2  23cd975f47  …     d902fbdbc1                    NO  790fcdf47e
3  23cd975f47  …     d902fbdbc1                    NO  790fcdf47e
4  d902fbdbc1  …            NaN                   NaN  f1884b7332

  device_platform device_model device_locale  device_app_version  \
0         android     SM-A515F         nb_NO  Unloc 4.6.9 (1036)
1         android     SM-A515F         nb_NO  Unloc 4.6.9 (1036)
2             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)
3             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)
4             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)

  lock_holder_id lock_holder_type lock_holder_country_iso
0     e183b94f37           office                      NO
1     e183b94f37           office                      NO
2            NaN              NaN                     NaN
3            NaN              NaN                     NaN
4            NaN              NaN                     NaN

[5 rows x 21 columns]
```

[5]:
```
data.isna().sum()
```

[5]:
```
action                      0
key_id                      0
operate_ms                1259
timestamp                   0
lock_id                     0
vendor_lock_name         2562
vendor_lock_type         2981
lock_postal_code        41664
lock_country_iso        41648
user_id                  1062
user_country_iso         1062
key_creator_id          75784
key_creator_country_iso 75784
device_id                1062
device_platform          1062
device_model             1062
device_locale            1062
device_app_version       1062
lock_holder_id          21723
lock_holder_type        21723
lock_holder_country_iso 21723
dtype: int64
```

data.dtypes

[6]:
```
# Convert UTC string timestamp to datetime
data['timestamp'] = pd.to_datetime(data['timestamp']).dt.strftime('%Y-%m-%d %H:
↪%M:%S')
data['timestamp'] = pd.to_datetime(data['timestamp'])
```

[7]:
```
data.head()
```

[7]:
```
     action      key_id  operate_ms           timestamp     lock_id  \
0    locked  48fc13b5f3      4145.0 2021-05-26 15:30:07  8e5a08a742
1  unlocked  48fc13b5f3      3848.0 2021-05-26 15:11:16  8e5a08a742
2  unlocked  5f12e06341      3602.0 2021-05-27 14:51:41  878dc4b8af
3    locked  5f12e06341      4605.0 2021-05-28 12:21:03  878dc4b8af
4    locked  c6aab7e2b7      4965.0 2021-05-28 13:35:06  878dc4b8af

  vendor_lock_name vendor_lock_type lock_postal_code lock_country_iso  \
0         danalock         danalock              NaN              NaN
1         danalock         danalock              NaN              NaN
2         danalock         danalock              NaN              NaN
3         danalock         danalock              NaN              NaN
4         danalock         danalock              NaN              NaN

      user_id  … key_creator_id key_creator_country_iso   device_id  \
0  b967f27244  …            NaN                     NaN  1db3db1d4c
1  b967f27244  …            NaN                     NaN  1db3db1d4c
2  23cd975f47  …     d902fbdbc1                      NO  790fcdf47e
3  23cd975f47  …     d902fbdbc1                      NO  790fcdf47e
4  d902fbdbc1  …            NaN                     NaN  f1884b7332

  device_platform device_model device_locale  device_app_version  \
0         android     SM-A515F         nb_NO  Unloc 4.6.9 (1036)
1         android     SM-A515F         nb_NO  Unloc 4.6.9 (1036)
2             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)
3             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)
4             iOS       iPhone         nb_NO   Unloc 4.6.4 (988)

  lock_holder_id lock_holder_type lock_holder_country_iso
0     e183b94f37           office                     NO
1     e183b94f37           office                     NO
2            NaN              NaN                    NaN
3            NaN              NaN                    NaN
4            NaN              NaN                    NaN

[5 rows x 21 columns]
```

0.0.3 Analysis and Visualization

[8]:
```
actions = data['action'].value_counts()
```

[9]:
```
actions
```

[9]:
```
unlocked      103432
locked         11224
viewedCode       196
Name: action, dtype: int64
```

[10]:
```
print('Unlocked Percentage:', actions[0]/len(data))
print('Locked Percentage:', actions[1]/len(data))
print('ViewedCode Percentage:', actions[2]/len(data))
```

```
Unlocked Percentage: 0.9005676871103682
Locked Percentage: 0.0977257688155191
ViewedCode Percentage: 0.0017065440741127712
```

[11]:
```
print('Total keys used:', len(data['key_id'].value_counts()))
```

```
Total keys used: 17079
```

[13]:
```
print('Total locks used:', len(data['lock_id'].value_counts()))
```

```
Total locks used: 2998
```

[14]:
```
# Group by Date and get count of records
count_by_dates = data.groupby([data['timestamp'].dt.date])['action'].size().
↪reset_index(name='Count')
```

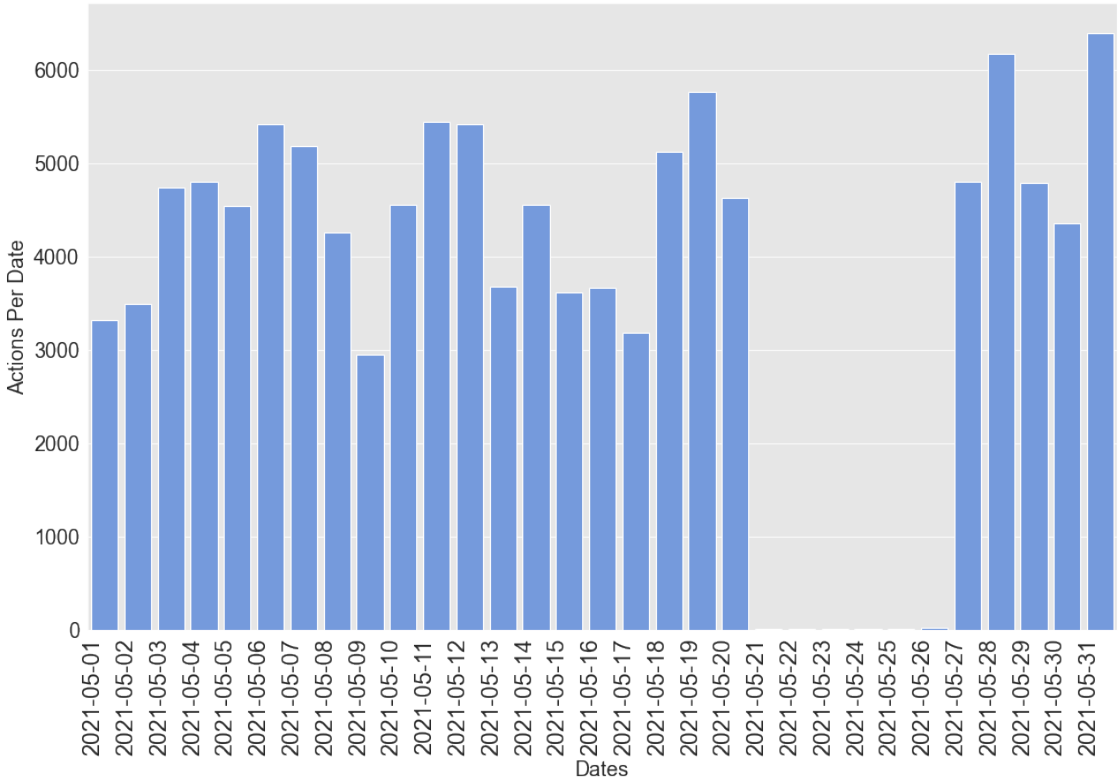
[15]:
```
count_by_dates
```

[15]:

	timestamp	Count
0	2021-05-01	3313
1	2021-05-02	3494
2	2021-05-03	4737
3	2021-05-04	4796
4	2021-05-05	4539
5	2021-05-06	5419
6	2021-05-07	5180
7	2021-05-08	4254
8	2021-05-09	2950
9	2021-05-10	4550
10	2021-05-11	5441
11	2021-05-12	5417
12	2021-05-13	3681
13	2021-05-14	4558

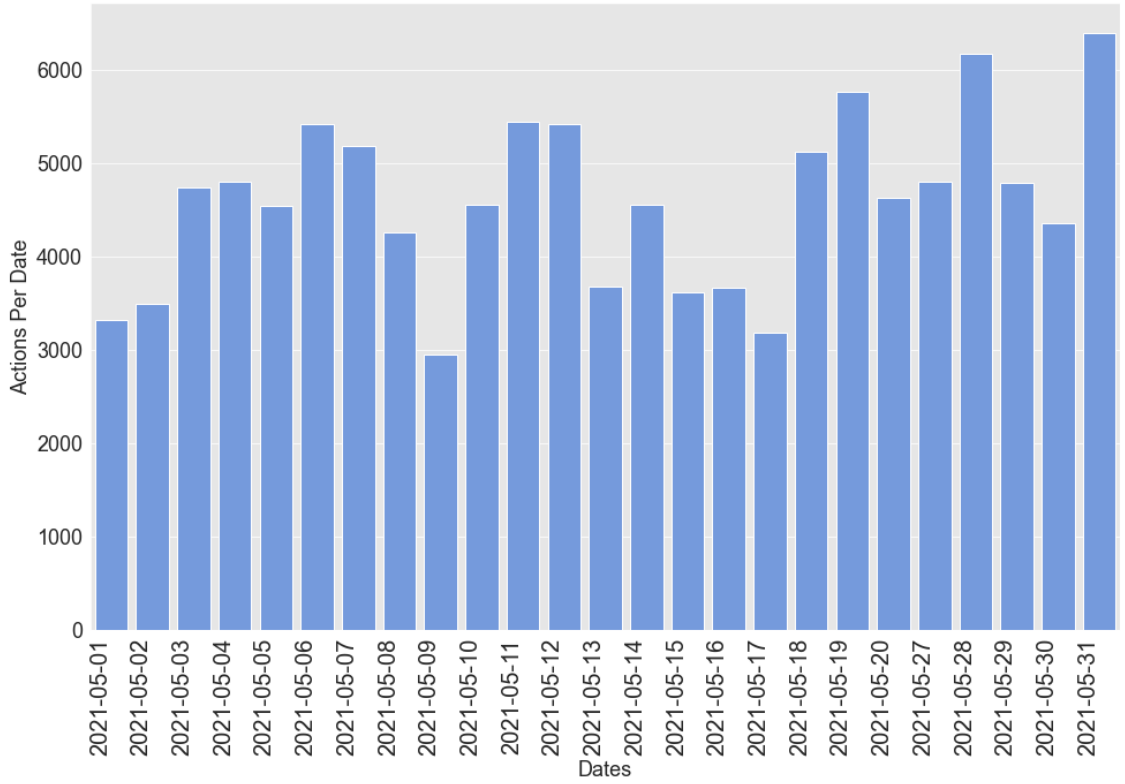
```
14  2021-05-15   3617
15  2021-05-16   3660
16  2021-05-17   3184
17  2021-05-18   5122
18  2021-05-19   5759
19  2021-05-20   4632
20  2021-05-21      3
21  2021-05-22      3
22  2021-05-23      1
23  2021-05-24      5
24  2021-05-25      9
25  2021-05-26     20
26  2021-05-27   4804
27  2021-05-28   6176
28  2021-05-29   4782
29  2021-05-30   4351
30  2021-05-31   6395
```

[16]:
```
# Plot Barplot
count_by_dates['timestamp'] = pd.to_datetime(count_by_dates['timestamp'])
plt.figure(figsize=(16,10))
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
all_dates = sns.barplot(x = count_by_dates['timestamp'].dt.date,␣
↪y=count_by_dates['Count'], data = count_by_dates, color="cornflowerblue")
plt.xticks(rotation=90,horizontalalignment='right',fontweight='light',fontsize␣
↪= 20)
plt.yticks(fontsize = 18)
plt.xlabel('Dates',fontsize = 18)
plt.ylabel('Actions Per Date',fontsize = 18)
plt.show()
all_dates.figure.savefig('All_Dates.png')
```



[17]:
```
# Drop Outliers
count_by_dates = count_by_dates.drop(count_by_dates[count_by_dates.Count < 30].
 ↪index)
```

[18]:
```
# Plot Barplot
plt.figure(figsize=(16,10))
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
all_dates_filtered = sns.barplot(x = count_by_dates['timestamp'].dt.date,␣
 ↪y=count_by_dates['Count'], data = count_by_dates, color="cornflowerblue")
plt.xticks(rotation=90,horizontalalignment='right',fontweight='light', fontsize␣
 ↪= 20)
plt.yticks(fontsize = 18)
plt.xlabel('Dates',fontsize = 18)
plt.ylabel('Actions Per Date',fontsize = 18)
plt.show()
all_dates_filtered.figure.savefig('Dates_Filtered.png')
```



[19]:
```
# Group by week days and get count of records
count_by_week_days = count_by_dates.groupby(count_by_dates['timestamp'].dt.
↪day_name()).sum()
```

[20]:
```
# Day Name with dates
count_by_dates['Days'] = count_by_dates['timestamp'].dt.day_name()
```

[21]:
```
count_by_dates
```

[21]:

	timestamp	Count	Days
0	2021-05-01	3313	Saturday
1	2021-05-02	3494	Sunday
2	2021-05-03	4737	Monday
3	2021-05-04	4796	Tuesday
4	2021-05-05	4539	Wednesday
5	2021-05-06	5419	Thursday
6	2021-05-07	5180	Friday
7	2021-05-08	4254	Saturday
8	2021-05-09	2950	Sunday
9	2021-05-10	4550	Monday
10	2021-05-11	5441	Tuesday
11	2021-05-12	5417	Wednesday

```
12 2021-05-13   3681   Thursday
13 2021-05-14   4558     Friday
14 2021-05-15   3617   Saturday
15 2021-05-16   3660     Sunday
16 2021-05-17   3184     Monday
17 2021-05-18   5122    Tuesday
18 2021-05-19   5759  Wednesday
19 2021-05-20   4632   Thursday
26 2021-05-27   4804   Thursday
27 2021-05-28   6176     Friday
28 2021-05-29   4782   Saturday
29 2021-05-30   4351     Sunday
30 2021-05-31   6395     Monday
```

[22]:
```
days_counts = count_by_dates.Days.value_counts()
```

[23]:
```
count_by_week_days['Number'] = [days_counts.loc[idx] for idx in␣
↪count_by_week_days.index]
```

[24]:
```
count_by_week_days['PerDay'] = count_by_week_days['Count']/
↪count_by_week_days['Number']
```

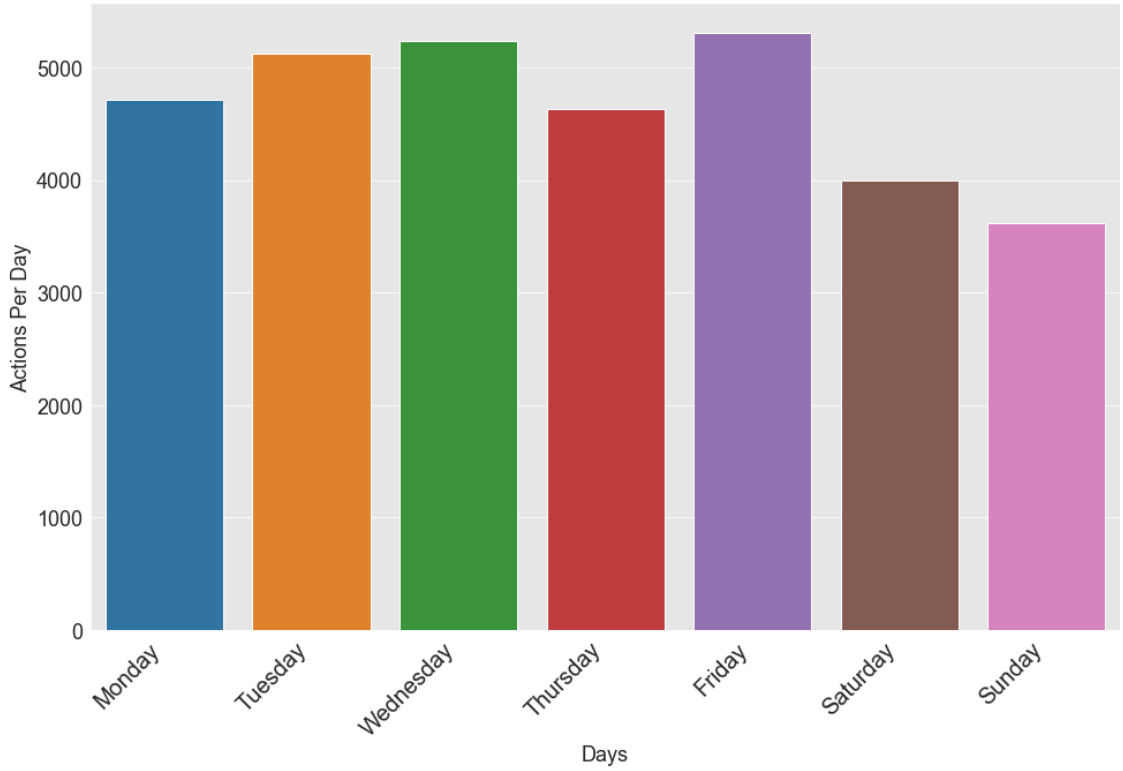
[25]:
```
count_by_week_days
```

[25]:
```
           Count  Number       PerDay
timestamp
Friday     15914       3  5304.666667
Monday     18866       4  4716.500000
Saturday   15966       4  3991.500000
Sunday     14455       4  3613.750000
Thursday   18536       4  4634.000000
Tuesday    15359       3  5119.666667
Wednesday  15715       3  5238.333333
```

[26]:
```
# Sort Weekdays
cats = [ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',␣
↪'Sunday']
count_by_week_days = count_by_week_days.reindex(cats)
```

[27]:
```
# Plot bar plot for weekdays
plt.figure(figsize=(16,10))
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
perDay_plot = sns.barplot(y='PerDay', x=count_by_week_days['Count'].index,␣
↪data=count_by_week_days)
plt.xticks(rotation=45,horizontalalignment='right',fontweight='light', fontsize␣
↪= 20)
plt.yticks(fontsize = 18)
```

```
plt.xlabel('Days',fontsize = 18)
plt.ylabel('Actions Per Day',fontsize = 18)
perDay_plot.figure.savefig('PerDay.png')
```



[28]:
```
# Remove Outliers from Data
data_filtered = data[(data['timestamp'].dt.date < datetime.date(2021,5,21)) |␣
↪(data['timestamp'].dt.date > datetime.date(2021,5,26))]
```

[29]:
```
# Group by hour
count_by_hour = data_filtered.groupby([data_filtered['timestamp'].dt.hour]).
↪size().reset_index(name='Count')
```

[30]:
```
count_by_hour
```

[30]:

	timestamp	Count
0	0	719
1	1	584
2	2	269
3	3	282
4	4	977
5	5	2752
6	6	5457

```
7            7   5036
8            8   5019
9            9   6525
10          10   7095
11          11   7405
12          12   8088
13          13   9308
14          14  10345
15          15   9661
16          16   8557
17          17   7159
18          18   6349
19          19   5081
20          20   3912
21          21   2262
22          22   1190
23          23    779
```

[31]:
```
# Convert to string and make time format of hour and minutes
count_by_hour['timestamp'] = count_by_hour['timestamp'].astype(str)
count_by_hour['timestamp'] = count_by_hour['timestamp'] + ":00"
```

[32]:
```
# Calculate Average per hour
count_by_hour['Count'] = count_by_hour['Count']/len(count_by_dates)
```

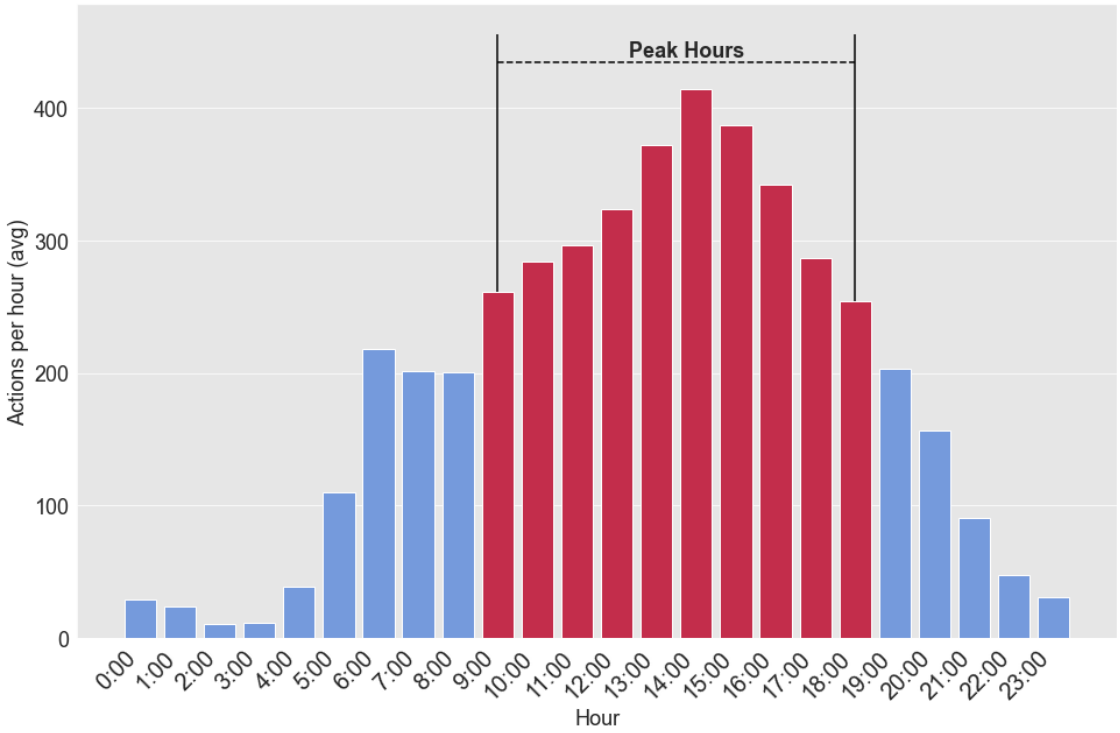
[33]:
```
count_by_hour
```

[33]:

	timestamp	Count
0	0:00	28.76
1	1:00	23.36
2	2:00	10.76
3	3:00	11.28
4	4:00	39.08
5	5:00	110.08
6	6:00	218.28
7	7:00	201.44
8	8:00	200.76
9	9:00	261.00
10	10:00	283.80
11	11:00	296.20
12	12:00	323.52
13	13:00	372.32
14	14:00	413.80
15	15:00	386.44
16	16:00	342.28
17	17:00	286.36
18	18:00	253.96

```
19      19:00  203.24
20      20:00  156.48
21      21:00   90.48
22      22:00   47.60
23      23:00   31.16
```

[34]:
```
# Avergae hourly Bar Plot
plt.figure(figsize=(16,10))
clrs = ['cornflowerblue' if (x < 250) else 'crimson' for x in␣
 ↪count_by_hour['Count']]
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
hours_plot = sns.barplot(x = count_by_hour['timestamp'],␣
 ↪y=count_by_hour['Count'], data = count_by_hour, palette=clrs)
plt.xticks(rotation=45,horizontalalignment='right',fontweight='light', fontsize␣
 ↪= 20)
plt.yticks(fontsize = 18)
plt.xlabel('Hour',fontsize = 18)
plt.ylabel('Actions per hour (avg)',fontsize = 18)
plt.vlines(9,(count_by_hour.Count).iloc[9], (count_by_hour.Count).max()*1.1,␣
 ↪color='black')
plt.vlines(18,(count_by_hour.Count).iloc[18], (count_by_hour.Count).max()*1.
 ↪1,color='black')
plt.hlines(count_by_hour.Count.max()*1.05,9,␣
 ↪18,color='black',linestyle='dashed')
plt.text(12.3, count_by_hour.Count.max()*1.06,'Peak Hours', fontsize=18,␣
 ↪fontweight = 'bold')
plt.show()
hours_plot.figure.savefig('PerHour.jpg')
```



[35]:
```
max_action_hour = count_by_hour[count_by_hour['Count']==count_by_hour['Count'].
↪max()]
min_action_hour = count_by_hour[count_by_hour['Count']==count_by_hour['Count'].
↪min()]
```

[36]:
```
print('Max Actions: ',int(max_action_hour['Count'].max()), ' at hour␣
↪',max_action_hour['timestamp'].max())
print('Min Actions: ',int(min_action_hour['Count'].min()), ' at hour␣
↪',min_action_hour['timestamp'].min())
```

```
Max Actions:  413  at hour  14:00
Min Actions:  10  at hour  2:00
```

[37]:
```
vendor_lock_type_counts = data['vendor_lock_type'].value_counts()
```

[38]:
```
vendor_lock_type_counts
```

[38]:
```
universal             80047
danalock              29372
zw_340_5_1             1056
XS                      785
masterlock              399
codelockEasyAccess       91
```

```
codelockYaleDoorman       55
generic                   37
codelockIDLock            13
electric-lock              6
zw_560_3_1                 4
zw_270_9_1                 4
zw_883_3_1                 2
Name: vendor_lock_type, dtype: int64
```

[39]:
```
print('Universal Percentage: ',vendor_lock_type_counts[0]/
 ↪vendor_lock_type_counts.sum())
print('danalock Percentage: ',vendor_lock_type_counts[1]/
 ↪vendor_lock_type_counts.sum())
```

```
Universal Percentage:  0.7155294937919568
danalock Percentage:  0.26255240410830333
```

[40]:
```
vendor_lock_type_counts/vendor_lock_type_counts.sum()
```

[40]:
```
universal              0.715529
danalock               0.262552
zw_340_5_1             0.009439
XS                     0.007017
masterlock             0.003567
codelockEasyAccess     0.000813
codelockYaleDoorman    0.000492
generic                0.000331
codelockIDLock         0.000116
electric-lock          0.000054
zw_560_3_1             0.000036
zw_270_9_1             0.000036
zw_883_3_1             0.000018
Name: vendor_lock_type, dtype: float64
```

[41]:
```
# dataframe for lock types
df_lock_types = pd.DataFrame(vendor_lock_type_counts)
```

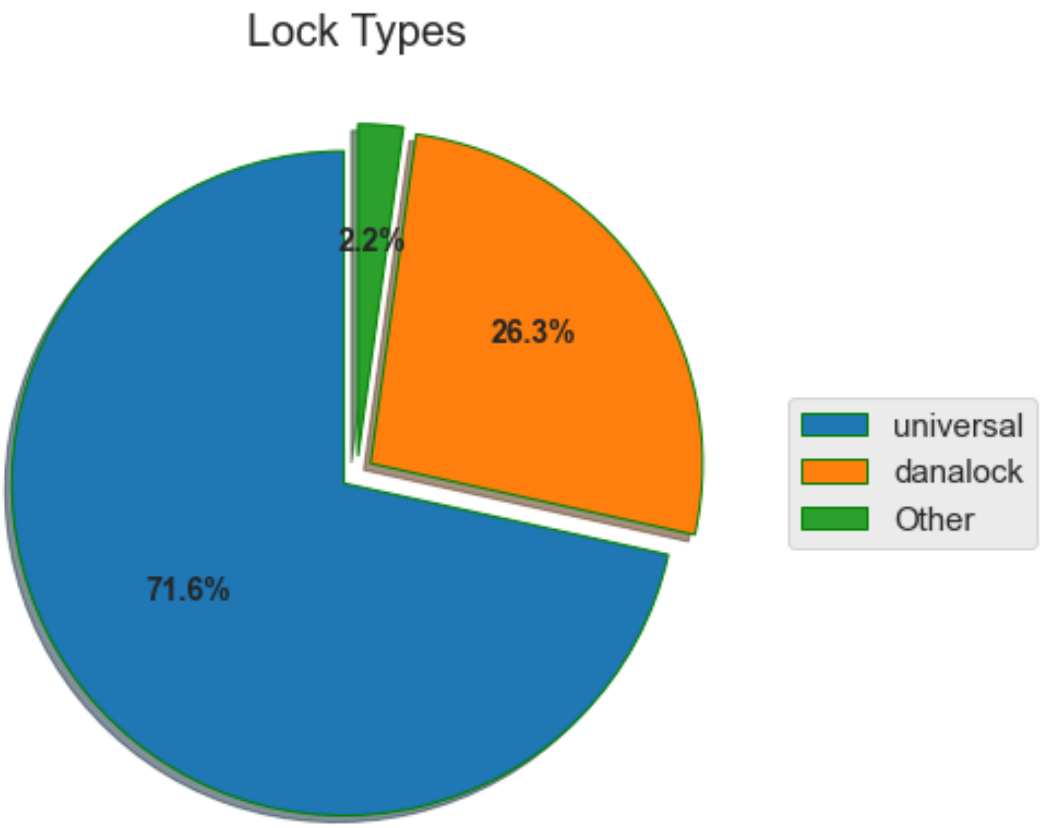
[42]:
```
# Compile lock types with less percenatge to one entry for beter visualization
df_lock_types.reset_index(inplace=True)
df_lock_types = df_lock_types.rename(columns={"index": "Lock_Type",␣
 ↪"vendor_lock_type": "Count"})
df_lock_types.loc[(df_lock_types.Count < 2000),'Lock_Type']='Other'
df_lock_types = df_lock_types.set_index('Lock_Type',drop=True)
df_lock_types = df_lock_types.groupby(df_lock_types.index).sum()
df_lock_types=df_lock_types.sort_values('Count', ascending=False)
```

[43]:
```
df_lock_types
```

[43]:
```
             Count
Lock_Type
universal    80047
danalock     29372
Other         2452
```

[44]:
```
# PieChart for Lock Types
wp = { 'linewidth' : 1, 'edgecolor' : "green" }

# Creating explode data
explode = (0.05, 0.05, 0.05)

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    print(pct)
    return "{:.1f}%\n".format(pct, absolute)




# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(df_lock_types['Count'],
                                                                autopct =␣
 ↪lambda pct: func(pct, df_lock_types['Count']),
                                                                shadow = True,
                                                                startangle = 90,
                                    explode= explode,
                                                                wedgeprops = wp,
                                                                )


ax.legend(wedges, df_lock_types.index,
                 loc ="center left",
          fontsize = 15,
                 bbox_to_anchor =(1, 0, 0.5, 1))

plt.setp(autotexts, size = 14, weight ="bold")
ax.set_title("Lock Types",size = 20)

# show plot
plt.show()
fig.savefig('lock_types.png')
```

```
71.55295014381409
26.2552410364151
2.1918101236224174
```



[45]:
```
print('Most Used Lock Type: ',vendor_lock_type_counts.index[0], ' with count: '␣
↪,vendor_lock_type_counts[0])
print('Least Used Lock Type: ',vendor_lock_type_counts.
↪index[len(vendor_lock_type_counts)-1], ' with count: '␣
↪,vendor_lock_type_counts[len(vendor_lock_type_counts)-1])
print('Known Lock Types:', vendor_lock_type_counts.sum())
print('Unknown Lock Types:', data['vendor_lock_type'].isna().sum())
```

```
Most Used Lock Type:  universal  with count:  80047
Least Used Lock Type:  zw_883_3_1  with count:  2
Known Lock Types: 111871
Unknown Lock Types: 2981
```

[46]:
```
# Lock Country data
data['lock_country_iso'].value_counts()
```

[46]:
```
NO    72508
SE      696
Name: lock_country_iso, dtype: int64
```

[47]:
```
# Lock postal code data
data['lock_postal_code'].value_counts()
```

[47]:
```
0575    3053
0478    2884
0661    2866
0175    2564
0196    2135
        …
4521       1
5610       1
3612       1
0266       1
2953       1
Name: lock_postal_code, Length: 251, dtype: int64
```

[48]:
```
print('Unique Users: ', data['user_id'].nunique())
```

```
Unique Users:  8616
```

[49]:
```
# User Counry data
country_data = data['user_country_iso'].value_counts()
```

[50]:
```
# Percentage of users by country
country_data/ country_data.sum()
```

[50]:
```
NO    0.979119
SE    0.011908
DK    0.002707
FR    0.001485
GB    0.001090
IT    0.001028
FI    0.000571
DE    0.000404
IS    0.000378
US    0.000360
CZ    0.000281
BE    0.000255
ES    0.000211
PL    0.000088
NL    0.000062
CH    0.000026
EE    0.000026
Name: user_country_iso, dtype: float64
```

[51]:
```
print('Total Countries: ', data['user_country_iso'].nunique())
```

```
Total Countries:  17
```

[52]:
```
device_type_count = data['device_platform'].value_counts()
print(device_type_count)
```

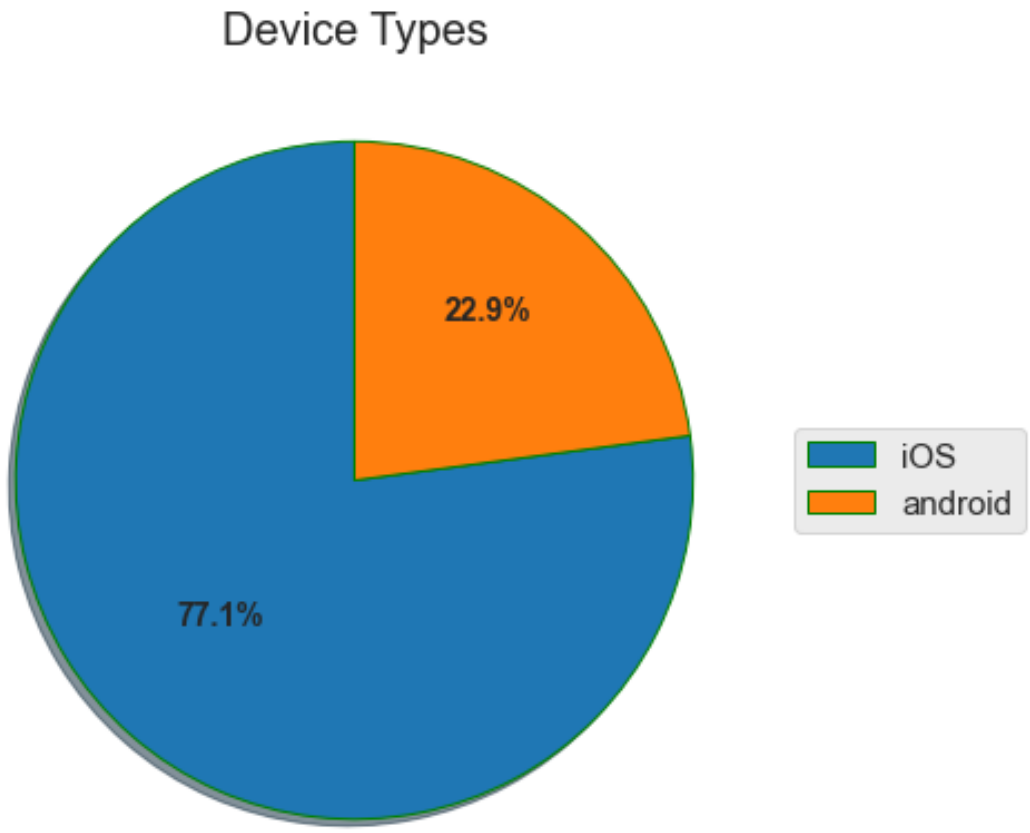
```
iOS        87740
android    26050
Name: device_platform, dtype: int64
```

[53]:
```
print('IOS Percentage: ', device_type_count[0]/data['device_platform'].
 ↪value_counts().sum())
```

```
IOS Percentage:  0.7710695140170489
```

[54]:
```
# Pie Chart of device type
wp = { 'linewidth' : 1, 'edgecolor' : "green" }



# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    print(pct)
    return "{:.1f}%\n".format(pct, absolute)




# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(device_type_count,
                                  autopct =␣
 ↪lambda pct: func(pct, device_type_count),
                                  shadow = True,
                                  startangle = 90,
                                  wedgeprops = wp,
                                  )

ax.legend(wedges, device_type_count.index,
                loc ="center left",
        fontsize = 15,
                bbox_to_anchor =(1, 0, 0.5, 1))

plt.setp(autotexts, size = 14, weight ="bold")
ax.set_title("Device Types",size = 20)

# show plot
plt.show()
fig.savefig('device_type.png')
```

```
77.10695266723633
22.89304882287979
```



[55]:
```
# App Versions
data['device_app_version'].value_counts()
```

[55]:
```
Unloc 4.6.4 (988)     45081
Unloc 4.6.2 (950)     25139
Unloc 4.6.3 (972)     22202
Unloc 4.6.3 (970)      6786
Unloc 4.6.5 (994)      6456
Unloc 4.5.9 (904)      2879
Unloc 4.5.10 (906)     2782
Unloc 4.6.0 (932)      1343
Unloc 4.6.1 (938)       773
Unloc 4.6.4 (984)       101
Unloc 4.6.6 (996)        68
Unloc 4.6.5 (992)        54
Unloc 4.6.4 (982)        28
```

```
Unloc 4.6.4 (986)        20
Unloc 4.6.6 (998)        13
Unloc 4.6.3 (962)        13
Unloc 4.6.5 (990)        13
Unloc 4.6.4 (980)        12
Unloc 4.6.2 (948)         9
Unloc 4.6.7 (1010)        7
Unloc 4.6.4 (976)         5
Unloc 4.3.5 (370)         3
Unloc 4.6.9 (1036)        2
Unloc 4.5.8 (900)         1
Name: device_app_version, dtype: int64
```

[57]:
```
# Lock Holder Types
lock_holder_type = data['lock_holder_type'].value_counts()
print(lock_holder_type)
```

```
cooperative        62822
lockholder         14458
propertymanager     7125
office              5610
coworking           3114
Name: lock_holder_type, dtype: int64
```

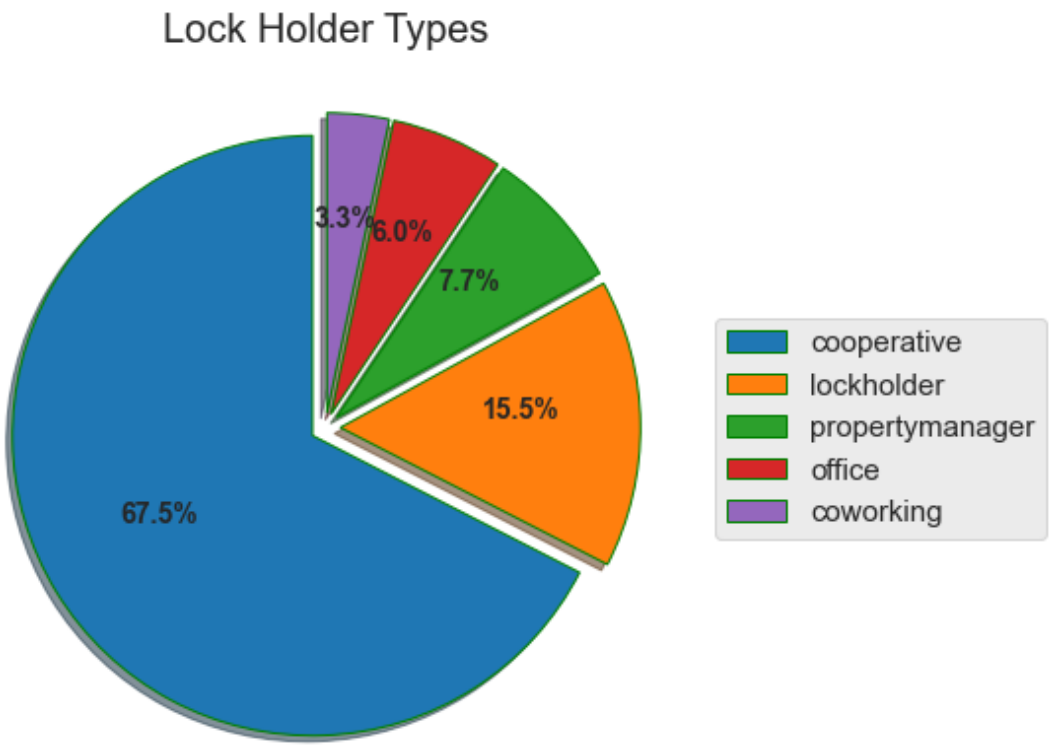
[58]:
```
lock_holder_type/lock_holder_type.sum()
```

[58]:
```
cooperative        0.674570
lockholder         0.155247
propertymanager    0.076507
office             0.060239
coworking          0.033437
Name: lock_holder_type, dtype: float64
```

[59]:
```
# Pie Char Lock holder types
wp = { 'linewidth' : 1, 'edgecolor' : "green" }

# Creating explode data
explode = (0.05, 0.05, 0.05, 0.05, 0.05)

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    print(pct)
    return "{:.1f}%\n".format(pct, absolute)
```

```
# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(lock_holder_type,
                                                                    autopct =␣
 ↪lambda pct: func(pct, lock_holder_type),
                                                                    shadow = True,
                                                                    startangle = 90,
                                    explode= explode,
                                                                    wedgeprops = wp,
                                                                    )

ax.legend(wedges, lock_holder_type.index,
                 loc ="center left",
         fontsize = 15,
                 bbox_to_anchor =(1, 0, 0.5, 1))

plt.setp(autotexts, size = 14, weight ="bold")
ax.set_title("Lock Holder Types",size = 20)

# show plot
plt.show()
fig.savefig('lockholdertypes_type.png')
```

```
67.45696663856506
15.52470177412033
7.650677859783173
6.02390244603157
3.3437490463256836
```



[60]:
```
data['device_locale'].value_counts()
```

[60]:
```
nb_NO    86735
en_NO    10148
en_GB     3745
en_US     2635
sv_SE     1638
         …
bs_NO        1
nb_AT        1
en_EE        1
it_GB        1
nb_PT        1
Name: device_locale, Length: 131, dtype: int64
```