Komodaa EDA

November 27, 2020

0.1 Import Necessary libraries

[1]:
```
# Data manipulation
import pandas as pd
import numpy as np

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.style as style

# Network analysis
import networkx as nx
from networkx.algorithms import community #This part of networkx, for community␣
↪detection, needs to be imported separately.

# System
from operator import itemgetter

pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

0.2 User Tabel

To Do List

1. Gender Distribution
2. Users Distribution Based On Cities
3. Cites Registration Growth During Period Given
4. Monthly Registration
5. Daily Registration

Reading data

[2]:
```
users_df = pd.read_csv('komodaa/users.csv')
```

Get familiar with data

[3]:
```
users_df.head(5)
```

[3]:
```
                                   uuid                                 city_uuid  \
0  06abe0e5-c408-4a33-9d81-5bfacc891f15  22972566-8739-40ca-bf19-a270d320dc83
1  33d7f40c-ab30-4517-881c-24d61acc62b8  22972566-8739-40ca-bf19-a270d320dc83
2  392e5ac0-fd87-4ea0-b9e4-bfffe077cca4  22972566-8739-40ca-bf19-a270d320dc83
3  16848d64-b555-4182-a5b4-c0ae1e850e01  22972566-8739-40ca-bf19-a270d320dc83
4  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83

   gender                 date
0       1  2017-10-10 00:14:33
1       1  2017-10-10 00:14:33
2       1  2017-10-10 00:14:33
3       1  2017-10-10 00:14:33
4       1  2017-10-10 00:14:33
```

[4]:
```
# check columns type
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   uuid       6000 non-null   object
 1   city_uuid  6000 non-null   object
 2   gender     6000 non-null   int64 
 3   date       6000 non-null   object
dtypes: int64(1), object(3)
memory usage: 187.6+ KB
```

[5]:
```
print('Number Of Users: ', users_df.shape[0] )
```

```
Number Of Users:  6000
```

[6]:
```
# Check for Dublicate
users_df.duplicated().value_counts()
```

[6]:
```
False    6000
dtype: int64
```

There are no Duplicate value in DataFrame

[7]:
```
# Chech for Null Value
missing_percentage = users_df.isnull().sum() / users_df.shape[0] * 100
missing_percentage
```

[7]:
```
uuid         0.000
city_uuid    0.000
gender       0.000
date         0.000
```

```
dtype: float64
```

There are no Null value in columns of DataFrame

0.2.1 Exploring Data

How many days?

[8]:
```
# convert data from object type to pandas datatime
users_df["date"] = pd.to_datetime(users_df.date, cache=True)

print("Start Of The Period : " , users_df.date.min())
print("End Of The Period   : " , users_df.date.max() )
print("Time Span           : " , users_df.date.max() - users_df.date.min())
```

```
Start Of The Period :  2017-10-10 00:14:33
End Of The Period   :  2018-10-31 20:44:33
Time Span           :  386 days 20:30:00
```

Gender Distribution

[9]:
```
'''
Gender Distribution
Gender:
        Male   ---> 1
        Female ---> 0
'''
users_df['gender'].value_counts()
```

[9]:
```
1    6000
Name: gender, dtype: int64
```

Intersting! **All** the Users are **Male**!

So we cannot do anything special with this column.

Users Distribution Based On Cities Let's find out which cities have the most user in our dataset.

[10]:
```
# Users distirbution based on cities
cities = pd.DataFrame(users_df['city_uuid'].value_counts())
cities.reset_index(inplace = True)
cities.rename(columns = {'index' : 'city_id', 'city_uuid' : 'Num_users'},␣
↪inplace = True)
cities.head(20)
```

[10]:

	city_id	Num_users
0	d29f1e68-56ce-476f-b09b-c960a4f5674f	1769
1	22972566-8739-40ca-bf19-a270d320dc83	1183
2	308873a5-249c-4bd3-b721-9e5aa54cd0ce	319
3	cb735029-1b57-4f97-99f1-407a3e3dd074	305

```
4   b4716b80-6d6b-4bdf-82f3-8a735760806c        269
5   db04bfc4-9f26-4866-83af-85eb70bb8779        117
6   c77c98ad-15be-4ce8-8549-b126ae0105ae        111
7   060661e0-4799-49ab-a841-57684f48b473        110
8   b741806d-d2ce-4e07-8623-bc6cf4038303         66
9   1b1f27b2-40da-4fa7-bec4-697dca294770         64
10  821a043c-a8c3-45f1-8a18-d12fa7e2f34b         58
11  236488ce-c329-474b-a4f4-3943ea54dca7         55
12  7a458fd8-cbd8-4e05-9856-943c381df966         54
13  1c3b84cd-8d28-4385-8e99-002c4c6761cc         51
14  e86a45c1-3443-45ea-8cb6-d3d472ec0cbc         51
15  4fb5e2f6-aad3-4f2c-bea8-1debad63deb3         50
16  0d55bacb-cb27-4936-8682-3daf34552c37         50
17  907bef2e-4846-44a8-bea1-c5e77c3a328d         48
18  4e236d2d-18de-439e-bfda-2638c9767a54         46
19  5de00abb-8d60-46f0-841b-e607ec29af09         42
```

[11]:
```
# check for number of cites
print('We have {} different cities in our Dataset! '.format(cities.shape[0]))
print('City Wirh the ID " {} " has the most number of users with "{}". '.
 ↪format(cities.iloc[0]['city_id'], cities.iloc[0]['Num_users']))
```

```
We have 264 different cities in our Dataset!
City Wirh the ID " d29f1e68-56ce-476f-b09b-c960a4f5674f " has the most number of
users with "1769".
```

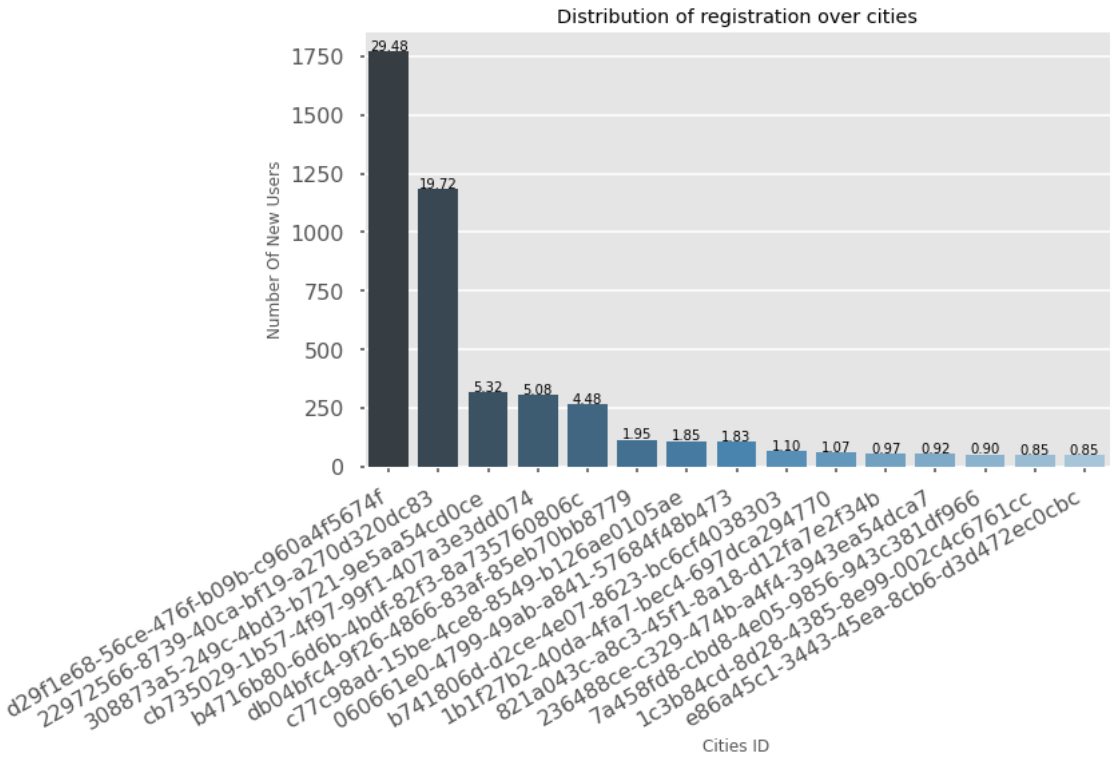
[12]:
```
print('We have {} cities with more than 10 users.'.
 ↪format(len(cities[cities['Num_users']>10])))
print('We have {} cities with less than or equal 10 users.'.
 ↪format(len(cities[cities['Num_users']<=10])))
print('We have {} cities with exactly one user.'.
 ↪format(len(cities[cities['Num_users']==1])))
```

```
We have 52 cities with more than 10 users.
We have 212 cities with less than or equal 10 users.
We have 96 cities with exactly one user.
```

Now let's visualize our data to get more sense of this numbers.

[13]:
```
style.use('seaborn-poster') #sets the size of the charts
style.use('ggplot')


fig= plt.figure(figsize=(10,6)) #figsize

ax = sns.barplot(x="city_id", y="Num_users", orient= '90', palette=("Blues_d"),␣
 ↪data=cities.head(15))
```

```
ax.set_xticklabels(ax.get_xticklabels(),rotation=30,␣
 ↪horizontalalignment='right')

# Add title and axis names
plt.title('Distribution Of Registration Over Cities')
plt.xlabel('Cities ID', fontsize='12')
plt.ylabel('Number Of New Users')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format((height/users_df.shape[0])*100),
            ha="center")

# Save graphic
plt.savefig('Images/Distribution of registration over 15 cities.
 ↪png',bbox_inches = 'tight')
```



It can be seen that almost **30%** of our users come from **single** cities. Furthermore, almost a **half** of our users comes from just **two** cities. Then we have **3** cities with the sum of **15%**, each one almost 5%. The rest cities, are less than **2%** each.

User Registration Growth by cities During Period Given We other thing that could be interesting to investigate is to find out about user registration growth during this period based on region and cities.

[14]:
```
# Make a new dataframe based on date(monthly), cities and number of new users
city_month_count = pd.DataFrame(users_df.groupby([users_df["date"].dt.
 ↪to_period('M'), users_df.city_uuid])['uuid'].count())
```

[15]:
```
city_month_count.shape
```

[15]:
```
(494, 1)
```

[16]:
```
city_month_count
```

[16]:

		uuid
date	city_uuid	
2017-10	22972566-8739-40ca-bf19-a270d320dc83	184
	22c4976f-4e01-45b9-b05d-0c854e7bd0a0	1
	308873a5-249c-4bd3-b721-9e5aa54cd0ce	5
	4f12d9d2-01ed-4b26-a2f0-b6dc4ae8f398	1
	5796c72a-af37-4ce0-bcfe-9f481ce7ecbe	1
…	…	…
2018-10	f8599cf8-1268-43d4-9fd9-3df9cdf5963d	1
	f8d0ef97-9f0b-411b-aa9a-69b73a5272e6	8
	f9bedfa1-deaf-41cd-8c06-f6e010a3b487	8
	fb728ded-0321-4a80-b0f5-6ff0ca16bf74	1
	fca5738c-0401-48f1-b446-408fce2a5e98	1

[494 rows x 1 columns]

There are almost 500 cities. That's alot of records! For visualization, It is better to be more specific.

For example, Here I will fillter out cities with less than 10 users per month.

[17]:
```
city_month_count = city_month_count[city_month_count['uuid']>=10]
```

[18]:
```
city_month_count.shape
```

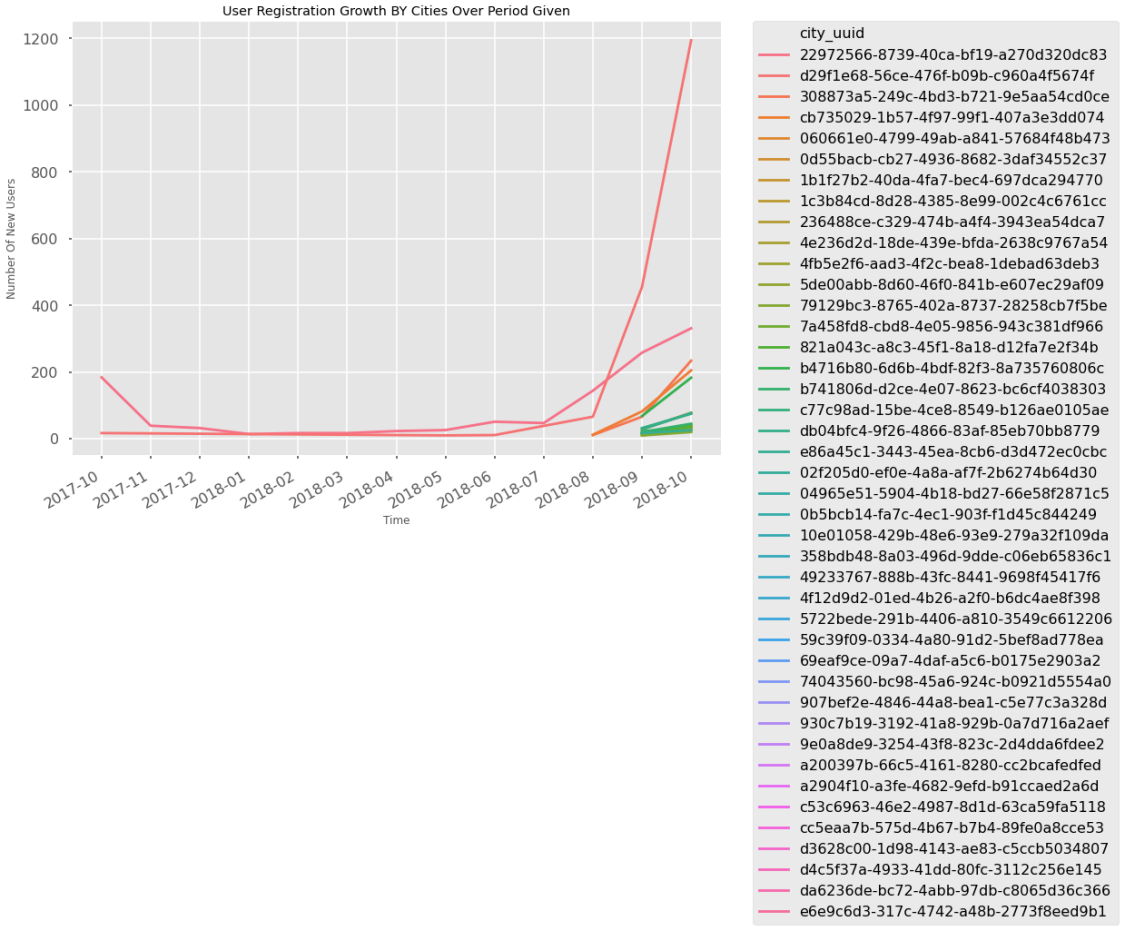
[18]:
```
(79, 1)
```

Still too much! However, it now makes muchm more sense.

[19]:
```
# Reset index
city_month_count.reset_index(inplace = True)
```

[20]:
```
# save dates for X axis lable
li_date = city_month_count.date.unique()
```

[21]:
```
plot = sns.lineplot(x=city_month_count['date'].astype(str), y='uuid',␣
↪hue='city_uuid', data=city_month_count)


plot.set_xticklabels(li_date,rotation=30, horizontalalignment='right')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)                ␣
↪     # move legend outside the chart

# Add title and axis names
plt.title('User Registration Growth By Cities Over Period Given')
plt.xlabel('Time', fontsize='12')
plt.ylabel('Number Of New Users')


plt.savefig('Images/User Registration Growth BY Cities Over Period Given.
↪png',bbox_inches = 'tight')
```



To be honest, because of these IDs I cannot suggest anything. However, for example, if we consider

two red line as Shiraz and tehran, It is obvious that the number of user registration increase during period shown. Likewise, for other cities we can see an increase in figures from August 2018.

Exploring Users Registration based on Month, Day and Hour Now, We can visualize user regestration Monthly, Daily and hourly in order to develop a sense about when user prefer to sign up in our system.

[22]:
```
# Check for date type
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   uuid       6000 non-null   object
 1   city_uuid  6000 non-null   object
 2   gender     6000 non-null   int64
 3   date       6000 non-null   datetime64[ns]
dtypes: datetime64[ns](1), int64(1), object(2)
memory usage: 187.6+ KB
```

Month I begin with monthly registration visualization

[23]:
```
# Create a column containing the month
users_df['month'] = pd.to_datetime(users_df['date']).dt.to_period('M')
months = users_df['month'].sort_values()

month = pd.DataFrame(users_df.month.value_counts())
month.reset_index(inplace = True)
month.rename(columns = {'index' : 'Month','month' : 'Count'}, inplace = True)
month.sort_values(by=['Month'], inplace = True, ignore_index=True)
month.head(20)
```

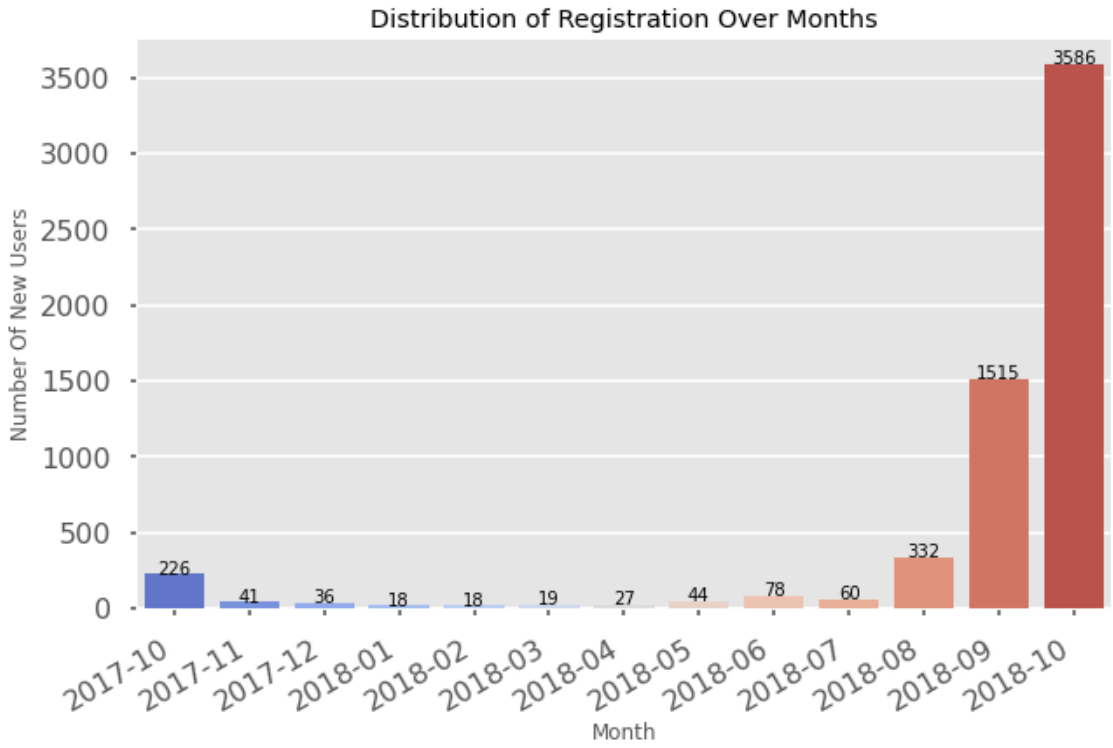
[23]:

	Month	Count
0	2017-10	226
1	2017-11	41
2	2017-12	36
3	2018-01	18
4	2018-02	18
5	2018-03	19
6	2018-04	27
7	2018-05	44
8	2018-06	78
9	2018-07	60
10	2018-08	332
11	2018-09	1515
12	2018-10	3586

We have **226** new users in the frist **month(2017-10)**. Then it can be seen a drop in number of new users registration for **5** month.(There should be a reason for that) and on month 6, again the number of new user registration increased steadily. On **2018-09** the figure for new users registration increased by **1200**, maybe because of some Online marketing campaign or some incentives for new users! We have a considerable figure for the last month new user of **3586**.

Now, I illustrate our data to make it more readable.

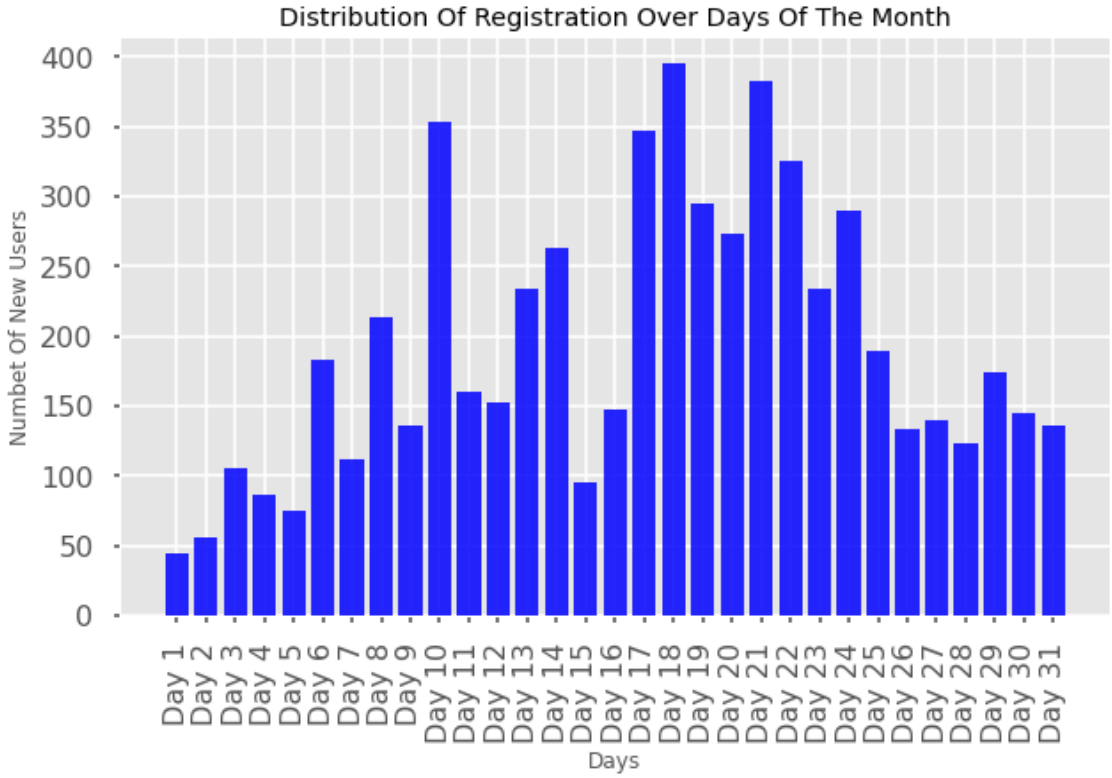
[24]:
```
# Monthly distribution count
style.use('seaborn-poster') #sets the size of the charts
style.use('ggplot')

fig= plt.figure(figsize=(10,6))
ax = sns.barplot(x="Month", y="Count", orient= '90',  palette=("coolwarm"),␣
 ↪data=month)
ax.set_xticklabels(ax.get_xticklabels(),rotation=30,␣
 ↪horizontalalignment='right')

# Add title and axis names
plt.title('Distribution of Registration Over Months')
plt.xlabel('Month', fontsize='12')
plt.ylabel('Number Of New Users')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

# Show graphic
plt.savefig('Images/Distribution of Registration Over Months.png',bbox_inches =␣
 ↪'tight')
```



Day I continue with Daily Registration

[25]:
```
# Daily distribution count
daily_count = users_df.groupby(users_df["date"].dt.day, as_index = False).
 ↪count()

fig= plt.figure(figsize=(10,6))

# Choose the names of the bars
bars = []
for i in range(1,32):
    bars.append('Day '+str(i))

y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, daily_count.date, color = 'Blue', alpha = 0.85)

# Create names on the x-axis
plt.xticks(y_pos, bars, rotation=90)
plt.yticks()

# Add title and axis names
```

```
plt.title('Distribution Of Registration Over Days Of The Month')
plt.xlabel('Days', fontsize='12')
plt.ylabel('Numbet Of New Users')


# Show graphic
plt.savefig('Images/Distribution of new Users Registration over days.
 ↪png',bbox_inches = 'tight')
plt.show()
```



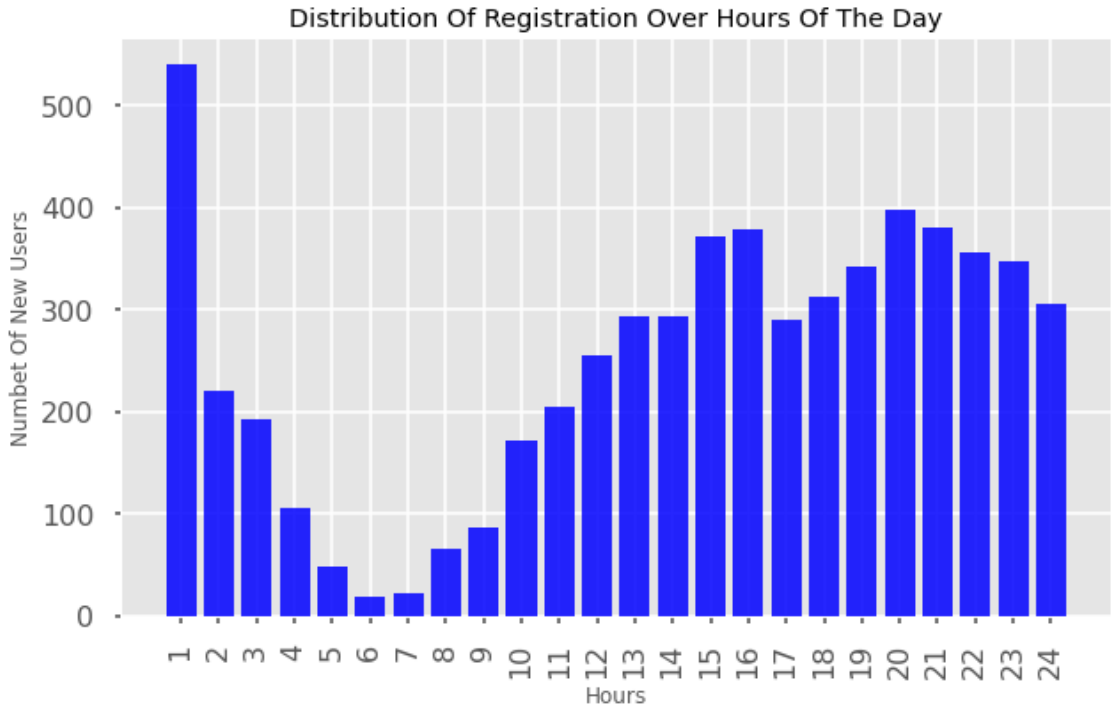
From the chart, It seems that people tend to register in third week of a month!

I cannot say why does this happen and it needs alot more data and investigation.

And continue with hourly registration.

[26]:
```
# Hourly distribution count
hourly_count = users_df.groupby(users_df["date"].dt.hour, as_index = False).
 ↪count()

fig= plt.figure(figsize=(10,6))

# Choose the names of the bars
```

```
bars = []
for i in range(1,25):
    bars.append(str(i))

y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, hourly_count.date,color = 'Blue', alpha = 0.85)

# Create names on the x-axis
plt.xticks(y_pos, bars, rotation=90)
plt.yticks()

# Add title and axis names
plt.title('Distribution Of Registration Over Hours Of The Day')
plt.xlabel('Hours', fontsize='12')
plt.ylabel('Numbet Of New Users')


# Show graphic
plt.savefig('Images/Distribution of Registration over hours.png',bbox_inches =␣
 ↪'tight')
plt.show()
```



It is not a surprise that most of users signed up at night and we have the least user registration during morning.

0.3 Follows DataFrame

To Do List

1. Most Active Users
2. Users Network (Directed Network)
3. Community Detection
4. User's Activites over period given

Read Data

[27]:
```
follows_df = pd.read_csv('komodaa/follows.csv')
follows_df.head()
```

[27]:
```
                              user_uuid                              follow_uuid  \
0  7effce15-12ba-469c-b4b4-b529425488db  372fe527-470c-48a3-a95e-fe32cc4bcc15
1  f262a138-9585-4570-9d2d-6815c4d8b6a6  ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd
2  18566d57-99ea-4f5d-aa23-7769f1b0dcc2  2098ef7a-9215-4d2e-8e95-0ab7e745392f
3  958029e7-6897-4ae2-be6c-515730568371  498439f1-52ed-4e3d-b07a-fe77061837b5
4  4afbf6d4-b841-44b1-8f04-eba6e6b102e3  498439f1-52ed-4e3d-b07a-fe77061837b5

                  date
0  2018-10-05 18:47:56
1  2020-06-29 14:57:37
2  2019-09-02 08:15:12
3  2018-09-05 05:02:57
4  2018-09-06 21:59:08
```

Get familiar with data

[28]:
```
follows_df.shape
```

[28]:
```
(5886, 3)
```

[29]:
```
# Check for Dublicate
follows_df.duplicated().value_counts()
```

[29]:
```
False    5886
dtype: int64
```

[30]:
```
# Chech for Null Value
missing_percentage = follows_df.isnull().sum() / follows_df.shape[0] * 100
missing_percentage
```

[30]:
```
user_uuid     0.000
follow_uuid   0.000
date          0.000
dtype: float64
```

[31]:
```
print('We have {} distinct followers in our dataset.'.
 ↪format(follows_df['user_uuid'].nunique()))
print('We have {} distinct followees in our dataset.'.
 ↪format(follows_df['follow_uuid'].nunique()))
```

```
We have 1454 distinct followers in our dataset.
We have 993 distinct followees in our dataset.
```

Most Active Users I want to find out who are the most active users. Those who follow other users more. It is important to recognize such a user in our data. They are socialable and easy going and as a result are more more open and can spread the words in the network better.

[32]:
```
# user who follow the most
active_user=follows_df['user_uuid'].value_counts().to_frame()
active_user.head()
```

[32]:

	user_uuid
3cef3fd4-984e-481b-89da-c0692bcc529b	277
7f61564b-4203-4fb6-ac6a-2cc555465ed6	119
e5b4f882-7dca-4e63-a665-f412cb383e39	110
e54cea93-9399-47c1-9c57-be1571252e0a	106
5dda1c4a-8b66-440b-896c-8c0be38645b9	79

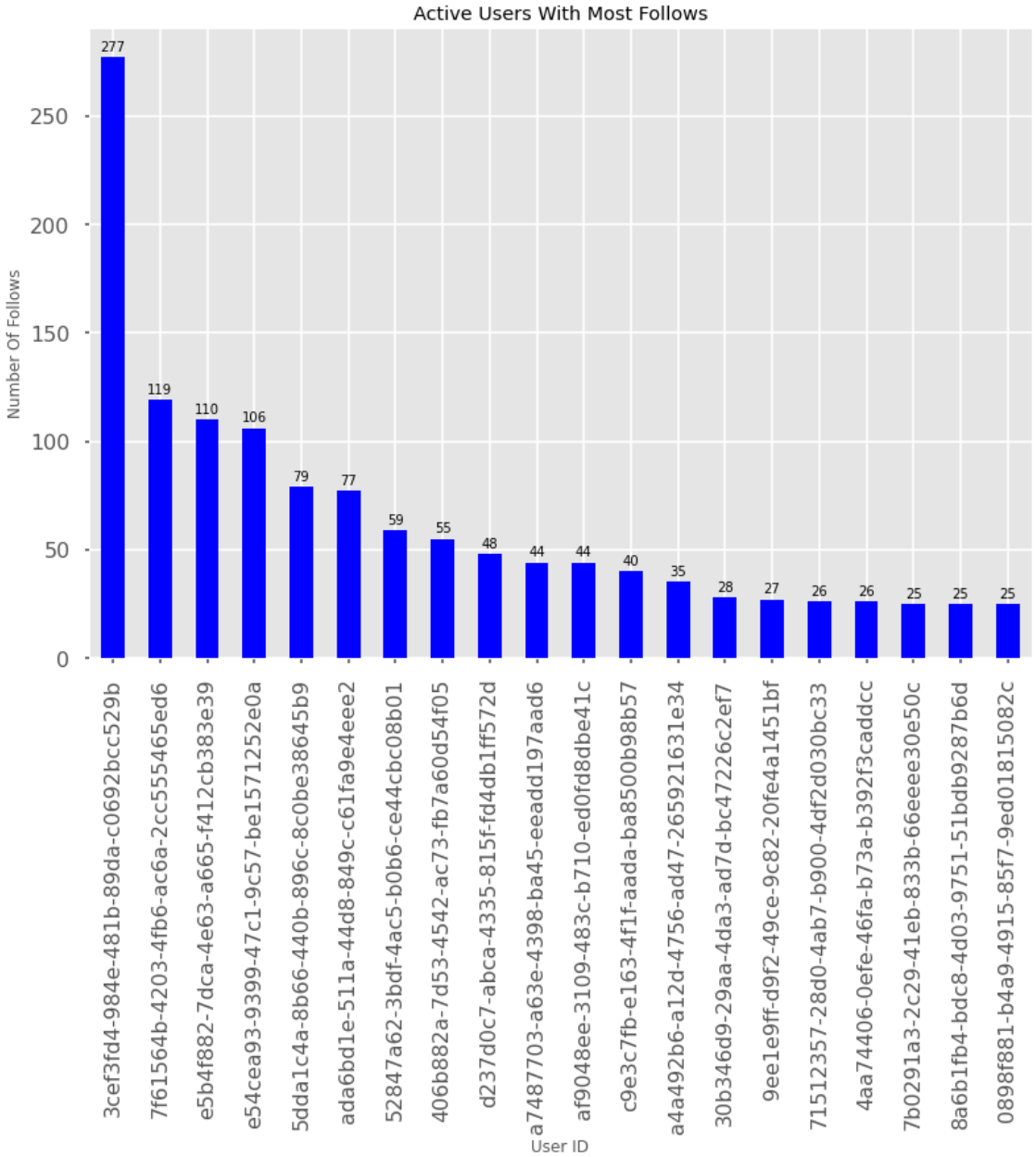
[33]:
```
print('We have {} users who follow more than 7 other users.'.
 ↪format(len(active_user[active_user['user_uuid']>7])))
print('We have {} users who follow less than or equal 7 users.'.
 ↪format(len(active_user[active_user['user_uuid']<=7])))
print('We have {} users who follow exactly one user.'.
 ↪format(len(active_user[active_user['user_uuid']==1])))
```

```
We have 159 users who follow more than 7 other users.
We have 1295 users who follow less than or equal 7 users.
We have 655 users who follow exactly one user.
```

I consider **7** as a threshold as the six degrees of separation theory states that any inhabitant of the Earth could meet anyone in the world with a maximum of six or fewer mutual connections between them and another person.

[34]:
```
ax = active_user.head(20).plot(kind = 'bar', color = 'b', legend = False)


#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")
```

```
# Add title and axis names
plt.title('Active Users With Most Follows')
plt.xlabel('User ID', fontsize='12')
plt.ylabel('Number Of Follows')


# Show graphic
plt.savefig('Images/Active Users with most follows.png',bbox_inches = 'tight')
plt.show()
```



Most popular user

[35]:
```
# user who is followed the most
popular_user = follows_df['follow_uuid'].value_counts().to_frame()
popular_user.head()
```

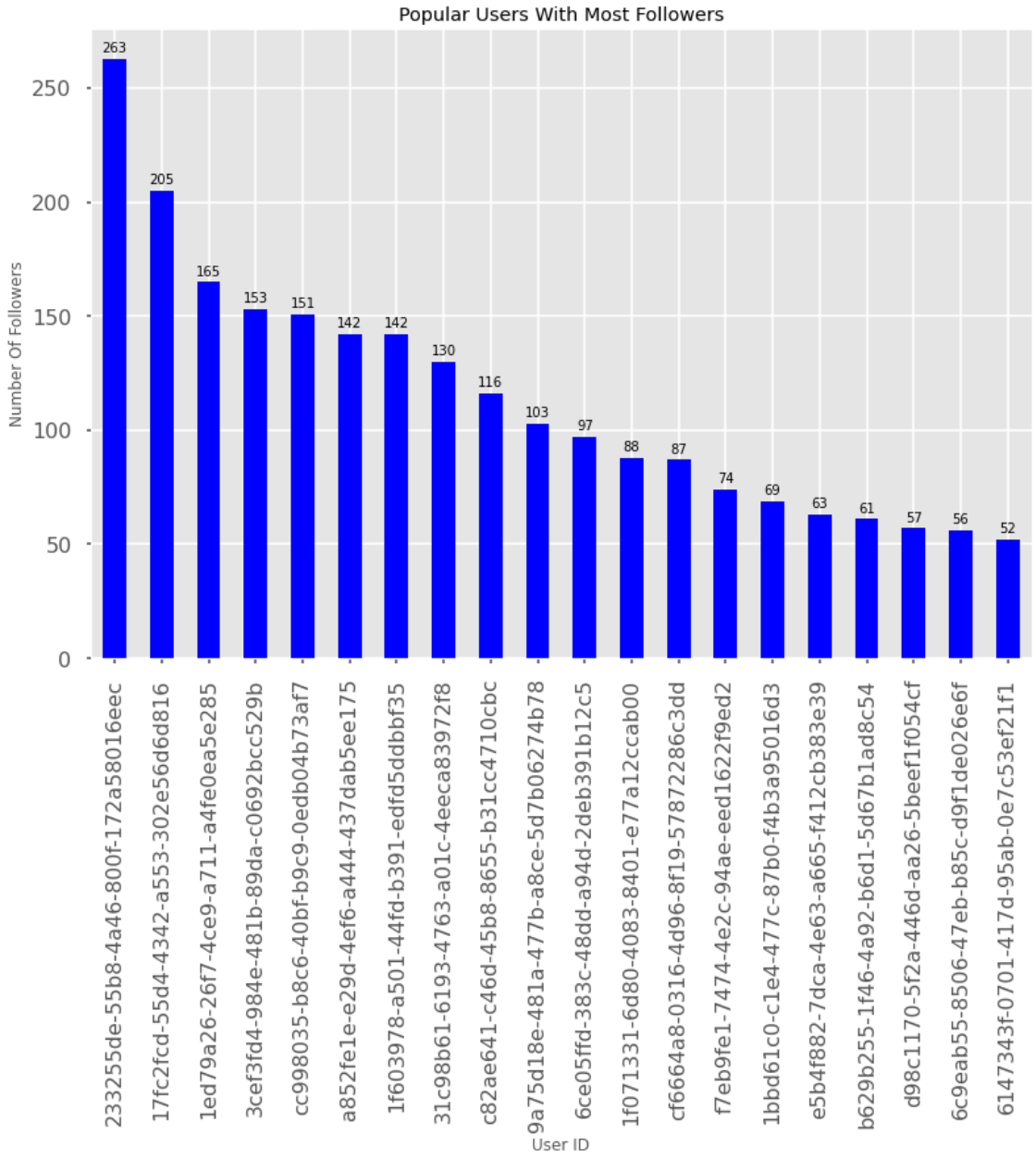
[35]:

	follow_uuid
233255de-55b8-4a46-800f-172a58016eec	263
17fc2fcd-55d4-4342-a553-302e56d6d816	205
1ed79a26-26f7-4ce9-a711-a4fe0ea5e285	165
3cef3fd4-984e-481b-89da-c0692bcc529b	153
cc998035-b8c6-40bf-b9c9-0edb04b73af7	151

[36]:
```
print('We have {} users who are followed by more than 7 other users.'.
 ↪format(len(popular_user[popular_user['follow_uuid']>7])))
print('We have {} users who are followed by less than or equal 7 users.'.
 ↪format(len(popular_user[popular_user['follow_uuid']<=7])))
print('We have {} users who are followed by exactly one user.'.
 ↪format(len(popular_user[popular_user['follow_uuid']==1])))
```

```
We have 126 users who are followed by more than 7 other users.
We have 867 users who are followed by less than or equal 7 users.
We have 528 users who are followed by exactly one user.
```

[37]:
```
ax = popular_user.head(20).plot(kind = 'bar', color = 'b', legend = False)

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

# Add title and axis names
plt.title('Popular Users With Most Followers')
plt.xlabel('User ID', fontsize='12')
plt.ylabel('Number Of Followers')


# Show graphic
plt.savefig('Images/popular Users with most followers.png',bbox_inches =␣
 ↪'tight')
plt.show()
```



Users Network (Directed Network)

[38]:
```
follows_df.head(2)
```

[38]:

	user_uuid	follow_uuid	date
0	7effce15-12ba-469c-b4b4-b529425488db	372fe527-470c-48a3-a95e-fe32cc4bcc15	2018-10-05 18:47:56
1	f262a138-9585-4570-9d2d-6815c4d8b6a6	ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd	

```
1  2020-06-29 14:57:37
```

The first step is to extract nodes and edges from the dataframe

[39]:
```
# spacify node's names
nodes = follows_df.user_uuid.unique().tolist()
nodes[:10]
```

[39]:
```
['7effce15-12ba-469c-b4b4-b529425488db',
 'f262a138-9585-4570-9d2d-6815c4d8b6a6',
 '18566d57-99ea-4f5d-aa23-7769f1b0dcc2',
 '958029e7-6897-4ae2-be6c-515730568371',
 '4afbf6d4-b841-44b1-8f04-eba6e6b102e3',
 '8e7730ec-96b4-4730-820d-9b8b64f7834f',
 '763930ba-0d7f-46dd-8e76-1fe3f18aba17',
 '19cc4b3b-7a64-441d-8e90-c68f9ceb6022',
 '41730565-9335-4b37-9f74-d167a63d7a00',
 'ce1b7a97-e6ac-4e2c-9251-9e69e7331276']
```

[40]:
```
# spacify edges
edges = list(zip(follows_df.user_uuid, follows_df.follow_uuid))
edges[:10]
```

[40]:
```
[('7effce15-12ba-469c-b4b4-b529425488db',
  '372fe527-470c-48a3-a95e-fe32cc4bcc15'),
 ('f262a138-9585-4570-9d2d-6815c4d8b6a6',
  'ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd'),
 ('18566d57-99ea-4f5d-aa23-7769f1b0dcc2',
  '2098ef7a-9215-4d2e-8e95-0ab7e745392f'),
 ('958029e7-6897-4ae2-be6c-515730568371',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('4afbf6d4-b841-44b1-8f04-eba6e6b102e3',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('8e7730ec-96b4-4730-820d-9b8b64f7834f',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('763930ba-0d7f-46dd-8e76-1fe3f18aba17',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('19cc4b3b-7a64-441d-8e90-c68f9ceb6022',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('41730565-9335-4b37-9f74-d167a63d7a00',
  '498439f1-52ed-4e3d-b07a-fe77061837b5'),
 ('ce1b7a97-e6ac-4e2c-9251-9e69e7331276',
  '8f237e9d-52c7-4ec8-ae23-6898070ba539')]
```

[41]:
```
G = nx.DiGraph()
G.add_nodes_from(nodes)
G.add_edges_from(edges)
```

[42]:
```
print(nx.info(G))
```

```
Name:
Type: DiGraph
Number of nodes: 1799
Number of edges: 5886
Average in degree:   3.2718
Average out degree:   3.2718
```

Why visualize a graph at all?

Usually, we simply have a set of vertices and edges as input. We can compute some statistics or graph metrics based on such data, but it is not enough to get an idea of structure. A good visualization can clearly show if there are some clusters or bridges in a graph, or maybe it is a uniform cloud, or something else.

Problems? Commonly, the visualization of a large graph looks messy because there are too many objects in one plot. Also, graph visualization algorithms mostly have awful algorithmic complexity

Layout Layout is a way to map a coordinate to each vertex. Usually, this is coordinates on a 2D plane. **What is a good layout?** It is easy to say if something looks good or bad. It is not so easy to name criteria, how could machine evaluate it. In order to make a "good" layout so-called, aesthetic metrics can be used. Here is some of them: 1. Minimum edges intersection It is obvious: too many intersections make the plot look messy. 2. Adjacent vertices are closer to each other than not adjacent This is logical that connected nodes should be close to each other. It represents the main information that is present in a graph by definition. 3. Communities are grouped into clusters If there are a set of vertices that are connected to each other more frequent then to other parts of the graph, they should look like a dense cloud. 4. Minimum overlapping edges and nodes. It is obvious too: if we cannot determine if there few vertices or one, then the readability of plot is poor.

There are plenty of different tools for Large Graph Visualization. However, despite graph visualization problem is relatively old and popular, there is a very bad situation with tools that can handle large graphs. - GraphViz - Gephi, the most powerful graph visualization tool that I know. It has GUI, it contains several layouts and a lot of graph analysis tools. There is also a lot of plugins written by the community. For example my favorite layout "Multigravity Force-Atlas 2" or sigma.js export tool, which creates an interactive web-page template based on your project. In Gephi users can color nodes and edges by its features. Although Gephi is abandoned by developers, for the this project I decided to use it for visualization. - igraph - LargeViz - Graph Embeddings and many others

Graph Theory

A good metric to begin with is network **density**.

This is simply the ratio of actual edges in the network to all possible edges in the network. Network density gives us a quick sense of how closely knit the network is.

[43]:
```
density = nx.density(G)
degree_centrality = nx.algorithms.degree_centrality(G)
print("Network Density:", density)
```

```
Network Density: 0.0018196983740194311
```

In this case, the density of our network is approximately **0.0018** On a scale of 0 to 1, not a very dense network, which comports with the visualization. A 0 would mean that there are no connections at all, and a 1 would indicate that all possible edges are present (a perfectly connected network): this network is on the lower end of that scale, but still far from 0.

Next is the **Graph Diameter**, it is a graph distance. In other words, a graph's diameter is the largest number of vertices which must be traversed in order to travel from one vertex to another when paths which backtrack, detour, or loop are excluded from consideration. Diameter, is the longest of all shortest paths. After calculating all shortest paths between every possible pair of nodes in the network, diameter is the length of the path between the two nodes that are furthest apart. The measure is designed to give us a sense of the network's overall size, the distance from one end of the network to another. However, it needs the graph to be fully connected and running this command on the Komodaa graph will yield an error telling the Graph is "not connected."

[44]:
```
# chech if the graph is fully connected then calculate the diameter
if nx.is_strongly_connected(G):
    nx.diameter(G)
else:
    print('The graph is not fully connected and has more than one component.')
```

```
The graph is not fully connected and has more than one component.
```

Instead we can calculate the diameter for the largest component of the Graph

[45]:
```
# Use nx.connected_components(undirected)/nx.
 ↪strongly_connected_components(directed) to get the list of components
components = nx.strongly_connected_components(G)

# Use the max() command to find the largest one
largest_component = max(components, key=len)

subgraph = G.subgraph(largest_component)
diameter = nx.diameter(subgraph)
print("Network diameter of largest component:", diameter)
```

```
Network diameter of largest component: 11
```

Since we took the largest component, we can assume there is no larger diameter for the other components. Therefore this figure is a good stand in for the diameter of the whole Graph.

[46]:
```
nx.average_degree_connectivity(G) # For a node of degree k - What is the␣
 ↪average of its neighbours' degree?
```

[46]:
```
{18: 42.49444444444445,
 5: 70.27466666666666,
 4: 61.98790322580645,
 8: 58.994140625,
 21: 43.92063492063492,
```

3: 72.03298611111111,
24: 31.7,
32: 20.0,
37: 28.905405405405407,
14: 37.38690476190476,
2: 60.75800711743772,
10: 60.48695652173913,
16: 37.125,
7: 61.839285714285715,
1: 64.45035971223021,
6: 60.634408602150536,
13: 53.84102564102564,
101: 22.742574257425744,
11: 60.93560606060606,
116: 27.310344827586206,
26: 45.88461538461539,
120: 30.55,
17: 51.3235294117647,
27: 21.83703703703704,
25: 35.586666666666666,
52: 20.798076923076923,
9: 43.550925925925924,
61: 9.87704918032787,
41: 18.317073170731707,
117: 2.247863247863248,
30: 36.2047619047619,
96: 7.953125,
15: 51.36,
49: 3.663265306122449,
22: 46.21590909090909,
36: 22.90740740740741,
23: 49.07608695652174,
51: 11.235294117647058,
20: 51.075,
153: 6.509803921568627,
12: 56.37820512820513,
19: 39.06578947368421,
430: 7.402325581395349,
39: 41.06410256410256,
71: 6.140845070422535,
57: 23.912280701754387,
173: 21.98843930635838,
74: 20.10810810810811,
85: 37.07058823529412,
29: 9.844827586206897,
100: 5.33,
152: 1.730263157894737,

```
 34: 33.61764705882353,
 60: 12.116666666666667,
 42: 8.333333333333334,
 31: 13.109677419354838,
 78: 6.666666666666667,
 65: 34.52307692307692,
 54: 17.98148148148148,
 43: 35.325581395348834,
 168: 0.02976190476190476,
 48: 14.583333333333334,
 53: 4.60377358490566,
 45: 17.666666666666668,
 210: 1.3857142857142857,
 38: 0.8157894736842105,
 28: 0.21428571428571427,
 131: 0.04580152671755725,
 263: 0,
 142: 0,
 103: 0,
 35: 0}
```

The final structural calculation is **triadic closure**. Triadic closure supposes that if two people know the same person, they are likely to know each other. If Fox knows both Wolf and Lion, then Wolf and Lion may very well know each other, completing a triangle in the visualization of three edges connecting Fox, Wolf, and Lion. The number of these enclosed triangles in the network can be used to find clusters and communities of individuals that all know each other fairly well.

One way of measuring triadic closure is called transitivity. Transitivity is the ratio of all triangles over all possible triangles. A possible triangle exists when one person (Fox) knows two people (Wolf and Lion). So transitivity, like density, expresses how interconnected a graph is in terms of a ratio of actual over possible connections.

Measurements like transitivity and density concern likelihoods rather than certainties. Transitivity allows us a way of thinking about all the relationships in your graph that may exist but currently do not.

[47]:
```
triadic_closure = nx.transitivity(G)
print("Triadic closure:", triadic_closure)
```

```
Triadic closure: 0.026756633640815396
```

like density, transitivity is scaled from 0 to 1, and you can see that the network's transitivity is about **0.0267**, somewhat higher than its **0.0018** density. Because the graph is not very dense, there are fewer possible triangles to begin with, which may result in slightly higher transitivity. That is, nodes that already have lots of connections are likely to be part of these enclosed triangles. To back this up, you'll want to know more about nodes with many connections.

After getting some basic measures of the entire network structure, a good next step is to find which nodes are the most important ones in your network.

In network analysis, measures of the importance of nodes are referred to as centrality

measures.‘

Degree is the simplest and the most common way of finding important nodes. A node’s degree is the sum of its edges. If a node has three lines extending from it to other nodes, its degree is three. Five edges, its degree is five.

The nodes with the highest degree in a social network are the people who know the most people. These nodes are often referred to as hubs, and calculating degree is the quickest way of identifying hubs.

[48]:
```
degree_dict = dict(G.degree(G.nodes()))
nx.set_node_attributes(G, degree_dict, 'degree')
```

[49]:
```
sorted_degree = sorted(degree_dict.items(), key=itemgetter(1), reverse=True)
```

[50]:
```
print("Top 20 nodes by degree:")
for d in sorted_degree[:20]:
    print(d)
```

```
Top 20 nodes by degree:
('3cef3fd4-984e-481b-89da-c0692bcc529b', 430)
('233255de-55b8-4a46-800f-172a58016eec', 263)
('17fc2fcd-55d4-4342-a553-302e56d6d816', 210)
('e5b4f882-7dca-4e63-a665-f412cb383e39', 173)
('1ed79a26-26f7-4ce9-a711-a4fe0ea5e285', 168)
('a852fe1e-e29d-4ef6-a444-437dab5ee175', 153)
('cc998035-b8c6-40bf-b9c9-0edb04b73af7', 152)
('1f603978-a501-44fd-b391-edfd5ddbbf35', 142)
('31c98b61-6193-4763-a01c-4eeca83972f8', 131)
('7f61564b-4203-4fb6-ac6a-2cc555465ed6', 120)
('c82ae641-c46d-45b8-8655-b31cc4710cbc', 117)
('e54cea93-9399-47c1-9c57-be1571252e0a', 116)
('9a75d18e-481a-477b-a8ce-5d7b06274b78', 103)
('ada6bd1e-511a-44d8-849c-c61fa9e4eee2', 101)
('6ce05ffd-383c-48dd-a94d-2deb391b12c5', 100)
('cf6664a8-0316-4d96-8f19-57872286c3dd', 96)
('1f071331-6d80-4083-8401-e77a12ccab00', 96)
('5dda1c4a-8b66-440b-896c-8c0be38645b9', 85)
('f7eb9fe1-7474-4e2c-94ae-eed1622f9ed2', 78)
('6c9eab55-8506-47eb-b85c-d9f1de026e6f', 74)
```

There are other centrality measures that can tell you about more than just hubs. **Eigenvector centrality** is a kind of extension of degree. It looks at a combination of a node’s edges and the edges of that node’s neighbors. Eigenvector centrality cares if you are a hub, but it also cares how many hubs you are connected to.

It’s calculated as a value from 0 to 1: the closer to one, the greater the centrality. Eigenvector centrality is useful for understanding which nodes can get information to many other nodes quickly. If you know a lot of well-connected people, you could spread a message very efficiently.

Betweenness centrality is a bit different from the other two measures in that it doesn’t care

about the number of edges any one node or set of nodes has. Betweenness centrality looks at all the shortest paths that pass through a particular node. Betweenness centrality is also expressed on a scale of 0 to 1 and is fairly good at finding nodes that connect two otherwise disparate parts of a network. If you're the only thing connecting two clusters, every communication between those clusters has to pass through you. In contrast to a hub, this sort of node is often referred to as a broker. Betweenness centrality is not the only way of finding brokerage (and other methods are more systematic), but it's a quick way of giving you a sense of which nodes are important not because they have lots of connections themselves but because they stand between groups, giving the network connectivity and cohesion.

[51]:
```
betweenness_dict = nx.betweenness_centrality(G) # Run betweenness centrality
eigenvector_dict = nx.eigenvector_centrality(G) # Run eigenvector centrality

# Assign each to an attribute in your network
nx.set_node_attributes(G, betweenness_dict, 'betweenness')
nx.set_node_attributes(G, eigenvector_dict, 'eigenvector')
```

[52]:
```
sorted_eigenvector = sorted(eigenvector_dict.items(), key=itemgetter(1),␣
↪reverse=True)

print("Top 20 nodes by eigenvector centrality:")
for b in sorted_eigenvector[:20]:
    print(b)
```

```
Top 20 nodes by eigenvector centrality:
('233255de-55b8-4a46-800f-172a58016eec', 0.33266079736633003)
('17fc2fcd-55d4-4342-a553-302e56d6d816', 0.32067296193815653)
('1f603978-a501-44fd-b391-edfd5ddbbf35', 0.2599995488674807)
('3cef3fd4-984e-481b-89da-c0692bcc529b', 0.2540671320258265)
('a852fe1e-e29d-4ef6-a444-437dab5ee175', 0.22852299905172455)
('cc998035-b8c6-40bf-b9c9-0edb04b73af7', 0.19504616539726835)
('31c98b61-6193-4763-a01c-4eeca83972f8', 0.1564087282150767)
('1f071331-6d80-4083-8401-e77a12ccab00', 0.15080839217447564)
('c82ae641-c46d-45b8-8655-b31cc4710cbc', 0.1463700015770921)
('1ed79a26-26f7-4ce9-a711-a4fe0ea5e285', 0.13608870045734683)
('f7eb9fe1-7474-4e2c-94ae-eed1622f9ed2', 0.13485112050812012)
('cf6664a8-0316-4d96-8f19-57872286c3dd', 0.13306877329977343)
('6ce05ffd-383c-48dd-a94d-2deb391b12c5', 0.13161623769776556)
('1bbd61c0-c1e4-477c-87b0-f4b3a95016d3', 0.11568972379164869)
('9a75d18e-481a-477b-a8ce-5d7b06274b78', 0.11542285028885722)
('8d853d53-dff3-4c64-998a-fcb6846ff74b', 0.10651843716610093)
('d98c1170-5f2a-446d-aa26-5beef1f054cf', 0.09932733390160216)
('e5b4f882-7dca-4e63-a665-f412cb383e39', 0.09794150281746294)
('abf55690-de58-4b4e-964f-8cddcecc0ae6', 0.09748812035826519)
('6c9eab55-8506-47eb-b85c-d9f1de026e6f', 0.09018868845725658)
```

[53]:
```
sorted_betweenness = sorted(betweenness_dict.items(), key=itemgetter(1),␣
↪reverse=True)
```

```
print("Top 20 nodes by betweenness centrality:")
for b in sorted_betweenness[:20]:
    print(b)
```

```
Top 20 nodes by betweenness centrality:
('3cef3fd4-984e-481b-89da-c0692bcc529b', 0.17181568706495995)
('1f071331-6d80-4083-8401-e77a12ccab00', 0.061100219607097744)
('17fc2fcd-55d4-4342-a553-302e56d6d816', 0.041879041479161924)
('e5b4f882-7dca-4e63-a665-f412cb383e39', 0.038061797844850254)
('cf6664a8-0316-4d96-8f19-57872286c3dd', 0.02958144828013708)
('b8e9c4e8-6821-47af-88df-afbdb232a935', 0.023323905546090085)
('a7487703-a63e-4398-ba45-eeadd197aad6', 0.021356548440168835)
('91674087-2b1b-4fe1-8569-ad6207922522', 0.015288483872622683)
('1bbd61c0-c1e4-477c-87b0-f4b3a95016d3', 0.015155226342864477)
('ada6bd1e-511a-44d8-849c-c61fa9e4eee2', 0.014369080930366162)
('a852fe1e-e29d-4ef6-a444-437dab5ee175', 0.013598259220916547)
('e54cea93-9399-47c1-9c57-be1571252e0a', 0.013264513693908228)
('f8fe6f0a-cefd-4ced-b08f-914c6fc0c29b', 0.013259923459299902)
('61c3b4b1-e33a-47d6-b62a-3dd2c78e4497', 0.012268990999026199)
('31c98b61-6193-4763-a01c-4eeca83972f8', 0.012125831363723687)
('6ce05ffd-383c-48dd-a94d-2deb391b12c5', 0.011322632268717561)
('0951273a-dc83-42ea-a44e-05b6c728d404', 0.01077131737331839)
('1f7c0ee4-4933-478f-a269-6f07c7fb9c0f', 0.01076516749578788)
('a124212b-959b-4676-9fdc-ab223cf56c58', 0.010757445035763149)
('5dda1c4a-8b66-440b-896c-8c0be38645b9', 0.010652058991306994)
```

Many, but not all, of the nodes that have high degree also have high betweenness centrality.

[54]:
```
#First get the top 20 nodes by betweenness as a list
top_betweenness = sorted_betweenness[:20]

#Then find and print their degree
for tb in top_betweenness: # Loop through top_betweenness
    degree = degree_dict[tb[0]] # Use degree_dict to access a node's degree,␣
 ↪see footnote 2
    print("Name:", tb[0], "| Betweenness Centrality:", tb[1], "| Degree:",␣
 ↪degree)
```

```
Name: 3cef3fd4-984e-481b-89da-c0692bcc529b | Betweenness Centrality:
0.17181568706495995 | Degree: 430
Name: 1f071331-6d80-4083-8401-e77a12ccab00 | Betweenness Centrality:
0.061100219607097744 | Degree: 96
Name: 17fc2fcd-55d4-4342-a553-302e56d6d816 | Betweenness Centrality:
0.041879041479161924 | Degree: 210
Name: e5b4f882-7dca-4e63-a665-f412cb383e39 | Betweenness Centrality:
0.038061797844850254 | Degree: 173
Name: cf6664a8-0316-4d96-8f19-57872286c3dd | Betweenness Centrality:
0.02958144828013708 | Degree: 96
```

```
Name: b8e9c4e8-6821-47af-88df-afbdb232a935 | Betweenness Centrality:
0.023323905546090085 | Degree: 39
Name: a7487703-a63e-4398-ba45-eeadd197aad6 | Betweenness Centrality:
0.021356548440168835 | Degree: 60
Name: 91674087-2b1b-4fe1-8569-ad6207922522 | Betweenness Centrality:
0.015288483872622683 | Degree: 39
Name: 1bbd61c0-c1e4-477c-87b0-f4b3a95016d3 | Betweenness Centrality:
0.015155226342864477 | Degree: 71
Name: ada6bd1e-511a-44d8-849c-c61fa9e4eee2 | Betweenness Centrality:
0.014369080930366162 | Degree: 101
Name: a852fe1e-e29d-4ef6-a444-437dab5ee175 | Betweenness Centrality:
0.013598259220916547 | Degree: 153
Name: e54cea93-9399-47c1-9c57-be1571252e0a | Betweenness Centrality:
0.013264513693908228 | Degree: 116
Name: f8fe6f0a-cefd-4ced-b08f-914c6fc0c29b | Betweenness Centrality:
0.013259923459299902 | Degree: 61
Name: 61c3b4b1-e33a-47d6-b62a-3dd2c78e4497 | Betweenness Centrality:
0.012268990999026199 | Degree: 6
Name: 31c98b61-6193-4763-a01c-4eeca83972f8 | Betweenness Centrality:
0.012125831363723687 | Degree: 131
Name: 6ce05ffd-383c-48dd-a94d-2deb391b12c5 | Betweenness Centrality:
0.011322632268717561 | Degree: 100
Name: 0951273a-dc83-42ea-a44e-05b6c728d404 | Betweenness Centrality:
0.01077131737331839 | Degree: 18
Name: 1f7c0ee4-4933-478f-a269-6f07c7fb9c0f | Betweenness Centrality:
0.01076516749578788 | Degree: 32
Name: a124212b-959b-4676-9fdc-ab223cf56c58 | Betweenness Centrality:
0.010757445035763149 | Degree: 54
Name: 5dda1c4a-8b66-440b-896c-8c0be38645b9 | Betweenness Centrality:
0.010652058991306994 | Degree: 85
```

Community detection with modularity Is our network one big, happy family where everyone knows everyone else? Or is it a collection of smaller subgroups that are only connected by one or two intermediaries?

A community has high density relative to other nodes within its module but low density with those outside.

[55]:
```
H = nx.Graph(G)
communities = list(community.greedy_modularity_communities(H))
```

[56]:
```
modularity_dict = {}                                    # Create a blank␣
↪dictionary
for i,c in enumerate(communities):                      # Loop through the list␣
↪of communities, keeping track of the number for the community
    for name in c:                                      # Loop through each␣
↪person in a community
```

```
        modularity_dict[name] = i                          # Create an entry in␣
↪the dictionary for the person, where the value is which group they belong to.

# Add modularity information
nx.set_node_attributes(G, modularity_dict, 'modularity')
```

[57]:
```
# First get a list of just the nodes in that class
class0 = [n for n in G.nodes() if G.nodes[n]['modularity'] == 0]

# Then create a dictionary of the eigenvector centralities of those nodes
class0_eigenvector = {n:G.nodes[n]['eigenvector'] for n in class0}

# Then sort that dictionary and print the first 5 results
class0_sorted_by_eigenvector = sorted(class0_eigenvector.items(),␣
↪key=itemgetter(1), reverse=True)

print("Modularity Class 0 Sorted by Eigenvector Centrality:")
for node in class0_sorted_by_eigenvector[:5]:
    print("Name:", node[0], "| Eigenvector Centrality:", node[1])
```

```
Modularity Class 0 Sorted by Eigenvector Centrality:
Name: 233255de-55b8-4a46-800f-172a58016eec | Eigenvector Centrality:
0.33266079736633003
Name: c82ae641-c46d-45b8-8655-b31cc4710cbc | Eigenvector Centrality:
0.1463700015770921
Name: f7eb9fe1-7474-4e2c-94ae-eed1622f9ed2 | Eigenvector Centrality:
0.13485112050812012
Name: cf6664a8-0316-4d96-8f19-57872286c3dd | Eigenvector Centrality:
0.13306877329977343
Name: b5f81ad9-ca01-4728-8a0c-c14f0edf3e66 | Eigenvector Centrality:
0.08582988577514133
```

Using eigenvector centrality as a ranking can give a sense of the important people within this modularity class.

[58]:
```
for i,c in enumerate(communities):                         # Loop through␣
↪the list of communities
    if len(c) > 2:                                         # Filter out␣
↪modularity classes with 2 or fewer nodes
        print('Class '+str(i)+':','\n', list(c), '\n')                #␣
↪Print out the classes and their members
```

```
Class 0:
 ['4cd14ded-6376-4251-8244-3055914be497', '07e72109-d983-4d7f-9adf-
befe2f33bd41', '4b4e8392-6598-4523-b5ea-86fe1e9d62c1',
'8bd54eaa-82a7-40a6-b1fc-76c599201264', '29493aea-00cc-4a8e-83b9-4ec9c87ebe44',
'a2607f08-a0ad-4972-8c18-05ac05e42103', '9f2f64e5-8afc-4518-9607-ffccb0020f97',
'0f11c9fc-10b8-4b6e-99fc-5c775a7940ea', 'cee3fea4-2237-4597-8e82-4327a5cfd5f3',
```

```
'cf6664a8-0316-4d96-8f19-57872286c3dd', '63b737d0-1d21-46d6-ac9d-1f63bf74bb72',
'c823cdf4-324e-4afd-b8be-526afdbaf0bc', 'b0eddeb6-0418-4bc4-918c-56caa0cf288d',
'f9b535b7-98e5-48bf-b467-bb9d2f2bc9c0', '8c942c58-9029-4edf-851e-642327d6cf8c',
'5bcfe486-512f-442f-b15c-a4494aec181c', '1475728d-dbb5-4bc0-af7f-d84ed76d29cd',
'94b23a7a-d191-47a7-b462-5cf36185ce91', '06fb04b4-2790-4ae2-84d4-2b3c754a119b',
'0e6d45b5-978b-4a14-95e3-8e7c59d74b79', 'aa0d299c-92b4-46ac-a778-141e12ffa5dc',
'd1df4da4-f9ce-48d2-b483-76a0db2c7a0b', '949543ad-ee51-4cd2-abd7-f97933b77e01',
'49d246fe-e634-4202-bd73-39bbac8d39a8', 'bcb42d91-bcc2-41c7-b518-d24d1d5d09c0',
'fced77f2-3247-4eea-bc42-69b9a967b0cb', '9ca53cb9-6373-4b7b-8338-4e75a399c0b9',
'198dac16-6d4e-4152-9223-e1149962f2b0', '4cd74252-8a0d-4606-89aa-0ca6e68b2263',
'85b91b0a-ec55-4768-a31a-79c592935cf0', 'a1b8285a-afe4-4959-817e-05b27baeed16',
'08618673-2f9c-437d-95a7-ef4aef85ca10', 'b3acc910-706f-41c1-9dba-fdc6b5927163',
'50a28be8-a8e1-4580-bb88-0b58b44a7053', '4ef78744-005e-4d97-b903-7ac9f7eb567a',
'31af5342-ec7a-44eb-b9e2-bcba4e9ea885', '4e63e137-22b5-4f38-91f8-71c2bc842403',
'f958c10c-7dd3-4e61-abf4-010f2016bd8d', '5ca467f3-b03f-4701-81bd-1143209ff5ab',
'da6f1b78-dd45-4cf8-a54f-878f880275e9', '89b6e5e9-699c-4df6-9ffd-3b791de028b7',
'1ca10126-e1ae-4dd8-8526-f828c0c41dba', '9f14d58c-82b0-49e7-8183-14f8b517034c',
'f4fd10ab-85ed-4293-bff0-5082f0081044', 'e527a43d-e223-4f31-9fb3-b3e3616a78ff',
'af0a750c-6068-41f4-82d0-b458284453d4', '3ff2d9f2-6c03-468e-bb37-b25a8dc22566',
'21353b65-4636-4254-9fdf-bb3a4fc9fbee', 'edc5a7b4-bb81-4c9a-b2f8-e2af2bb2b6a0',
'46c8535b-897a-402f-a267-1b511e92bee2', '440f1406-ac17-4efb-bdbb-d10c1fdefee7',
'ae43ce31-a415-41cc-98c5-ee59585c76f8', '8287c674-c380-4e1e-85fc-b21f3cfb698b',
'f40c3a11-3aec-4472-a83b-14a3822c77ac', '094cb72d-d653-401b-8c99-0d68e9641880',
'1251084a-fbe0-42ac-b223-4aa07bac4dcc', '6b92ff75-6603-4ce6-8b8f-9b5c9507f3c4',
'203a766c-3004-4608-801d-54bb2ad53fa7', 'd237d0c7-abca-4335-815f-fd4db1ff572d',
'82f39084-c872-407a-bb6e-5cb0484f2b24', 'eb7f6d46-3eab-4e60-9d3c-c904902c48da',
'70026658-82a5-4cf1-8ed6-7be822787519', '1decef9b-a0ac-4e49-859c-dc776210c1ee',
'763930ba-0d7f-46dd-8e76-1fe3f18aba17', '54ec1a3d-526a-46eb-b9c6-e836bfb2ecbd',
'383c4c41-3e3b-446b-9c39-8deec34df9ab', '56a7acdc-8a1a-482e-ad07-d9c716b9486a',
'23ec0f78-0e70-4473-98d9-a2f70c3aa404', '34da91dd-7d7b-465e-97d0-3800b68d2ea4',
'bac23646-c4d3-4812-b970-9bb8b2a741b1', '911adfe9-d5d3-4e59-bf6f-8768c5c2060e',
'e1dcce06-0bc5-4e30-b93d-ec8c5975973d', 'c5fcf9c0-f5c4-48a2-8a5b-00ff637f71e0',
'1dfd39d5-3050-4b70-a3cb-c0d69e1ec7a8', 'e6bff75d-e680-4a45-86a0-3e018b2b0a52',
'ff020f9b-c6d9-4f83-b5bd-f0314d206789', '5bde8989-6306-441a-ad0e-8fd67974d075',
'8ff35f92-3638-4cbd-9305-1525201323e3', '577a1f63-863e-4fa8-8598-81f67c30ea1b',
'd781f94d-d479-4d74-8ef4-3470b28d7555', 'a6a3ae9b-2885-4981-9b8b-1b6bdabd05c7',
'e2fe64db-4653-4487-8e5e-1ba8af853eea', '9be96114-80a8-4e5c-8d4f-e804f1ab0382',
'a3f711f9-7a3a-4069-bcfb-8d442a1ef30b', '9d0a2aed-4edd-4b31-a3fe-4047c62f5121',
'e236cb2c-8a6e-4712-abcf-66b39682fdcc', 'b2c570eb-393e-4007-912a-e7a3d557348a',
'f4046c8a-6254-4211-a0a6-f1c138247690', 'aee0b185-b67a-45a2-816d-2c355f0b1df1',
'fda45378-e801-40fd-bd55-d6240b4f6ef2', '0f0f4349-5dd7-4c1b-880c-f04a3ac37a27',
'b629b255-1f46-4a92-b6d1-5d67b1ad8c54', '97e22a0a-2309-487f-a8a3-25b1416ef04f',
'e54cea93-9399-47c1-9c57-be1571252e0a', '64e217b8-623e-4bf4-9b1f-31e23db98ec3',
'4ebe2cf7-63d2-42f1-bc9d-d8cbfc21d5ce', '0ad461bd-76f5-4061-aa59-1208c9453935',
'73399ca8-dde6-4b02-ad82-8987c79191f8', '807424a0-2ec0-4414-b451-0d80a42a12ac',
'1becf832-66a3-452b-b625-b8061f7548b8', 'cff5e45b-49e6-44b2-92f8-41b38755557f',
'942fe883-fc87-4cb8-a9ff-5c2d2fe2d598', '74abac40-2694-4ba9-95dd-7eada1bf01b8',
'18e5a061-3c33-4bbe-8de1-30bbdfa081ae', 'd0040d51-4211-466f-91fc-ff2906c0d5af',
```

```
'9e121f52-a40d-4384-8cd5-7b4a96822275', 'f6f460a5-e216-4ef8-86af-f3539b42f007',
'ac215a3f-4b77-4c8e-bbf5-418c7e71faf7', 'cc50172d-c120-43d2-8692-8d16c26fe44b',
'87a821d6-d778-44c4-88ad-c8d2f194b150', '5d34e5de-d18a-4edb-866b-3be2dbb65e0a',
'0c0bc21b-ef45-4824-9d93-fa71e4d545ad', '7b5ad1ce-46ae-408f-9543-15007879071e',
'be5d9049-8878-4f20-9881-a22d8f85845f', 'a9549e08-ebbc-463b-ac07-16fdbbccfcc9',
'1c6fd6f4-401a-4588-be1e-2f0492b1e1d9', '4b8bd898-852b-44d0-9741-f64a8435a072',
'3d5ee4dc-1817-4578-b388-92e4495b6eca', 'd57b3e35-3709-43a1-b4fd-cb1b4c699f2b',
'50d40c89-bb28-4d55-b29d-3647818dd5bf', '958029e7-6897-4ae2-be6c-515730568371',
'ba11ca6a-b6da-4ae3-9f70-ccf06fd18118', 'a4a492b6-a12d-4756-ad47-265921631e34',
'e5d33003-9efb-4878-8964-65b9271585f3', '956e7c05-0206-4a2d-a37a-1131462eefcf',
'bd3dc6bf-6879-4a2c-929b-ae0632dd4a9d', '6147343f-0701-417d-95ab-0e7c53ef21f1',
'0f20febd-2861-40b4-85b8-51ac53c0a06b', 'd4f0b071-c979-4e8a-bbdc-dda012993269',
'71512357-28d0-4ab7-b900-4df2d030bc33', 'c8f338a7-9df6-4669-bc6b-3a4afd4c3726',
'2f910651-0345-4dfc-82c8-64edfba9786f', '861558bb-8c03-40d6-966a-d37bf739fc1d',
'30980c69-567f-4466-a229-ac6af14934fb', '84ca8d6a-84e5-49ed-9936-58b0bf10063f',
'81d8c9d3-3855-40a9-9abc-a6b84b6b8213', '202def3f-fe21-43f4-969c-8f8b39ccdc8d',
'fae08429-b712-4fbc-811c-cce5a2c02809', 'e9f65b56-a6cc-4755-8258-a0e524123da5',
'6affcffa-791c-4c0b-b0c1-4c3037278645', '8d0e7b24-147a-4477-bb77-71866d4f08fe',
'bb65030c-3e24-4bf3-9b9f-23982607695c', '1eacfe58-9aaa-4fd7-a907-ee9372537f3a',
'fe957a2c-c83b-465c-836d-02d7cc9a89a4', '78b2181c-7d8e-48ef-8c4c-7c1f2ec0deab',
'298b8070-9e7b-40c6-bde3-325ca0676d72', '46e3be91-d7f6-4f2c-a107-251f606350ca',
'5c717cf0-32fc-40fb-8f56-1e5bd7afa10e', '2d69de7b-dd06-4639-86e1-1e67bc4dd91b',
'6077e4ee-f070-448f-8628-8135e0ae76d2', '38f7c5ca-2fc5-4ea4-94db-02a418f04156',
'4975942e-1169-42ac-b9ec-c52a2ea2ba6a', 'd1213db6-91ee-416c-9c16-2a54a304a3a0',
'c5333d13-497c-4e04-9e80-9bf9b51761ad', '3216ce56-08f6-4783-bcc8-2c8def2f2ba6',
'e20f932f-c879-4ec3-b073-a4192e708ce4', '757d2c4d-a231-475d-92da-8d851be12a8a',
'e52047ce-7347-4204-8497-7be2ae341ae0', 'b24cccee-12d1-4971-a8a7-362b4de545ee',
'104b4141-40de-444d-afd0-cb909bddbf00', '1c10d2cf-d351-476a-986c-25661cfa34c1',
'0c907f24-ceb6-4602-9b0b-59125e6d10cb', 'be110e39-4e13-4bc5-be23-203bf735672a',
'45d5bf15-7255-4714-b125-37f9b181e6ee', 'f7b4fd3f-5521-4619-9d4a-a5d725ea50b1',
'fb22c11e-ca28-41cb-a4e5-efd0ee4744b5', '931e51dc-4270-4f4d-b76f-a25359caba16',
'54d06aa3-21c7-4386-9fc4-8f89699c57aa', 'ba6906eb-49cc-4106-b2c4-4783382fba01',
'60b15974-5ea8-4c7c-a29c-1c0122209ec8', '49247db9-c4d5-4eab-8660-1372fcebf960',
'89da9ace-2478-401e-b94d-d65cdbae4779', 'a8e33826-79df-4102-aa60-8a09a04f4854',
'480acdc4-4c22-4044-b1d1-119ee25465a1', '6eeb8191-2f48-4825-ba9e-d23a189ea530',
'b3494c85-651c-4011-ba63-e9b8521b0f8b', 'decd8465-773a-4052-a7cc-796fd79b61e6',
'0d65bba2-efd2-42f1-bbea-e9e178849bc0', '0951273a-dc83-42ea-a44e-05b6c728d404',
'f26a1c5b-ee04-4f79-a17e-55e32a8f0ad5', '9770d423-165f-4113-8fa8-041f98dd616d',
'0c31b4bd-35ca-4528-b1a4-59199c0da0a5', '20d6559e-2fc6-43fd-948c-ab5e4b776461',
'7af924cc-ae8b-4002-8543-2268716f89ab', '79c1c583-b9ee-4609-acb4-5d9778e776ca',
'aab823ea-d85e-4c9a-8188-25accddc2443', '7c6d8464-05b7-4721-836e-6a82990869a9',
'34876538-452c-4a5a-a148-f4dea6022c28', 'b45a5a42-608f-48bf-bbe6-6feb3a76478c',
'288ba742-5464-4077-9a1a-d76b93b88e10', '14abf3b6-2edb-4ed8-a8ec-ac2d9393566a',
'c92b48c6-2e8f-4189-8b60-98327df5bade', 'ff8554be-c603-41b1-ae35-1b81b85510ee',
'40bcdb40-fed7-4221-aa9b-3841476fb050', '8e7730ec-96b4-4730-820d-9b8b64f7834f',
'cab0af32-de69-4b8b-bd4e-2e69437fc657', '6c1f34f5-90c4-4f6a-b5ca-b55f278a0f96',
'0a7ced45-63df-472f-b98a-e94f1c9e2ed4', '7e5df497-80c6-4a14-b807-103e6cd5cf19',
'd747816c-6db9-4a8a-95d8-a06f42a83558', 'b42f1fd8-83e5-4c45-9bea-d8425ad8f3f1',
```

```
'd7f16f8a-9ec7-4a4d-9861-f03f9de2b1af', '5f02e5df-4124-403d-a6c4-80fd93156627',
'd672dc14-9ff4-4421-a60f-547be05912fd', 'ef190f74-ca26-4a9c-a2c2-5cceb0a172bf',
'f9937978-1b18-450b-a6f4-f895b84de003', '49f1c366-4ce3-46ec-903d-65993f960505',
'fbe9627a-7326-4211-8dad-38ca91a662b5', '6421e091-dd08-4b1d-b31b-08e26c638b3f',
'2106d03c-9c05-4e96-a7b4-8b2bf9258c3a', '72b857d6-eb8e-474e-8197-f7ef52a4e5ee',
'67b65a8e-b276-41b9-be30-1a0aa515667f', '772316b9-a612-4543-800b-b92e5a580b4c',
'899f0535-25b9-40f8-86e8-0173b7c123db', 'c82ae641-c46d-45b8-8655-b31cc4710cbc',
'21aa99d6-ada9-48de-8776-87b67e4b5854', 'c9df4001-ecf2-4ba6-a22b-4ed02e0e681f',
'e4b8d634-a533-4cfb-bd54-17da07e50e0c', '0bb716a6-368d-42a5-9a04-d4fbe4475be8',
'ad603986-9d91-4bad-9166-e152a6267040', 'ea2a998d-c360-41f9-b0a6-b0e021d2a11f',
'182575fb-a376-4d58-a3b3-70e9441b0098', '08fceb0a-53bb-4a7b-a682-bbfbd6203ce7',
'211e3731-5f7d-4805-bebb-79ec39313ac5', '7ed3c1d4-ad71-4fa1-98c9-5e264929ae27',
'786d286b-0d39-41e5-ab25-6936de5a6033', 'ea4939a7-c597-49cf-ac1b-536023cc7522',
'f5e07f2f-ebf8-42b2-b2f4-bfeec9d93a88', '4afbf6d4-b841-44b1-8f04-eba6e6b102e3',
'78393319-3347-4ac3-b868-bca150dd8407', 'f7eb9fe1-7474-4e2c-94ae-eed1622f9ed2',
'c39bee37-385a-4fbd-a1b8-a4512bd9bc8e', 'a60f4780-2e54-4d6e-a331-05e592f7ea2d',
'93491b2c-fab7-45d6-b6e1-bfbded31bc3c', '4abec3a0-066d-4fe0-b564-e67805de4d81',
'92582f16-6596-4f50-91b9-45e8d64c943f', '333c0261-9000-4876-8514-485603dcf1e3',
'af70974b-de27-44a4-9c58-8ac53907500b', 'af49dadd-ad89-4d2e-b965-ca4ab57aadf7',
'71ff9a81-eecc-4f52-950a-9b4e004eb915', '0fc48758-5ba9-4231-9a85-684206c4703c',
'26dd2f96-bbb2-473d-980b-38dd2b4ab1d3', '89d70e63-3e54-4a83-81e9-4ec8104e32c4',
'8e202561-44f0-43f8-80f4-9827561179fb', 'efe01705-82f2-4fe8-a3a2-b7d3ff0c88b0',
'216a4e4c-c5c6-4da8-8262-2061557d5da1', '357fff60-497b-470a-a1c1-dfc283b27177',
'3e51bad9-1bb3-40a0-8bc8-c443b63b8912', '07d85486-4097-41ad-991e-6a34fed93174',
'9976c84b-dcec-4916-95b5-afee1329b5a6', 'd2168086-a2fc-4749-bf51-1ab0e7a1e839',
'b1733a9d-64a5-44c0-b169-f451346c46e0', 'ffa4985f-9898-4521-bcd7-3929d665ad9a',
'36383022-5f3b-4068-84be-4e8ca520f513', 'eb01d841-1dfd-45d9-b4ee-8854fce0db3c',
'f1dccdf4-f735-482d-897b-f9b2dec78106', '2a90a57d-9d2d-40b9-8699-24d6b89f3b59',
'723b03c9-b13a-45b1-8f91-d9ee70728c97', '9f312b85-2add-4fb4-8063-d27980a5b199',
'dfd2bf53-3b1b-4c70-b176-63af6161ab01', '55247692-bf15-4c56-b759-27cc18582493',
'561bce67-91fb-4f1f-aa0e-eaa53bd9526c', 'd2b38d3a-3762-4d84-ae22-99b4967f75f8',
'd38e542b-28fd-4e49-9547-95640defe032', 'df0e86ce-7625-4e8f-932d-0dc969213e56',
'e7afbd56-b23c-43e2-98bb-f1dad0f04aec', '13046783-ec2f-4880-a32d-b22afefe126b',
'dc35ebd4-99a2-44a1-9a7b-d9e44527c6b7', '6fa26e97-14fe-4084-89b7-f3d26a252e1b',
'222f32d1-0e30-42ab-b95a-9e5af7701244', '520ff5b4-8b8d-4d95-bb31-162d04e4b436',
'121d2eca-53c6-4675-b361-53bd21a2da88', 'e8f0b522-2c21-42ac-ae94-f25874aad696',
'498439f1-52ed-4e3d-b07a-fe77061837b5', '0d379df1-8831-4f6a-aec3-7fc824b87474',
'93e988d5-a8a0-4e84-9e2d-943e088166b6', 'bfb0fe7e-a930-4f04-87da-e8d451e0f89f',
'a0afd0ec-f65d-48dc-8613-ec33f2a32b32', '331739cb-6aa3-4689-b002-ff1ba9014532',
'dd8598ce-45ff-4b27-91da-860a151f24a2', 'c2af5917-f6fb-48f6-bffc-7d267e94f59e',
'5a9763b3-0a9c-4068-845f-a2a874e6f464', 'a97ba4d9-7e26-4f60-a43e-1fca8fc20712',
'42ab81f7-dafc-4c0e-89a0-da56d91bd860', 'ac06aafd-3297-462d-8ac8-b0c466fced61',
'1033d7f2-e3de-43ab-a9fe-0b35c2fddc89', 'ab35fca7-05fb-450d-8a18-6aa5103e63ea',
'45c345ee-6aab-4bd5-bd0a-84e43056e5ec', '4784a1aa-43df-4a9f-a7a7-7583b0df055b',
'8309db33-c247-459b-973c-6e0fbecd82cc', 'b31555e4-1620-49b6-99f9-37a9d840d773',
'5a45374e-6e86-4528-b696-9bb2bc1b7f6f', '4eee2aa7-522e-493e-9a77-2b6ccae2652f',
'2ca6ae3e-016e-4675-bbba-7ff0b18e0010', '6f9fd6ad-720a-4a35-b485-08ae7328308d',
'a2ee0dec-31b0-46e4-812b-e6cafb73ae7d', 'a95223dd-6129-4ee2-8a39-cb45c4aa4ef0',
```

```
'0890ea34-823e-4988-8b99-25f049015956', '21f9d6d3-6ef0-4fa5-b731-163ab3c07420',
'8daf4e74-1d55-4ad1-be3f-6ab5b74fe91d', '632000c1-be8a-41a6-bc7f-1b750c561bd1',
'ae30b311-8105-4627-993d-7f94af70a357', 'f102e3f9-450a-4960-ae70-f7eca1dc04bd',
'91f34f4d-4a1f-4111-bc82-e2c10659169a', '118fd1df-8921-46a8-8327-89b244c3abda',
'fa9a5ca8-3893-4496-aa01-2cdab8cb2a5d', 'f7a854f7-f67b-49a9-9494-b5addeb4c45c',
'b5f81ad9-ca01-4728-8a0c-c14f0edf3e66', 'b36c5bb7-e083-4bcb-8f51-b967296236af',
'2132c86b-394f-430a-851d-d0e94e7b60b8', '233255de-55b8-4a46-800f-172a58016eec',
'783f680e-7307-471f-b738-0943fca15383', '5884b3be-c9c6-4b6b-af71-e516d7e9faab',
'ba654354-5818-4e6d-b027-2532fcd46ea3', 'df3c9a59-f709-4cc3-a65e-046c29dcda49',
'65feba53-2a5c-46bb-8498-8bc3e6952bf4', '20b39560-51fd-42aa-ac0b-d1246932e9ba',
'936f4940-7da9-4894-8569-c24cfe8e2051', '9583261f-72c0-4aa6-8fea-900f882a32da',
'227a5aa8-c711-4f0c-ac9c-1ac5aaa44bc7', 'f93d6fa0-9a50-4473-8c57-3a725de9f1c5',
'a57f1ced-f8f3-4b5e-9c4f-3c0b0ee92ece', 'f376dd4b-3927-4c56-a218-e046955325f0',
'5f556ea0-501f-40ac-b45b-bd8773e4f51d', '2cf121b5-fd1b-4d11-9307-ca4a34ec3fc5',
'68d31ea5-a803-4761-a546-70cbd04098cd', 'd2712640-770c-401d-ac92-80133473ebf0',
'fe5ed975-a962-4f9b-b54a-e3ca3c226f39', '921dc3bc-4f4f-4302-970d-a749d3a6098a',
'08d1152a-7d62-4a8e-a33d-06995b6ebd07', 'c3e8173f-5457-4167-bfd6-912aab1eb709',
'9699118b-4a4f-4a1e-9520-9f7a2de36e32', '89789ce9-151c-4d37-9380-2a588572417d',
'f3edfc68-50ed-4423-81c8-d34ee451d0e4', '5b5b411b-5bc4-4633-87f5-b2e7727546e1',
'32f6565d-6727-4ff7-8353-2003588d9a3c', '3f3385bb-90fd-4d56-9177-d0365e62f472',
'9ee1e9ff-d9f2-49ce-9c82-20fe4a1451bf', 'f8fe6f0a-cefd-4ced-b08f-914c6fc0c29b',
'10feaefc-8588-4edb-8e1f-7576170e165e', '7c7c46c8-ad8f-49ec-8816-68f16cd1d4da',
'aedd642a-ae94-4025-8c1a-d8974ed47ecb', 'b2ba6512-8265-414b-95eb-1032749af4f6',
'0bd09d48-7966-437a-ad8e-8425b6042204', 'fd0bc220-3e6c-48ef-a80f-0f9ec19017b6',
'b2daa577-e75e-49b5-ad36-868405e9bbb9', '9697195f-8bbe-47a7-9d95-69838ec4e490',
'd542cb86-24a8-43c3-a63b-b036123cdc87', 'e022356a-ead7-4246-97de-f2e5743a576e',
'a260d482-7047-4e73-9d42-45957ac80a28', '5eb48945-814b-48fc-9c60-dfbfa4599e68',
'c595a190-0c26-426e-9b67-f723a237b297', 'fec97f25-936e-4120-bc65-0340ec828985',
'e5f11335-6c6b-4cd4-984f-5502d3e1edf4', 'aeb57543-4969-4c89-aace-1c7e04d85196',
'f381a586-fdf2-4384-934f-39e2b4d4160f', '0898f881-b4a9-4915-85f7-9ed01815082c',
'756a56a2-3cee-4fb2-b8a3-4890dfaa064b', 'd8eacc29-7c3a-4d46-8cbb-e059a05e0b99',
'0c1df9fe-9471-4575-ab50-c2abd1b159aa', 'ab46c5af-2906-4f0b-80ae-717137b13226',
'c185e8a2-e284-43a8-9316-99a2f4fcd6d3', '2881b0f5-cee4-4a90-a821-be3ce44f2b1d',
'2b27796c-6e38-4ae0-82a3-980e041df9a5', 'fa52ee71-45de-4c9d-b3f5-50768b6fe45c',
'9f3c832b-3a5a-49dc-b79d-cecc98558ea6', '7acddd4c-343e-4f83-8cd5-4ed68a946b1f',
'eda81893-c3c3-4ac6-a3c1-f677cf6b2c18', '68ee88f1-1fb7-409c-a62d-8ade74471acb',
'93e116e0-452f-4517-9492-4f580e0b03ed', '309caa65-12e8-4d44-b5b0-238e541704aa',
'a4b767cd-5246-4541-af37-4f90814ff50b', '68346f46-65df-4b67-a8ab-d84d4c31786c',
'372fe527-470c-48a3-a95e-fe32cc4bcc15', '2c2f3afd-656d-4cc4-aa22-16deb620f0cd',
'9efcd104-d315-41c1-85f7-1fe5f071f288', '657ae5ca-8fa1-41a5-8956-890932e6975a',
'e243069b-4f1f-4b79-b172-031c28424936', 'f737159e-d947-499c-bac9-7983b2a8e08c',
'd090e338-a2ad-496d-99a0-184b47414fc0', '0008cf8e-57ec-4981-9c57-7dc6f817d215',
'cee6067d-55a6-4469-aaa9-550e93046a52', '3998b99f-e6fb-410c-9adc-00b59d28cee8',
'1365a455-c040-4fbc-b10b-1b5391c4039d', 'd82a30b8-10b0-473f-84fd-bd16609c06bf',
'd7c4c334-3496-48aa-a820-15fa5cb223af', '151fdb8c-ca17-475c-8989-dfb8833ecc08',
'634edd5c-318e-40ec-aeb9-1e863836754d', 'f6bd7891-c126-4f28-bb39-1c9b5bd3949c',
'239ef93d-8cd6-43a2-863c-ca75659b943d', 'b67782c3-fcf5-49c8-b55e-66d7a9100ab3',
'f08f7dbf-7f60-4a05-9f26-598302178ecc', '7279940e-f651-4734-9c30-7c6b97fc7534',
```

```
'fbb53a64-dc32-4901-8f62-97b2112cbe62', '1a06652b-f1d2-4368-804e-40be74925cb0',
'0c29a58f-f55e-43c6-80d1-ce0d5957a89a', 'd0faaa32-77a7-428e-bb8f-cb442e99c2a5',
'a1c51bdb-94ad-4ec5-a0e3-395edf947172', '9dff17dd-5d60-4358-9cc5-2fb48a58ddc6',
'e33e4fc6-7f05-4383-a846-209bff2f5f19', '7eaad34c-e230-42e8-b26e-0f77807825f1',
'675b1053-6ea2-4ce1-989b-9490aeb8f206', '770728cb-3f6c-4230-af5c-19cb9684be16',
'a7c5403f-3e86-40d9-91b6-47426e016033', 'b373ac34-3dfa-4904-8a1f-4af147ff2776',
'7effce15-12ba-469c-b4b4-b529425488db', '5f824051-80db-4441-af9e-19bc0a45884c',
'8608a01b-09e9-487f-a1d4-15309222973a', '124a3d94-49f5-4c2b-8683-112c7aa4fc1b',
'3ac513f9-4e68-4906-bbd6-be0668cbcfa3', '623b2414-336d-44c0-842c-bb3245c68126',
'eaace328-00ca-4d54-bc6a-f918fb4f8e06', '7fcb5137-b2ef-407f-b14f-87c2a82a8714',
'372ca589-52bb-40b1-bef3-5a3f4995a6bc', 'aa87d198-66b1-4b1f-b100-3f24123a33f5',
'bcebc799-1295-435d-befb-732fffe7e8a0', 'c97563be-14c1-4651-987d-8b42a47454ee',
'd7d1d7c6-4bab-4a9e-a490-caafbd2a37f6', 'ab7b1d55-f6e1-47bc-9688-1403b9184ab1',
'4093de4a-28fc-4466-adca-43aa84927ec9', 'a8a71808-96de-4445-9023-80309644900c',
'263147b5-ecdd-4a3a-9953-2336929deb11', '79e56cab-26ac-4773-8b4f-b73f4bd5d947',
'c094bd8c-5b7e-4501-8f99-ea12e8be00ac', '181febbb-0fda-440e-9179-bc924a2cfe0b',
'4f0a2925-bb9c-4c68-bc50-64b03aaddc9e', 'b727e7af-afc8-4d60-86e6-21358c0de17e',
'8f237e9d-52c7-4ec8-ae23-6898070ba539', 'b4009dd3-6042-49e7-b24d-490cd1e66b71',
'52cbeecb-c33e-4d11-95e1-c77a5fd9c3b6', '970ef8d9-c50e-4821-9687-145379af2b9a',
'46e6f1c5-3d47-4cee-a1f7-283012e9e9f0', '52e2e594-d8cc-4576-b5fb-ed47377325f0',
'3820bf83-fa20-4baf-8fc2-326e2e49f650', '35e8ebef-2f2b-4a00-bcae-b5d74f2bfba9',
'261cef55-cda9-4621-96ad-15628559da1f', 'ac09d80e-695f-4f19-ae65-9aca66172ae5',
'b5ae58f7-55cc-46be-a2a7-d414115df1c6', '4dae54a4-ebff-40df-9242-90a6bb24619c',
'53aab193-be49-49e3-9012-a0619496509f', '50c764ac-b386-44d1-a037-8dfc9e6dbae9']

Class 1:
 ['4eba102c-c6bd-49b0-a439-6c402bd66c9b',
'80ac405b-8914-444b-a862-cb02d4c24243', '1f603978-a501-44fd-b391-edfd5ddbbf35',
'ed288478-52a8-4871-bfe8-d8dafdf8a02b', 'baa6ea31-0322-4295-8c8a-d6a4c8f6ef4d',
'80b61ad3-93c8-4484-a381-01c07a3828bf', '545c8bf0-613c-44e3-ac40-d409f646322c',
'0034fa5a-8a0e-44d4-91dd-933588dc14a4', 'd2c6cbb6-8147-44dc-8fe1-4f210d316383',
'7b3e73ed-0d07-41cb-b530-3386c4af8e83', 'de344dfc-cd2d-477b-b293-323879e3f61d',
'c43820b8-ee0f-48b5-953d-700a93f675d3', 'ad194939-6cd1-49d6-a0fe-8da29bd45af7',
'46779bf5-0626-4f2c-9b91-0abe4ee70f1e', '7ed55cfc-965d-448c-8732-27d5d8f13919',
'1f7c0ee4-4933-478f-a269-6f07c7fb9c0f', 'da548417-0122-43a9-b345-673ed4a05caa',
'd3ffcce5-ead8-47f1-add0-7e4838cc73f9', '806e5ba2-b604-405d-aca1-d68637b927b0',
'b8b7aa0f-8faf-4aaf-8b8d-ca3504e067b5', 'dc2c8fc7-31e5-4380-b01e-5415448d23d9',
'3eaad4d4-f686-4c2d-b513-ea31fe7ef0ce', '8d1c318e-8e1f-47ec-95a0-d714633140e3',
'21ed01c3-80c4-4e2b-a422-757af3fa1618', 'ddafad4e-f0ba-424f-be23-9f657489f87a',
'62fe1a3f-a888-4c04-bf19-e073190c8ded', '244a10bf-285b-4be0-9061-b5b89c081e95',
'2f98f017-2db5-42f3-a565-5c0ded839637', 'ce0165af-51d4-463a-9d81-4087ae1eeada',
'c66dbf77-6814-4b14-b63a-748f894ea54d', '266a989f-3de1-4636-9c2c-da4f61c42868',
'fc47fe05-4d39-4787-9837-fcc5b0cb7722', 'b8fdae19-0dea-4015-8ed3-c0626ad3488b',
'1e18e2a9-242a-4609-a453-3d969314b23d', '75b7b34d-10fe-496f-ae56-0df7bf3febb9',
'55a1fa44-0691-4e4d-8ac6-e48509a45a8f', '6a7b4f7b-3526-4c39-b475-38d6ac803497',
'7b902c85-f261-4d57-8a43-b5ee52764b0f', '0c6a525a-1225-466f-a2d2-a321be3490d6',
'333f49f4-b0f9-4887-be06-367c6a18c533', 'dc927d2c-eb3d-4f36-91e6-a4f38883fe47',
'f49e8e76-569d-4104-bfed-6bf4c7710f91', '05cca986-ccf0-4898-ab36-c71ca7abdb10',
```

```
'a0c34c3a-32ac-4f94-96f5-e2b9a2c074fb', '04996835-6671-4e17-a2f0-9742b9d4f66f',
'8ae89189-2e8c-4f0d-9324-4f2b3902970e', '463d6870-3e3f-4030-9103-66f4fc61d787',
'53a14e08-4329-4ae2-ab58-71877e69c87c', 'a7487703-a63e-4398-ba45-eeadd197aad6',
'245752f2-1612-4e32-96ff-67c7dc2cad1c', '45939e3a-bba3-4abe-9cdb-e0b17bd8c9e1',
'59f05d1d-b01a-4004-bb55-e9f8a819b5bf', '4214bea8-b0d8-4487-b287-1738a4418cba',
'84cf379b-8c7b-4ecf-ba0f-2fc9ceb11d33', 'b9aacdd8-d34c-4559-97c8-ed609d186a43',
'b8e9c4e8-6821-47af-88df-afbdb232a935', 'a124212b-959b-4676-9fdc-ab223cf56c58',
'dee078de-312e-47c3-922f-11315601e76c', 'eea65d2a-293e-4fe8-af7a-37fd91d38894',
'fadc1510-5fa8-41a9-8d91-6fc124b572f8', '1a094566-d497-4e80-8a12-07a7061aea70',
'14ae903a-98e4-485c-a07f-e7f86e7b46be', 'f75d524e-a8d1-49fa-9edb-d8c7c4d8625e',
'91674087-2b1b-4fe1-8569-ad6207922522', '7567da55-8cbd-43ab-8275-3473390770c9',
'8096c913-277d-428f-9647-4742ceed60a1', 'e2ece317-cf19-4b5b-9c59-e028df00684d',
'45cc587e-4f3c-4b88-9510-4162bde28a5c', '2bb3d54e-8822-414b-8d44-5d9630e145e8',
'8e7e0fc4-142c-4de4-878d-9daf9d9e1865', '083f6146-9bff-4463-9ce9-cf9bbed31704',
'decc3f90-52df-4738-a89a-78ba5820ff01', 'ec977bcf-140e-4960-b702-39fc474ded54',
'0e91a277-a979-4bf5-9532-987cd58f5f98', '1c0d1237-142a-4d9d-a1f1-ae03f86d1c37',
'749ad832-8c5d-4ca9-9302-11015cf556b7', 'feb42cae-a418-4111-8eb8-ca2337bc8530',
'9ae7ec88-f904-4854-8ad4-af7247be7150', 'ffde8c06-7166-4bf9-b29e-8491a1114411',
'e4d11415-8438-441f-a4e1-5f6a98d04c2e', 'fa9ed9f8-c1d3-4001-9890-d8b22bfbd3d6',
'179602b1-35ae-4168-aae0-ef0f3150f574', '063a1b77-6bac-488d-9180-c6c88ce81288',
'664ef845-5c8d-45a1-89e1-cc469bb0380c', '4d4f7e20-26d4-4323-9bc6-c961485b21e9',
'f1cd9901-a4cf-4594-8321-e25792f62f58', '9aafb8f2-1760-4145-9fbe-e70729961d1c',
'1bbd61c0-c1e4-477c-87b0-f4b3a95016d3', '04090d0e-2860-49f7-8a0f-45c136969cb1',
'50b4e388-6700-49de-b98e-dbd91b28d404', 'da4daa08-ad68-4542-bf76-5114dc917733',
'8aa365d4-191a-40ab-9731-93129af3b105', '602227ba-999f-48f4-8f98-11490586045b',
'6c9eab55-8506-47eb-b85c-d9f1de026e6f', '1ed79a26-26f7-4ce9-a711-a4fe0ea5e285',
'a6908835-83b6-48ee-bcd1-007d558414b5', '7123ad69-cc7d-4caf-94da-ccda1a8a4837',
'8abf1063-e8d4-4630-83e0-b139cfebb37f', '6d71d65f-34cd-430f-9a67-efc5f824a220',
'1ae74e80-7a49-442a-8ed3-e5e5ea407b88', '054dfbd0-c852-4157-b8dd-c23cd99321bd',
'72ca2178-62bb-4144-baa0-0629ac50eb6f', 'eecf661c-2320-4de3-920b-dd62e98eaef4',
'ac077a65-e4c8-4bf1-a8ce-7bef601578d2', '79a3513d-9e24-478b-8d3a-1a82946faa1d',
'1a5ddb27-db27-4324-ac04-9c1a37a13ad3', '14e84f33-e40d-4221-8f9a-8eb9a1aae357',
'fd150484-16b3-4ad6-984b-72442676ae06', '81c88956-f967-4ace-8c0c-2232859411d8',
'a78c4770-a908-4540-a25f-a8ac17ec117a', '070a42aa-f27e-409c-a4f9-ae31df95bdaa',
'7594b6b8-94d1-4675-a112-50a22ac95ce7', '09fb872a-5b0c-47d4-a41b-1b7fc131bcd4',
'0512b7ac-bb49-4ff6-bf7e-8a1ce318f29a', '0fe638c4-27c6-4648-8562-3af24fec776c',
'86f8a3ec-68f7-4227-a3fc-33d0dfc50506', '408e149f-977d-4776-abe4-4d854933fbf5',
'd7b74af0-7024-426f-9011-2167b5a57f5c', '4270be76-02b5-43e0-9bb7-e23e32c8410e',
'7d452e65-31e4-467c-974b-40c09a70719f', '94f0b13f-192b-48d6-8d33-f8669cca1476',
'9f4ee550-6fee-4b20-9813-8371d95e67c6', 'caf07465-aea6-4623-a830-8d17b2c01591',
'ffcc58f1-9a90-481c-9224-8d69af6258c1', 'aea2316c-79b5-4bb2-8672-5b52044ea6e5',
'38b13358-2ed1-48a1-b06a-8eb668a0411b', 'bffbfc61-c7ea-4b47-8f1c-4231f7bf314a',
'f55eedb0-c456-4d53-907e-f9be10a9457b', '87e77b7c-ce22-4b61-9ed6-8d9f23e3ff69',
'086cad2f-1183-4ef8-801f-01facee41189', 'af9048ee-3109-483c-b710-ed0fd8dbe41c',
'22f32178-35c0-4897-856d-b7118295ddd2', '94688b50-291c-4921-91c2-19fe257fe369',
'44f79ff7-b3d1-496f-a7e4-1910ee12e551', '8cdfcbc8-231e-4596-ae86-691eea9e2c8c',
'0b0832d3-8d32-4aec-b7ed-1f7b12aeb28b', '2ac6cb5f-c758-43c4-9d77-3b61831998a2',
'728ca8fb-1753-4804-9351-458b900f1867', '1aa167b8-0ebf-4d43-8853-33c3faf539d6',
```

```
'a4c3364e-4851-4ad2-927b-7a07b475cd61', 'e4a6229f-e2bb-46da-8b32-504f8bdebec2',
'cbbd8c66-52cb-422f-881d-1732212159f7', '1484748a-9fb8-4787-8b76-2777a613ba0d',
'bf1037e1-b6a0-4d11-b862-f49545956c64', '847b729c-058e-4c9d-9404-b7ebed0da860',
'abf55690-de58-4b4e-964f-8cddcecc0ae6', 'e636bd04-4831-455e-80ca-3af6482f6ae7',
'475b6387-4a04-40d3-b54d-3006b1999194', '547c52bc-5047-400f-9ece-fac85f099cc4',
'c0949ca8-b714-4c07-b5db-a27c60b23b36', 'dbf58b51-b399-4d16-8eb5-eed71850bba4',
'a86094b3-94d2-46c1-8339-544f7891baf8', '28914fcf-1424-498c-8f31-3fee35fe4e1c',
'ab25cc82-e178-4bc0-bed6-83f13bfc3a6d', 'fc10b498-f5f1-4f81-93ff-e7ae9439a375',
'0e5fe515-91a3-452d-a3b0-b728b5e040b5', '406b882a-7d53-4542-ac73-fb7a60d54f05',
'8f517d39-2094-424f-ab91-38b57ef4e149', '0d5f59d7-54db-46e5-866b-96ddf97f69db',
'4aa74406-0efe-46fa-b73a-b392f3caddcc', '85139b67-66be-4a37-bd88-2727f09293e8',
'2212d953-4804-43a8-bf87-cd36b46f2bde', '2147f7ca-1243-47d8-b747-55fe2655d6cd',
'523749af-9f32-4189-afe6-8bd9f6a34bae', 'a9502826-05f5-4551-84b7-4ab459715245',
'f26a4409-8eef-4744-be87-4a3e6d93c180', '56a44741-2e7c-4e2b-8c19-bded06931263',
'ded34146-f135-4adc-a950-144252e65b5a', '1d818a70-3b19-40c8-9d67-0a43a5c5bfeb',
'f1701c4f-47cc-4c41-8f61-bfdf049659cd', '43f6e0d6-bc9d-4d96-a43d-2b5218a03173',
'f8ce8f22-86ee-4526-8280-45e64a06be1f', '7651c7a4-ced3-40df-a9f9-a24f7189894e',
'55133777-47e0-4938-9007-56aaab072f09', '01d7327e-382c-4b56-bbb8-db78ef8dc971',
'cfbb4eab-9bfe-4dd1-842a-f79a0371917a', 'c1c13565-844e-4e6e-b535-29ea3f038a12',
'930ed2a2-fbc6-4ea9-a008-30392b119d5b', '5d3265a4-513a-4750-8ea6-7033e519a4ce',
'3006ebd1-a60c-41d5-aa87-d1d57bd82f42', '1302ad8d-b642-4cfa-81f5-8a0a418c9f67',
'04952773-2718-4912-9b54-c40789f67677', '301c7ab8-073e-4049-b5b7-3721c6988ea8',
'e0adcb6f-89b1-4fe3-aa1d-dd6ac3419ef8', 'a2372c94-367d-4543-a1d0-b0183fde13db',
'26f72615-83ba-4171-8203-c13aaeaffec1', '8f51ce6a-dcb0-415b-a4be-099639f6a167',
'7aef7409-ba93-4e1e-90a7-ac7730c08dec', '04795c7f-d94a-4815-86ba-939e3437f248',
'1ffee701-1e2d-4b54-9544-ad016f3570dc', 'b3b502a2-76ca-4935-98f9-2f6e3dbd99ad',
'6891fa43-40a3-4a2e-93f5-062251b14543', '65b59e77-ae27-4f86-8469-95031d7d42f4',
'abc11dbd-fb83-46a6-aa2e-318251eb8d54', '3b905a77-a85c-4891-8fc4-0b1744eeb04b',
'b757909d-44db-479a-8e0d-1f43221d1b42', '63265f96-9c3c-4ef9-8b2d-a9fe5a2ab67e',
'77344ef1-5778-42a8-a0dd-7156d854258c', '7144a961-b1f5-4889-b53f-209bca4c9efe',
'7448ea30-a61c-491d-918a-d39a9e873e8d', 'cfc7da2b-a942-40f7-b658-4a2b24338c2e',
'd6855d87-0258-4a24-9d81-43b3ac068878', 'd1bed9f1-cd5a-4bc5-818b-6e0660d0b2ef',
'cf9af3d9-5d9f-4fd6-b4ba-05877db5c236', 'daa7d569-41df-47cd-8b5f-714efa3ebe62',
'8d853d53-dff3-4c64-998a-fcb6846ff74b', '0129efdb-f228-4476-9341-c2e97a6a1243',
'2d0b7c11-f812-4266-96ef-010a1823f85f', '1f925ea2-7faf-4761-a224-b179abbe81c7',
'0def6f93-8cc0-4096-80c0-811f0cbf0a63', '3ee79fc4-07b3-4ead-b043-fc005a289193',
'd0d3abab-6c94-4275-98c6-b708b357244e', '4383382f-fe1c-4ddd-93a7-cff4ec26cfbc',
'a3f26cfc-fd3b-4f46-b131-5e8d711dbab7', 'fd9d9df2-1b43-469b-bc1c-4d28b9b9575c',
'9f42abb6-00d3-4346-ae30-db6c953245ac', '9c079345-f6a8-47ed-9b65-55c26dc42708',
'efb6abd2-4a61-43b7-a9bb-4cea0c3f9ff3', '30b346d9-29aa-4da3-ad7d-bc47226c2ef7',
'b927b3a7-3f59-4d67-9bc1-b625da633cbb', '08cb1e70-8bb2-4c74-97a8-691478e22f81',
'4e4bbdab-f491-4878-9bca-6aadd687a788', '3e5537b0-585f-4493-97a4-d7a2ffb2e8bc',
'ecf4659b-65f5-4940-9714-492003a11de6', 'a24686b6-adea-4f1d-ab98-26c75877fac2',
'317aabde-a3d6-4e73-951d-40645fb8d89f', '93607967-2f0c-4186-b987-2f4bb8afaf5a',
'9f00bfd9-3442-438a-a167-2aebfe3bed85', '96746e44-497f-4a54-a3f4-8b960d585ecf',
'1cc0d1fa-1307-406e-9b60-c2533520c985', '2c1fc44c-8311-4f90-9c6e-1a9ecc886769',
'8d2548dd-5005-43e6-afc7-e856d0394984', '431554da-4383-4feb-ba62-3ebd40c0d3e8',
'814e7cd9-c959-480f-873b-db0e7a26bf81', 'c1cd3f06-4428-499e-95fc-367d48b5c82f',
```

```
'3a23a9b8-c87e-4b67-9e49-128a6104a5c4', '53a7748d-d67a-4496-984f-2f8a5e034216',
'f379cfda-e80e-4804-9735-1d300f443967', '9d87219c-714e-4342-856b-1c2cb4a51086',
'463f0c46-27de-4337-9024-c7bf2d1ce3d7', '494602f7-ba3b-4157-a098-84c550924149',
'b2ade04f-3a4c-4685-a57e-bc0c055f3504', '06caca2d-9d15-4a41-8c91-74b99bab10b5',
'3adeb75e-16c5-412d-8475-64db2e3ff480', '356d487c-365c-4a15-b964-7588e56ce7bb',
'c528efae-7061-46b7-855a-a33a0b121e9d', '3f0d8bf9-ad9a-42c1-a994-ef76fa4109c6',
'451ce760-ddbd-411a-9808-be504354fe77', '1a84101f-78d0-4f2e-bb24-e4bf1dc3afbd',
'862977fa-2ad8-4bfc-952b-2c6666e5dba3', 'c7e7f45c-e70a-489a-a454-43dd0b20a50b',
'237d01c6-d946-4eb0-a2ee-1bc7bd57e13e', 'c2e902b8-3163-47e5-8cd5-8a4bdcf2856c',
'15bd32f5-28e2-42bd-b09d-2b564dfe4d15', '21f11dbc-2e7f-40de-9191-c55b008a8656',
'ada6bd1e-511a-44d8-849c-c61fa9e4eee2', 'faa870aa-bd6e-4472-849f-8eaebcda9f0e',
'5cfe509f-41d3-44a0-8825-7014fae4574f', '445367d3-d403-450a-b18d-ccc9a23335d5',
'16f334df-db40-4e01-af2e-fbef18654326', '6e09064a-4789-4fb2-a4ce-d90071262bed',
'6520a8db-9806-4f33-95c2-3c6475e5b104', '5d008f90-485c-4683-811a-cdb1a85bf6d9',
'98e8767d-4ba4-4ea3-a217-e0bd5ee21129', '926a62e6-9698-4e22-998e-cffd93df5b93',
'e91d180a-468f-4567-821e-ce29ce6f4836', 'c62c53b3-1841-4034-82f4-a8c4d307d9e9',
'766d5bdb-7883-4d74-a976-003153d93c9b', 'f4cea870-aac8-48ab-b2a5-359038330231',
'38637863-8a3e-4a73-83de-10460853f4a2', '3dcd7081-bc16-4023-9d69-0dcd298ab094',
'4839e630-3911-43d0-bc23-3c2acf63c599', '97bf508f-6573-4ee4-a9ea-ead4c586292b',
'c96df637-02ed-4ece-9810-0717d6258e82', '3bfb18ab-c56e-42dd-b9d3-ac77026e0215',
'84d22f43-c74b-48ad-8cbb-6a43f68e3938', 'b284c6ad-7d54-4311-969d-1dadcfc4698f',
'a68aca49-359b-4f0f-b2a8-ba8e7e4bc8a8', '824abc12-e8d1-478b-94bf-576ba68a6159',
'ba2ee020-e8de-4258-b6c4-72e430f7cf93', '5223b76a-9299-4851-8cfd-9bad43fdf1c9',
'2bc4f763-390f-4fd0-985d-8b3bb8ba9296', 'a29806a8-fa15-4c64-a75d-05de6798d53f',
'0edc4003-a688-4f80-924c-d45d7b74dbac', '824980a7-28ba-4101-a66e-dc1578d427c1',
'b482e7d0-0914-4d90-b4ef-b5f8e5e1db49', '2096ba6e-528c-4b9f-9dfc-46d8a46bf045',
'ed251c4a-2b82-44bb-bea5-d254cf641959', 'c9e3c7fb-e163-4f1f-aada-ba8500b98b57',
'6f05cc06-37cd-4457-bf1a-acb640b9ff00', 'f52a46aa-5353-4bdb-a501-0a785df51b2b',
'b4efaff1-3316-44a4-9baf-3d4687af8726', 'df731fb9-83a1-4aaa-9d52-107a00b9c773',
'e49e17cf-d850-49cd-851e-d131503e38ba', '58dce8ba-9283-48c8-bfa5-c45bd10360f1',
'b38b7e31-7413-4831-99d1-53e0446223f3', '266fd922-decc-4260-bec7-993215977c68',
'3b5d125d-7356-4fe1-a760-23f83a8a2697', '32db888d-f0ab-43fb-945e-be5fe993fd11',
'0d06ee39-6538-48b5-ad53-6353650c2778', '8eaf028d-5817-46e9-ac62-5412063528fa',
'd260eee7-20a5-4e97-b0c3-f3f1a1138604', '4f4a5217-d03e-452c-90d3-96b8ceb7d4fd',
'52847a62-3bdf-4ac5-b0b6-ce44cbc08b01', '09bc53f4-fc94-492f-8704-3dbefaae661c',
'58bdb94a-a960-466f-9895-97833c709dcb', 'b163501d-43cf-417b-bd7b-e63af7d82837',
'43e9ac76-0ede-4b49-b2b8-702d8ef86751', '8313b893-141a-4df9-8370-f0222011396c',
'0594dff9-a1fe-4579-901f-98b771abcbd9', '4d6bc960-67ec-4e46-9f3c-3e780051fa2a',
'4ccb358f-c231-4c64-a5e1-39ec5e17cabf', '109f63e6-5a67-4c2c-810c-faf423c5da81',
'b7cb6e55-1d18-4463-99ff-f868690107e1', '7b9e9813-7981-4eb9-9887-ccdeae873b74',
'de91126e-5131-4e38-a9ef-4efe506e1ff4', 'a909a3a4-0760-43c2-b38f-45cad807ea2b',
'e774adf1-5ad1-4253-ad83-8f082115a89f', '82f2891a-6018-4438-a3f7-d1eb3337391f',
'973e4e1c-9baa-44c5-aefb-47354655b823', '631d7950-08e7-4782-a61f-ee0263b1f804',
'2900abf4-e99d-491b-8547-a7c4b6e8f58d', '29ce3ee5-694c-4667-af24-5ae1fd27918a',
'228a55d9-90c5-4c87-ab4d-287806618ca3', 'fecb50ba-5999-46c8-9f8f-1d6126a3cc66',
'0b05d091-ec01-4e0a-b11c-087f5d0324c2', 'e7ac3f43-6bd4-4987-b7c9-15519101bb74',
'243dd7e5-ff05-4c97-8eae-50e7cf50ea56', '968a791f-e5af-40ff-bd3e-334aadd5308c',
'd88ee47b-ec16-4135-8593-a91563e9d5c7', 'b1cf1ad4-1649-490c-961d-72fb8a185a7f',
```

```
'b1712f65-beef-432a-9118-d2001e59afc9', 'd3d83861-6c64-4939-8c6e-86850434ff63',
'10276932-b26c-43b7-bc16-d2fb1ce9a362', '656a9cb2-7392-4bb5-bef2-61b495df0a1c',
'f8736d38-bcae-4948-9e39-0a94ba5103e5', 'b185bbeb-6f58-47d3-9611-a29af81dc979',
'7dddd8ea-4fff-4b18-bebf-dbb706566c32', 'cf4aee4f-d24a-40aa-9cd5-5ad6d8651418',
'd34e408d-0c1c-4213-ab0e-e206cb2be20d', 'cb86cf07-5e94-45ba-bcbe-27adc59a06a4',
'd163c663-c055-42b8-a6fe-ad1d9d66bab5', '50a2e6f7-1b9b-48b7-9ebf-b5019eec4dc9',
'1a52586d-c22e-428b-b12c-5d304549c889', '17291f22-6f49-4440-8466-acd3248b6c85',
'6bc01adb-6b4b-4c31-ab40-8eec8e3b2124', 'c43a0ded-e15d-402e-9110-d72655275278',
'e5b872e3-2f49-4aa2-bffc-5ef43979dfb6', '5115115e-aa68-4204-b41a-a7b41fec17b5',
'de74e45d-da37-454f-85d6-4542a924d7ef', 'fc61675e-d490-4c3a-892a-fadeeb76b18c',
'492b850c-5f4b-4dd4-8d59-3ae6c1bed914', '6d71fb2e-c33a-4693-aed7-c0b5a36c3f81',
'd3626a64-fac3-4ef4-8b28-567539572a13', 'b688fee6-7632-4055-82c9-a951fc7b6fc9',
'5f231b05-158d-4602-9fc0-f1fe6a7e3397', '03d28ab1-ac10-420a-ad4c-c0b51f3a875c',
'505e09b7-e9e1-488a-a58c-21392e74dee7', '2e1f0271-adb6-4e99-82ef-14e3b9d2d85d',
'b3185e3d-0dfc-4058-8dcc-b40616263c4d', 'e03b6f9a-be35-4a92-98fd-f44413e1da67',
'6fcd7620-d097-474d-8d53-153425cde3fc', '277ff66a-7478-4b8c-a547-bebd3e0ae21b',
'238ab097-b7a9-40dd-9ea9-1595f8371f2d', 'd27cfff2-a887-451c-b367-b1d517c6529f',
'996035ce-72e9-4ef9-91ef-fc96578f35fb', '486369a6-f628-418e-8181-bcc8ce93a3df',
'565e8f82-03fb-4937-9e13-4684a5f85005', '53997ec0-acf7-4b47-87d4-90ca82d8713d',
'93690a75-9eeb-468f-89ad-95b1bfd646dc', 'd7b06fd7-0104-4d1e-b5fe-5793e8c64a08',
'c206719c-efbf-4ce6-9ffb-512df7b23685', '8ee9c4d9-d331-43eb-b4b5-1ac78173bda2',
'86d8f825-8dfd-4c9c-957a-3e3a2ab5a131', 'a09a339d-395e-4792-8095-c35246a16351',
'67a60640-6046-4e54-ba1f-a6ddaae3b4c4', '4b55db69-bf82-4cdf-ba12-9538f7047516',
'918ccf76-62b7-4c46-80d5-0da49ee40c3d', '47e1a4d3-3a5d-4b83-b427-05bb16fa94c7',
'7141b4a8-247c-4b19-967f-14f3d0638eb4', 'c3848b5d-a363-483a-9352-b0584fa79208',
'7d56ae91-6f3c-4369-a1f3-bbde98d4da82', '20de7c58-b721-40ea-933b-999f6d396d3a',
'49c035f1-c030-4436-ad5b-ecc789519fb0', '7c8ed6b6-1d5f-4a9c-bcc7-3b992fb3d61d',
'338d91fe-b2df-4b12-9186-a1b5ac28404a', '5dda1c4a-8b66-440b-896c-8c0be38645b9',
'e5b4f882-7dca-4e63-a665-f412cb383e39', '8a2fbec7-0a90-4f49-9ce5-3cba892c15cd',
'a9913ada-caab-44d0-9b8c-a9b6f0f25ba1', '1f058f69-c399-469c-8d2c-c0e3db5d9f68',
'882dc1a9-dea7-4d6e-b6ca-0020e9b15548', '0b6025a5-458b-41d6-947a-e9fe52762734',
'b9014d2a-f1e2-4f04-9481-5803ba68dd29', '18a93dd5-5855-4d19-881f-084d40c12ed4',
'8b11ed0d-316f-421f-9b8e-da5e1ed71b82', 'df231657-f352-499b-98e2-28f93e52b847',
'd150bc61-7c75-4051-b85e-ac370c7a5cf9', 'f38b84d8-9924-4d1b-b7e6-31e679c99473',
'257f2827-69b4-47dd-94f4-503f0020a9e0', 'afc30314-bbd6-429d-9bde-f192dc7b7a7a',
'060213b6-2c02-4e99-bfea-e2e736e3a023', '9d827fea-d886-46bd-b0f6-a1ca65646b76',
'f1f26f4b-647a-41b7-a480-6382f3ea67c2', 'b982b218-56c6-4d00-bd9d-3388e2b82410',
'982d8816-eb43-4efd-a740-8a39789ebc6f', '11c3bed8-5c5e-4762-abc3-3ffd6979f2ab']

Class 2:
 ['4a3d10fb-bc98-45b4-8137-6719c91bc4b7',
'8572bf73-187d-43c4-9063-450e57cbab06', 'c11f674b-ebef-4de2-bf96-a25aadd2346d',
'05658561-9358-427c-86ca-0bdb8a78c6f2', '2f6db0be-adf7-43f2-9000-306078e73737',
'e20c1e3d-751f-445c-8f2a-73076ebb168d', 'eb16d48d-365e-4ef8-acc0-0cb0ddb6b605',
'9d0b18fd-b395-4be5-a95b-ea5f18f15b3c', 'd5924d73-a6ba-4fd2-952f-27d31e1b2fc7',
'8f8d4562-ceb9-4557-a65f-0835a9a6f891', '750e5dc6-b801-492d-8b81-0c471a3dd5bd',
'f34576cb-7df6-434c-858b-b892645ad81f', 'f077f4c0-c773-4bb1-8f70-d06c9125546e',
'e5789a70-db24-483e-8d3f-85002011d954', '8a740d43-5b24-4c03-9764-f57b4f129473',
```

```
'5fbac5f3-ad00-4d9d-8cf6-25650bd409b5', 'fffa2a6c-cff8-455d-9f65-ee0319dfef0f',
'cb63c80e-62cc-4c98-9ccd-011a135531ee', '750d3556-2cfd-474f-bd29-c1ddf2e7fa79',
'f983802a-3b01-484c-9249-5826c7af340f', 'dfdc4459-1d67-4604-8542-7002b03f5ad1',
'635ea47d-b881-42bf-bda8-51a9a8d85aae', 'cbca6e97-1612-47ab-a73b-de6d5ec910fe',
'e9674aaa-192c-4cdb-899e-5278c9b74d70', '7bceb0fc-1e85-4b31-86c7-7fa6321a4be0',
'85eb7099-c70e-44e5-b65d-3241979d268f', 'e311a62d-b3b7-4a64-888d-5bad1455de5a',
'7f6015be-8001-453f-924a-da5531619aa0', '661cbbc4-1a61-4c95-bc89-ab4368d4be9d',
'dea9b0a5-dbe9-4cd9-89b2-f1ae3bac3c71', '11749210-6710-412f-8151-ea6c79ede607',
'375a1b87-40dd-4f09-b3e1-ab6a30f4e177', '514b5cc2-d7dc-41e7-b8f1-36122f6c90a0',
'da9526b9-2633-41fd-838a-e0c2790db555', 'b8df0451-ccce-4106-b396-f43f0f1ff4d0',
'81da48f5-98e6-4912-a315-0932e2f1a9d7', 'cecdf65a-fce3-4c55-87f7-5e168de6be02',
'5ee7687b-e640-43ee-8f53-4f606d0f9764', 'fa845565-1f49-4d27-bd17-fe185638cf66',
'54f35495-3df3-4f3e-b123-ccbda44982fa', 'dc77f09a-6938-4f68-ad8d-9554b842aec7',
'e7c3db48-86a8-4372-8174-56b8cefdb4c1', '72ffcd35-638b-4a8d-bb31-49ce7bd67358',
'a0d1f259-3994-47ad-88ea-4b2de5904b46', 'bfdd4e40-b024-4f60-93ff-8d24a6381b97',
'f1f0f11d-d1e0-4b7c-99c5-6f3d09fea5ac', '9df223db-1c93-4688-aa23-710a472bc3fc',
'3914be33-6188-4dae-a87a-b761c4518e70', 'd18c96e6-71d1-45a5-80a5-aa783227b663',
'770d8441-a264-48f2-a0eb-a2e3d6c37658', '5b22622e-6a32-4972-84ab-41e3df0cf15b',
'9595f5f2-633e-4e0a-a30b-68d8453638df', '17183013-21f7-4f3b-afec-4e1f30cff57e',
'25e363c1-3bbf-4940-a794-fe59eca48706', '5af8df1e-f842-4907-b2c4-297459c36917',
'2aeb0f71-6f8f-4645-bf39-6cfaf92ae884', '347d5caa-abe5-4bcf-b126-bcb91ab86604',
'7bd6dfcb-51c2-427b-8a0b-0b7115c2cde4', 'c6026329-4f6f-4241-ac1c-4ef6609c4a0d',
'a3a02754-8b7c-41dd-a96f-38699741f8e8', 'ff6316d0-140c-4b5e-a2f7-b21e23903a9a',
'9d7b9fe9-73bc-495d-a830-5e68e22f70bc', '9b597503-ecde-4836-9001-c57bcc65771d',
'e472b793-d22b-4b1b-ad44-b1caa0719c72', 'b803bcf9-df72-4555-8249-655fedf8400b',
'fc54e782-7b36-4bd2-90c3-ab2d33877c90', '22e27edc-f7e2-4911-a207-036d21ae1fc5',
'6d872185-1ff4-4560-9ec4-fbc71000b241', '399b67a2-e558-4ed9-8148-d674eaeac89c',
'be2e3db2-83bf-4bc7-aaf5-49a0f20b6e5f', '8517e698-e665-4d1b-b6c0-b88b0fdf82a5',
'482d10c0-2323-487d-abd4-ecfcd6eada62', '850303ee-0094-4aad-891f-2970bf22f343',
'f169df9b-8733-4399-920d-82e23b11258a', 'cb09110e-5302-423a-bd07-87fb21e2373f',
'77a3ea0a-3ce4-477c-b0b0-0e32bf302bb7', '76106fc5-66fd-41ac-93d3-32abcd75d1cd',
'c0abebdf-006b-4f3f-9a08-b50d2ec4a004', '9ba1542e-be9a-42c1-a062-16403f3bfa73',
'354a8adc-3be3-48e5-824b-2a35faaa96e4', 'b3762897-0db4-4475-a445-c527e414a848',
'0e958f97-1a7b-447b-bf4a-fbef9d0e74f6', '5837c4ce-581a-4e64-8555-100366458292',
'21c92bee-0925-4f2c-9618-1cc93453d257', '3df26dca-2ba8-4e6c-83c1-db9e0b174ce5',
'0a4c7c0f-198f-4a87-bc7d-4dae99a65b37', 'f92a54b6-7f99-4db1-944a-e5b5fcc8bd35',
'111422b7-7cca-458f-a02d-5ca4bb026f25', '6094db88-d968-4325-acc7-9e5c75261712',
'a4806d25-4ac9-4ac6-9361-2e1eb26689b2', '893c14f5-dfc0-4e21-87fa-53bfaa670593',
'3ec43226-4165-41e7-8ed7-d7595d50214c', '59085fbb-5ff4-425a-ab08-c4e58df6d5a7',
'a61de326-1b70-43b8-b11d-840121c02c37', 'bd20ff8b-1e3f-46de-8a09-bdc3b4f2c555',
'4fcffdd2-aded-4619-a5b2-e988d575d6a8', 'a7298534-96ce-4611-92d7-b2b9fbd21105',
'3cef3fd4-984e-481b-89da-c0692bcc529b', '77da97cd-22eb-43b5-b95b-eb4b0bd61066',
'96291410-cdf8-4ecb-96ab-450827f14d56', 'e3d154c3-4580-4186-b712-35bf7fc4d7c6',
'cdfc00f5-8d0d-4ac8-a95f-e485182a1e0e', 'cba5d1cd-0ff6-439c-b24c-e99db971e895',
'156d5f2d-c4e9-4875-9d23-2276c8a9c6d0', 'b1f47a92-bc2a-4603-8b68-8da11e046237',
'5907970d-944e-4737-a2a0-f8650bf1eb67', 'a8ae30db-4d12-4ed4-90c3-3c0621ccd82b',
'523e3250-5f69-46ee-b2af-7fc3cced1541', 'efddb9c0-29b5-407f-aede-f59391168234',
'9861628c-f85a-4657-88f9-3395b2a76eb3', '1cb356f8-feea-4a42-b327-216ce35d9606',
```

```
'6183920c-4ca9-421f-9275-412de35698f4', '5d29c348-98cb-4bd9-a56d-82bc1ebdd133',
'0e166d60-061a-4846-8443-ebf989af429c', '61abffe3-ce00-4b04-a3c4-4e389d9426fd',
'669a2bce-7233-41b0-b10b-1bd1f7a90a5a', '6293182e-0510-470a-826e-de2c52e2e05c',
'c870a64b-8236-4e1c-ab30-d3df060f81ec', 'c6e5e39d-e180-44bd-af8f-0fca86ff0a55',
'91f1db79-1f80-4956-8186-ad383055ec17', 'bc572d73-84f0-4f94-9c9c-7adc82875a4e',
'd98c1170-5f2a-446d-aa26-5beef1f054cf', '625929e6-ac4b-4058-8412-5f4323748a81',
'a8ebeeaa-456d-4c52-87b6-af933bb0f996', '2bee33e8-bf9e-4c12-95e8-c5f66a4d7ab3',
'3140f70e-f02e-49df-adbe-c20e62465b78', '7601a102-fda9-4754-890e-c237411c61f3',
'25807ad0-a62b-4257-896d-c2f2fcf42f8c', '1c236aae-e016-4fbb-a073-2c32492029e5',
'1d06e551-3f84-4601-a372-46d0cec645f9', '6b510abe-99c5-4c65-8652-10114ab8f7b4',
'd742e162-4fe3-4b66-8d58-6f60264b1a33', '96644236-b19d-425d-b6df-312afac95e2d',
'588e2fe5-b9ae-45dc-811f-69c8443db852', '134b043d-0236-48de-b795-537998327a64',
'061aea01-2466-4574-ba26-18ab57169e9a', 'ab70fd37-4109-4989-9b34-b5ea9197bd33',
'c6397905-eb08-4a88-9258-0e4b03af2ee0', '353be482-19d6-4468-98a3-1e36654e2890',
'a6adfc80-6189-47b1-a964-1e25ae8a7b6f', 'abc87435-b3d7-41a4-8f42-5a2f1a44b053',
'422799fd-a1e0-4069-a962-cf0861397649', 'f2089f27-b25f-4b30-9d60-135972f329af',
'896d19f9-5b78-4fec-878e-5404d07f567f', '2c1749d9-bdd2-439e-9342-3b7e499c8ed0',
'f342ff8b-54b7-4a0e-970e-7057b0cd6285', '74829301-398c-4dfd-92d3-a9879252cc8b',
'd3f8120d-dc3c-414c-b580-2345777431da', '978a4ca3-eccc-42fc-9990-3d6bfdeeeb42',
'445cad2e-6a07-4eb9-bae3-5d8e2e76d9c1', '52439b1a-e6ac-4d0b-a1e2-ef6b6b93efcd',
'1f3caa13-1896-4f2c-8fff-206e1c9d0c15', 'ded98d74-207b-4c0c-8e64-ae5372b620bb',
'647a49a3-58b7-4e06-870f-94d8cf28d333', '431ce957-c6a1-4712-a2f1-2accc2dc487a',
'93a663f2-24ee-4677-8a0a-2b1388bf505a', 'b10aaee0-52e8-48bb-b6ab-0cf560eb231a',
'208d298e-bf7e-4d1b-8bc1-17550803eac6', 'bf4cba45-9202-4e0e-bec8-55b21224cd07',
'40b6868e-aa74-4524-90bc-91ef86ca65ff', '52a089cb-f557-4a22-bbbe-7a61ce2e0145',
'e4ca5650-d29e-46d4-a237-efd20f656a53', '54d1de1c-0249-4813-b28b-4bc62b268161',
'8b2d0e86-5e1d-4787-8e6b-0897ffed0f59', '05626233-d502-410b-a69e-8b281d9dc242',
'cb3950fc-47cc-4e3c-8657-563e697d7a56', 'ea793247-64a8-4e0e-83a7-529ef39beaf0',
'914ac037-064f-4898-8eaa-d7ea63e8eae3', 'c630da83-9f02-45a3-8a02-19fc0b186b95',
'db123743-e29a-4862-81d4-a753557cc613', 'c8793ebc-9d1e-4d44-8d1d-b9c96ea75109',
'653dce63-9309-46ac-ae30-806148e30f93']

Class 3:
 ['24736b69-427b-44e8-86d2-f8beb56aa43b', '9f2399a0-fcac-4776-86cf-
caf8c6bf4a19', 'ce4af93f-6296-4f70-b42c-0c8146b090b5',
'd1e338cb-8e4d-45f4-a7a6-41da51e83e19', 'a1e6b473-b107-47b9-bab8-13d037fe0250',
'c3fa24f2-b8e9-481f-80bb-2309ae2ff091', '9575b516-263e-4875-a165-8a51e50ea996',
'5fdd4159-e99c-4b01-ad19-3f13653498e9', 'be72f1b2-323b-433e-acfd-c33065081584',
'26026284-f691-407c-b677-da2555a30396', 'ea06c408-ecd3-4180-85da-0fcbaa6b4abb',
'a8fdf07d-f3d8-40ca-bcfc-cbf494a80e9d', '67034269-49fc-495e-8840-03cd3a5daa3e',
'985952ed-ed10-48ca-b789-fa32a96b9ca1', '10d2a083-8227-4088-a2c9-f9ba836335af',
'654a697c-d4b7-41e6-a53d-bf6c7e106c7b', '62180733-8efa-4716-8232-1e43f604ec11',
'82f9eedc-cb2a-4c33-a2e2-4e6f34d0735b', '1f87f4f6-3169-49d8-84d4-8e2e84ddcb8f',
'30c00924-3bba-47a6-9700-f23065aef3db', '7ff60783-190e-4bcb-a78a-0ba20f469e3e',
'ce9552a7-f1b5-4b85-964f-931743337f46', '17d1bf44-59b7-4812-9e1e-adba7ebd06b0',
'577ccd53-7b5e-4801-ba31-c53c7ef4be13', 'ff325ba7-cf3c-41b4-93c6-e7912cbc15fe',
'ac3458d1-6fbb-4c4a-a0b0-87711cebd36c', '4c824be8-c9c5-4cc0-a0b7-beab5fb0d995',
'aa6bacff-61bd-478a-a56d-869276a0f551', '1b741056-8c1a-4804-bf46-507b5f97e647',
```

```
'7d455348-4caf-4db5-89db-3659160cb156', '7f87bfe9-ba63-4f1b-84f5-6bc5a6c14755',
'5ab152a1-d73f-42a2-8cf3-458076418643', 'ffb02b56-f859-4982-8a79-a49f2f44afca',
'a4f11e85-ef5e-4c16-b318-a1e668e57164', '306b3644-292c-4f47-8ef4-17c4222ab05d',
'3c504df8-1220-4267-8e75-dbbcf1c42f3d', 'fa864e64-3f28-4241-9675-384af86869dd',
'4259e76f-195b-488e-8bf2-4c4c42f0b76f', '91c5fd6f-214c-44c1-be6c-96e3dc9eb0a4',
'b6030695-2362-4a6d-9447-0b4fd7896e1a', 'd4cc1916-71f9-456a-9324-eebae4203868',
'c87ee734-60df-4ba4-9420-814bbea50253', 'b2a1840a-c206-4859-88df-009dcb99abfe',
'5a3714f5-7bd1-4a35-8207-cc0b58b83f5c', '6af780f0-8867-462c-9bf5-bb50249d62d5',
'0e15acfd-c467-44cc-95ae-c49244a1da33', '68f80617-e495-4a50-be9d-96f36ad20906',
'358f9ce0-1766-41d2-9942-b7ab70d4050f', '62701b32-bf53-4963-b0a8-71a7d3a8d0dd',
'f0f213d4-83a4-46c0-a6b7-f93030217e98', '2b22e1c9-63a2-45b8-b035-58889b8f1b5f',
'1ba0f700-ff09-4c2c-a09f-b8a9ddc2f467', '2c088b95-126e-46d5-90e7-612a61e9e6ea',
'20979c65-85ec-420c-af07-93656b1e19fb', '42e15a8e-50c4-4d8d-a5d8-b50d0c31b239',
'637053d1-b11d-4203-9eb8-4ad9d63ec73c', '8528754e-c324-446f-b57d-2ba37791678c',
'17fc2fcd-55d4-4342-a553-302e56d6d816', 'b982174c-8ce8-44d3-960e-7ab4f45097f2',
'6599ad3d-d607-4e7a-94bc-bf8a52272a91', '69083437-c5b6-4768-9206-6c1f54098bbd',
'713d8dcd-fd3e-4017-afc2-5b28b8fc1e04', '109b4199-c83a-46b6-bd7a-c5b66e9e54cf',
'e740ed1b-fee6-4033-b7a8-735f58f9d3e1', '22cc00d3-7295-4a76-9639-9bfa54b0b5eb',
'376aa71c-c2ea-4053-b4a3-c8c3532f3402', '85b23a55-4111-4ee1-90ed-0bc15b4eed77',
'2d398158-2fec-44d5-ba7b-12af7b7a7379', '03c958f8-3c69-44d0-8be8-593213ee0c14',
'1a4c523a-f7ea-4b26-a0d2-8ba0a729a5f2', 'a91e689d-666c-4395-933e-1ba7766dc8fe',
'96815f86-6c23-45a7-8b33-0d2eb4347e2f', '29ef89f5-1387-4710-884a-b39a7532e733',
'f3a2db37-c46d-4e8a-a1fc-c515e68788a3', '68b1cee5-0a01-4c38-a070-090e0c9f1018',
'53db51d2-1d70-4b6e-86b5-05ca362caf74', '6e5dd33f-2642-4448-9065-1e6d8d4401f2',
'8a6b1fb4-bdc8-4d03-9751-51bdb9287b6d', '48a4f6e3-199a-4e40-8f51-3765473290e5',
'ebbddb7d-83fb-44ad-b521-60c89c946315', 'ce7ad4a8-35d7-4085-9a55-d5eee8b28002',
'f79700c3-8930-46b9-af77-2df05999b847', '633648b2-df91-4def-8036-47a42b01fcd1',
'34d69f71-347a-4c2e-bad1-3800f147451a', '193ff09c-828f-4b0d-90d7-bf75137e4cc8',
'009a9aad-ea72-4724-bd8e-d44be1f727c1', '251ffb28-ff64-4ca4-9fc4-bbfe4ecf6084',
'3e257b88-fe0b-4962-a19c-7cd3f20161d2', '4d7b321e-6b6d-4fd8-bd0b-e94d83c06240',
'f9bfcdff-2cd9-4b61-b633-4dab7e0ecf20', '4d8f4978-b592-4f54-98e2-3d0523a727f0',
'a82bbfa3-eca3-4d4d-a8b8-51e7f71bc4c8', 'ff0ca5d6-6c16-4a72-af15-f9b84ba31032',
'db704e23-1f92-47ab-a675-19c783fe4977', '92a1320a-4a17-470b-ba04-3844dbc52618',
'890b4e5f-f382-4da2-b98a-d2a7eaf03a61', '18ef0bc8-c415-4bde-aeff-91e020116e1f',
'19ced23c-00ef-4239-b801-41936bd1faa3', 'b45ed235-6017-46b0-bb1b-e47672cf9f5e',
'0a6d5c06-cad3-4af2-b345-7f32401aa96c', 'f7e8eb11-cb98-4bf3-9021-7c143018b387',
'959279ec-8f2a-48dd-9505-12f09fbed1b1', '502080b9-a7ac-4ec0-95df-9f857c013103',
'41730565-9335-4b37-9f74-d167a63d7a00', '86fe84e3-b42a-47d7-bb8b-b9e67827b9e5',
'68cdf714-9034-4e4c-bb8b-8e478bd9116b', '294bf390-a0ae-4f2e-b70d-a12fd3746dc8',
'6e5f164a-475d-427d-8f8c-4517eb193b35', 'c00fb811-b8f8-43c2-8e7a-ddc24b801be8',
'4b821868-7015-4327-a651-50f1fb41c21d', 'e49fd7a9-fbe9-4278-b293-2f7de9a4af38',
'c2dd8a00-0153-4e4e-9089-3dbe8a652fc1', 'f92e231c-9c53-499b-bbdf-c85ef5b5f654',
'60387cf2-b5cc-49da-a076-cdcff45a2c0e', '704d21a0-2112-4b36-9569-0eb66bf73a91',
'88b6c538-3b24-498b-9801-0feef4094eb0', '6819bfc3-a009-4719-9f01-ef6ae54d5dc7',
'624f0ea9-b726-4b49-aa74-9d9ab5a05bf6', '2c8175d9-9007-48c4-9157-9f818a6a7a6b',
'c91c57c2-2a1e-4cee-8a03-bdb1c7fc94ef', '77eb3a2c-fea9-4bd0-b317-74416f55d115',
'f040ce69-a850-4f6b-a165-e08297d32e4f', 'ce6c7fca-4dc1-4ab2-8c61-41f3b4ca06a6',
'09299c3d-ff0b-462f-9f0e-8aea1b0e0613', '5bb7fcb9-d334-4ede-88a2-6009e988fccc',
```

```
'6d849a08-fd41-4125-a72c-a9fc6a39b042', '8c5a7228-e794-4b14-b9cb-91e2156c9c2c',
'0449be43-5ad3-4222-b4ca-2480480c6169', '34e8a100-6a5c-4dd6-a68d-5e173ed07b23',
'23331473-7355-4bb5-9351-eb866d7e96d3', '94fc67fc-901b-436e-83bf-25c38a15bd23',
'272bcc43-3074-4fa8-a5ba-fae9c1e0e7e7', 'e754f8cb-280f-4f11-b0d8-85922121b8e8',
'b0940b38-8909-43a2-a629-123dc6f45597', '0b43490f-c4d7-4eff-9342-9b111c2afef9',
'fbfaa8d2-6a6e-4427-a908-ad61318a7e04', '888ad283-c515-416e-9705-38ea43a24e17',
'8914c6d3-0147-40a4-9921-06637d694083', '767a8763-e48a-47be-ba8e-205775e04c33',
'92411abc-5ed5-4eec-bcb9-c7181dc48aa3', '57dde55e-70af-4071-b609-4343aa3addc3',
'07b2db9f-cdc6-4779-b13e-3c6f98da292d', '901b6894-6fd1-48a3-adf9-61b9513b256e',
'305c7ef6-09bf-408a-8049-acf1b52a833d', '363f2955-b2f1-463f-b6fb-7515b8597866',
'f3304cf8-7a47-4a02-af35-05e1508b768a', '34346ea7-8f9b-484e-88a9-db32822c2640',
'd8b96ff7-7bbd-4bee-90ad-8c6fe8574f89', 'd640b6e2-91cc-4f81-b4c3-756eca589f0f',
'6ce05ffd-383c-48dd-a94d-2deb391b12c5', '55db1a06-273b-4c21-bca9-ad563b564c11',
'7b0291a3-2c29-41eb-833b-66eeee30e50c']

Class 4:
 ['1da7cbc1-a731-4e32-afe6-0b2461e5972c',
'025f714f-c1fa-4100-a4f5-634428b9c6e6', 'a7f3da5c-82ad-4d0c-9b33-8df666c52fd9',
'f263806a-1207-4ccd-89f9-b4f0201ad7da', '4548acd7-7bcd-4376-b59a-45e39bcadde9',
'1b6a4afb-fefc-497f-90a8-528ef68a664e', '69fc662c-667d-4164-9add-8c0a0e8d9fff',
'7ea9eccc-c079-40df-9ca5-2566d47835a1', '6c3ef46b-4c7e-482a-ae0b-cbb128f4ef83',
'de75b429-4a91-4f14-9b75-f2bb54dc0f0c', '2075a62a-50d9-44c8-ba44-9290d5b0a8e7',
'eb45cdcc-1927-4cb8-8380-d6635bb816e3', 'd35dc90b-0a51-4085-935e-ca74e8d2eed1',
'c68e0e6e-98b3-498c-ab5f-4cbf43667142', '4a0935e0-da5a-4d10-91fd-2fc6309a678e',
'ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd', '5eb23031-a7ee-4fcb-8c44-5b8b72f1b53e',
'dd11e12f-84e6-4f9d-b645-99b9f49ae15d', '137db9ab-dc9f-48e8-b001-76abbb91e1ec',
'1a34d9c4-d2db-4e8b-926d-733f638961be', '479aea8b-bf97-4bfd-b91c-c46321d17cbc',
'e70d289b-867e-45e9-bfdc-c7c417f22093', '9f02f011-0d8b-4731-9813-e070cdbd291f',
'40185ad8-0bde-4791-bdd5-8c3e85c0289a', 'a1e3b7ea-f1b9-46c7-bd1a-62bbd35a416b',
'c183d11c-73ea-494b-b133-60dc19199886', '052e4f0d-fb55-48e5-9c9c-09f5397ee841',
'ebf95f46-d3d5-4cf2-b9a6-024648d8b923', 'b3b13763-8918-481e-8922-f250252d3fe5',
'6df1befa-8809-4058-881a-03fb996091a5', '99a46d8e-e427-4922-b8fe-e043aab6ee6e',
'dc5d14af-66af-444a-aad8-62754b474586', 'b7cc5ee6-eeef-4dcd-9061-0dca340b280c',
'822f392b-b0f8-4830-aaa9-f7af25adfdf7', '45b81165-1dbb-4d9e-a284-50dc73f2a34f',
'9d186454-ddeb-4828-a9c8-8e878f8bdffe', 'dff48d10-52c9-4f41-a4f5-966fe3eae4f4',
'ae260d0f-b521-41e5-8840-da3b32707a2e', '5a660386-e8d4-449c-b7ea-b2a7441b122e',
'9624f54e-3a81-4c7b-a3bb-0650fabcd6ff', '9c18d94d-50e7-4331-acc8-6981d382665b',
'0526fd1e-a272-40f8-a2a9-e878454ef3ac', 'cc7a1e79-1e52-4c3a-ad9a-2c4adb6ae750',
'4d376abf-28b5-40a3-9c3d-fb32372b7492', '40346dbc-0401-4d9c-9a13-74902d8c5bc8',
'56995f06-d288-4e0a-ab78-c2c122aa87e6', '9e18e6bb-2d32-430b-9e9f-142b019a5f6a',
'40c82bad-e845-4392-a468-c8eeee902df8', 'a76ae12f-0346-4367-8651-3adaafa67c3a',
'e5ba6f18-457f-4240-958a-e7272f3142c2', '5f2145a6-ac03-4d5b-a42b-dfb4bfab8da5',
'9c09e92b-c274-42e9-ac54-c20cf554ef15', '1bdafec3-ea8e-4ddc-b7ed-1a5f502eb6fc',
'dddf52a4-78a0-4a8b-96bf-b69726df7434', '33c6a4f8-6a51-476c-b9ee-1f8ede634d71',
'830a8cdc-b4d8-4743-af91-319ff755b510', 'ac6f0099-7b6e-4780-baac-8c7da3a28147',
'5bd3fdcc-8f99-4d3b-b52d-e6f825d6ccdb', '2a8dcb72-b011-495e-a0ff-46c0636f5324',
'7a591735-e9c8-4705-8c4f-ebe26b48dcaa', 'ed39cd26-44d9-41f3-ad59-e7bf3816f38a',
'fab70964-5dd4-40a0-aeab-250e4d6e82cc', '4978d38b-d684-4336-8d3b-10409b618cb0',
```

```
'092413f4-4d6c-4a0e-84f5-5422b1ffc6ac', 'dd567307-ee8f-4e45-a10d-907300cb2537',
'd211ae3f-7256-4ef7-b1b8-1018b244bd49', 'cc998035-b8c6-40bf-b9c9-0edb04b73af7',
'8eb7e1de-b19b-406a-95fb-e55cfb48794c', 'c820d2f3-14fd-48a8-99ad-07ab5b743966',
'2b090282-4341-492f-b723-2568ce1b6b6e', '62a4d739-8e25-429a-96cd-c223006895ea',
'49182561-7b61-442f-83aa-f0c3a2de0f6c', '83c13c27-39a8-4c6a-aee9-6377be03124c',
'9869da9b-21ff-4be9-98aa-60b0da7bfab9', 'ddd14864-c950-473e-bb10-d27765b3b138',
'4d51607b-2981-4142-91f5-81c4be3ceab4', '73f7a913-3734-4a90-8e16-ba4bf8aa11e5',
'1f071331-6d80-4083-8401-e77a12ccab00', '14ff3f06-c016-4b28-bdf1-5a720aa28bf9',
'a24db8d5-bac2-43ad-ae28-5f857ecdf12b', '463d5cc8-f4eb-47f2-8efb-eeef1b52149e',
'118abcb7-127f-4503-b9e6-b6f508916aa9', '935a1ac1-23df-4688-8b80-67c841cb1c44',
'509b86cc-b40b-4c17-8679-10618a2326ae', '6d93fab7-2c78-4944-8568-0a117a3d08fe',
'd6f2aa05-b3a7-41a4-96ea-cbd2c5647e5d', 'aa684849-fd78-497d-acb9-a51e65d3b064',
'52d114cf-82ba-4623-94f0-7d9a88b0cf89', '79f22933-f226-486c-8091-28346a90814b',
'078eede7-822d-412e-9733-952d3251b482', '96fdb22d-7643-41a9-887a-f365dcc89c98',
'3ec288e0-ea11-4a31-a8bd-d6721ddcb411', '182591c1-c329-4672-a42b-d9ad09501148',
'24ba36f9-4678-47d0-993e-f864f8bbfa49', 'b4f18f84-e42c-4b2f-9e18-d67c99e1da6b',
'c2c9c786-af61-43d4-a03b-973086dce92a', 'f52ce68a-b310-4fd4-ad10-30b75bbba3ed',
'2433c289-765e-41be-9af0-d555dfa76c97', '7f23a3e5-3f76-4de3-9ae0-efdb388ddff6',
'c828ba7b-0b66-443d-884e-0162fb6ba14e', '247db809-4dbe-4903-b340-0bf9287ea81c',
'33c0f33d-392e-45b8-b788-0e2d55148526', '6baaf689-6f7c-4e01-8cc4-e233156d82a0',
'9a63ca75-352f-485e-9659-553c823687ec', 'cd733f78-238a-48fd-9b94-14bb2502f28b',
'4146e97a-be37-4f33-9053-cef27b3ea744', '36ae4fb5-57d4-4ef8-ac5d-03795d9e3702',
'2f5a4b08-07e7-4529-b77e-89c5bfced1a8', '6d8d266e-419e-488c-b819-ed9ef1497d92',
'cc526a7e-6f5f-4802-a65d-3c245e237084', '4463424d-9fd3-4c5b-af7c-35c36e08acf7',
'fed92176-10df-4a1e-b331-f0a6dccb1956', 'acf4ee31-fadc-49c1-adaf-272743c7a89c',
'2acc84d2-80b3-4a6c-bfbe-f24b9d98c535', '908dccb0-b74d-4aa6-80f3-73401a562594',
'a8e4df68-88fc-49a9-abda-8ec489324277', 'be29f8a7-a2b9-4a65-bd4f-9dc15e6e1690',
'de34a3b8-4e6d-4c01-82a2-9098bcb83e1f', 'b3e84263-7926-4690-b60c-977fce3dbd6b',
'804f1e55-a2a9-4161-9007-02923cbb29ef', 'b8ea2e04-fb9d-42e3-958c-8e1c38d43190',
'7cbab9b4-5a05-4bf8-8e93-7dabb6067659', 'f262a138-9585-4570-9d2d-6815c4d8b6a6',
'e8f7058c-654f-43d0-9a87-e16fa0210127', '51a8d230-5fe1-492a-b7bf-e2460eb4e583',
'15ff4d9f-a9c0-48f1-85ea-cf310154fa96', '2aa75919-b549-4d12-b467-fd4ce0fd092c',
'2e403442-b986-4e7e-8027-e5e9f187e7a2', 'be01ab05-e355-4d68-a6a9-163c140938f9',
'c18aea9a-8d55-4f16-bc1b-81094e79aff2', 'af66b2e2-6c08-4e1f-8353-38c9bf3f6125',
'c13b7711-8bd8-4ae0-93df-084176839a9b', 'bc177bea-e917-409c-8597-c547ba43422e',
'2c90f5c1-16f7-4365-8943-d61c91a1435c', 'c34ebd07-42cd-497d-b81c-cf76e3f76c67',
'2b0fe1e7-a685-4a3d-aca4-94f56adb4c0f', 'e456398d-60e2-4f16-9357-c67bd12af980',
'4d6909f6-c16f-4a97-b61c-dec96567c81b', '943506e8-da33-40a0-8247-9af5a665a5f8',
'3b902631-1c57-4958-8348-301e3bdcb05c', 'c004b0eb-b92e-4b76-b036-04340cf47fba',
'5c1c8daf-3d46-403e-a8bc-d8f9107f4a31', '0fffacee-47d3-4fc9-928f-42fd4719ea5c',
'68d4f21b-2096-4ab3-8bca-37f5ff779c09', '28e7bcfa-148a-43e7-ad0c-78e3eff3a3a9',
'1bf17fa2-c0d1-405d-a46a-7d7b1d7c4a2c']

Class 5:
 ['a0746abf-bc31-437c-a3a2-6296815c9c9e',
'96553092-5ffe-475e-b365-d05221d8884a', '1336f8a9-8461-4905-b665-240d95a392e8',
'f4b88912-3646-47a3-a170-629102129459', '00f93b02-bf39-4a58-9a27-c90bb3f1e25b',
'9080288c-c9cd-430c-8b8b-4f32e82617a9', 'fd0c3185-a692-49bc-8dea-2432d022dc3a',
```

```
'1193ff6c-1324-40f2-bd11-a0774e143f84', 'a0c38268-115d-461c-b89d-74d8835174d4',
'664ac451-ff6d-4453-ad95-7ed0490c01d2', 'ea1941c3-457c-4301-8179-a4efaa6dfadb',
'8c74fd5b-f93d-4540-bb2e-a9c345b62fca', '1a734946-8531-4449-a9ce-0c8a3f731306',
'15cebaf3-33c6-4eda-b9c4-7e67fc55e70e', 'cde86235-3588-4273-8713-1e42f3da03b9',
'50df83cb-cac2-4bb3-a378-2a333df3e77c', '6f24536d-d9b1-4ad9-a187-c9d1e826d38a',
'f113151d-4ed4-4f45-b414-4aa0dadb5a98', 'f2c407ec-05ef-4b25-8132-71ab5dddec84',
'79f0696a-ddc7-4586-a1fd-f739576cdad7', '6ab7b48b-58cb-4ac6-8967-1f8ec3b3321c',
'59ebaaa8-6421-4ca5-87b8-faa18c694fcc', '79e7e5b7-4bb8-4986-ac78-c95167d822c8',
'56ecdf80-78b4-4c87-b7e7-0539014c2bee', 'a716888c-462f-435c-ac11-1fa1bbb9d1a4',
'c810062d-dc52-49f6-85f1-75959ad95212', '1201dcf2-ece2-4438-b9ca-324646255107',
'712c5c92-5b76-4c44-acdc-86b20e8b5733', 'ed2c0f85-ab9f-4114-b74c-3fcd56d4039c',
'dd0733f6-db55-4f9f-97e0-a9e92590052a', 'cf6e3f5c-4806-43c0-93eb-8827e474828a',
'0cf839cd-e368-4ee4-9996-0ad245d62c61', 'ac700b07-d448-45bb-82c7-77a3544ad118',
'43b91b61-17e2-45e1-a52f-f1c97f20dba7', '8970deeb-fe0a-4814-9451-0d567f9f82e1',
'99f38492-7290-423e-abe6-4ea65da7e9da', 'a2fa0449-f696-4527-a329-97e3da0fccc7',
'f7eae2ee-2d3d-4ebb-abbc-391696b60d53', 'e0254bad-397a-423b-887f-5037806b9ec8',
'10fc4100-4110-48a6-9b20-ec8c3b1f82bb', '1d5fd77c-97c6-48ba-8872-d82cc2397fc3',
'a8620b83-07f3-4573-9cc0-234e75a0987f', '5db4020d-c099-444c-a6d1-e6b469a71044',
'c5c2ef7b-915f-4d01-9c53-529bd2a31f82', '421c7170-ebb9-433a-bc33-324e367817dd',
'64a43894-0734-4eac-8db7-5ca7e13d5ef2', '5d38cb08-908b-453d-9627-8cdbe363df92',
'7de00438-c195-478b-bd5c-bad88506df31', '334171b1-9b02-486d-b933-3961b947f016',
'fea77b4c-de6b-4755-8d1f-c1dcf19db966', '3d34e2a6-d5ac-487c-93f0-35a5171922d0',
'6b111d37-4666-468a-9462-4cead066af69', '93d6d311-da76-444e-b2df-9603004c46a3',
'1c09e648-abb4-4b7d-a4ac-366f68e8ec05', '49821c60-53fb-4695-b4d7-4fc5dea72044',
'cb7ef9a4-d4ae-4d27-8a22-6a816a2c63e6', '4f4e02ee-8146-4494-82ba-d00e3fce48bd',
'0081fc53-45e5-4422-a9bc-c1c9a9d0e0be', '0241de37-f71c-4612-9806-1cfc512f8cf0',
'145091e3-429b-4c6e-9b40-9819e60c1475', '4438ee9d-e762-4483-a617-cfb38a03b329',
'f085e373-0705-4fd4-8ced-03ab339420f8', 'f5615c45-1061-4241-bdca-87405e9c2934',
'28be2de4-0078-4dd2-9e9c-6dfb4710f2d9', 'b6f75df5-df02-4dad-b243-eb8d66ea3a80',
'39c54c46-d65f-47a2-8fe8-6d80df03e8ca', '58ce7830-8d4e-4b7f-b984-cb07b7067697',
'190e9832-00b5-4d5e-a4a0-f9fb8a2a34d9', '615ab047-2dc0-44c5-9e5c-ddf7926b3ef7',
'fa2ba907-f9f0-452a-bdf0-89227d36bc81', '3fda8930-3741-4788-82ea-0c08ed80efdd',
'44d087c1-ce7e-4066-a487-1bc2bef282d0', '3bad07b3-c13c-4c31-bfe1-0efddbb9982a',
'8fa03163-7c2c-4da8-ad70-95dfc2010443', 'e333ba30-8116-420f-ab7e-010973b2a805',
'4bb93765-b14e-4ce7-bf37-fa308c25f36f', '38c15fdc-c902-4cfe-820d-4807c5f26d4f',
'eff2966f-7fd3-4036-bb7d-1e909e499806', 'e25c73c0-f373-4344-900a-230034bc2b4c',
'9ec4e88d-60e4-4d6a-bd6e-d1b7177d2d52']

Class 6:
 ['f0a6233c-5c16-4090-bda4-151a5cbad3fb',
'd2ef316a-ec66-4fba-a493-814caea17209', '2678a5ce-7804-4b90-8631-bca481a3f1e1',
'17da4205-839e-4426-bec6-c26f325962be', '9e108a2e-5407-4449-871c-08aba74b4ae3',
'f1b9d739-a235-494c-ad35-5b9bd6f416f8', 'a834f1b1-4458-41c3-a218-1791d55dee85',
'ecd16e09-2b0c-411e-aafb-106057682143', 'f07ed803-94e8-49b0-966f-32de0bac7edb',
'e9d8ff88-a18f-4127-bdae-6dc337d4c79f', '65f9dc82-8f88-4c8b-a903-9eb3d456b2a1',
'26fc8d89-7780-498a-8a5e-cad852d44832', '31c98b61-6193-4763-a01c-4eeca83972f8',
'7eb5e022-027f-48b0-8458-fac954415c7a', 'ad0d8dca-89a3-4c95-a0b2-ff324f929045',
'a2f83bf8-9e96-4403-83ee-c0f741875234', '61c3b4b1-e33a-47d6-b62a-3dd2c78e4497',
```

```
'a75771ad-ca23-42bf-98a4-32e2665c0558', '5856a04b-25d0-4319-a024-adb354ca4341',
'18f2b710-73bb-4741-8267-cb7de3d064dd', '64f7b7af-7917-4087-a742-8213a4d54870',
'c34b82ac-05c1-4a53-907a-0d0519a09f8f', 'e46adeae-0140-4e08-a4d0-77e96ef01785',
'299ca28f-5773-4764-b408-1d24471be9a3', '73381e02-eb81-4502-bd8c-ca1a698c105c',
'43f6531a-dedc-40f3-aac3-0a6091c03d69', '2a999785-737e-402c-8281-70cfedcb18b3',
'ae10f25a-c978-48d1-809b-910a39ad5952', '013c24a9-4491-4704-8d53-80c3e288b532',
'b2c89103-aa81-46fe-997e-c57ff6af77d2', 'ff012479-5107-4faf-bf1b-4c47995dac12',
'7ec60ff7-40a7-4e3c-a5d5-7670566f97dd', 'b42b8d30-d79b-4476-9a4b-edb639732b3d',
'112825fa-dd5f-42aa-a3b7-e1bc558c38ab', '337271e4-6316-49b2-aef1-27db09b31d9d',
'b0cfd924-4124-4953-93dd-81492075a96d', 'b81cf227-bf30-4209-95f2-64a0321c7173',
'224cb616-c754-4ac8-b734-45e24c951ee0', '2ac45b9d-c31e-4e46-a849-5b8e1b5c7466',
'ad9dc134-a9b3-4339-8743-0462f8f70fe2', '0e0016f7-87b5-4807-9821-462b7001dcb4',
'ba9d01a5-a673-46b9-acdf-0a8761c8acaa', '25872ead-5db3-4763-94b0-078a3fc7334a',
'fb14edcc-30ec-422f-b907-de0f27ad64cf', 'c46f64d1-64cc-4d8c-bc53-4ecd3b6f0b70',
'd5bfe7a3-7caa-459d-af38-a3eefb3b2c9d', '2d5ae499-07a0-454f-80e7-1a3b3518fd5d',
'8ed52f27-6921-4347-b173-3211cd0eabff', '7eb16026-fdfa-439d-b308-68896077cefd',
'08adfa87-aa31-4fb9-aea4-1324eaa4f8d4', '2040ddce-dfd8-4bfb-9513-cf65251e070b',
'efb99fb4-8314-47b7-82cb-9ce5897db3e9', '436ba373-d092-493c-94ad-1e48257142c9',
'959ba8ea-ca30-40df-bd0a-a29ffc354fca', 'b507c022-15eb-4f39-88cc-f7293175024a',
'd54b4a6e-893e-405f-93a2-52cec663971d', '4cce7eaa-9c63-4f28-a28a-aadcdbf79de9',
'df0ee0fe-dc2b-4052-8be0-a6cd9fbd4bc4', 'ab58d068-e9a5-4e59-b52c-a42336c29440',
'abcb2a5c-701d-4ab9-b0e0-ef52b01ecec1', 'b7f93af8-9f60-4fd0-85c7-bba0e93ef14a',
'7cd97c6f-aef2-4bc6-bb98-56b9d4cd026a', '0f9ecb34-c1fa-47ae-986d-d7223e5bee8e',
'9fbacb38-da87-4405-b458-ad95a647aaf9', '85c2befc-65f6-4eb5-8f3b-944005967fda']

Class 7:
 ['0f5b10f8-9d88-4ccb-a47c-5c358b74aec6',
'e199c6e9-81f1-4d69-b92d-046966f2caaf', '7a088147-79aa-461f-b28c-9bd57de7aba7',
'3e464a60-a621-45a0-a2ac-89f0d58ed021', '9bccb837-d9b3-4205-a2ed-34175c76658a',
'1c7c6ac2-4763-4ccb-847f-99ff733ea7fe', '2e3e8cbd-6b2d-4ea6-9e6b-633777213ffa',
'9254c8f0-40ca-4b40-b8bc-17808218a3f9', 'd0b8d1a2-47ba-4a16-9a68-a421f0007db0',
'f790e4ea-22d6-4dc2-b14e-51480449eb6e', '84a05458-953f-4e65-95ca-e3c60442e731',
'6a879294-1358-452a-8c15-465d8a69c92a', '72667ef8-5ab4-4f4e-b573-a2996cd5506c',
'de3d66b3-07fb-489a-b7ea-3edc3b9bded8', 'd9fb0856-fafa-4ebc-bb1e-e616ac84b505',
'85f5b2b7-3be9-4f45-bd20-e13528055ce9', '73b2370b-bac6-4115-af6c-c554c24ff8e4',
'cdddcc85-b56c-44d7-9f45-948af5059a33', '76476c81-4c15-4414-ada2-5b2161ff635a',
'98175ed8-fd86-47e0-9cc4-ba325e1b4063', 'ee0f9644-3a29-4b77-a598-1f5344e36b8d',
'81da1a0c-d3f3-48fc-a753-76e9bf7ab61f', '2dbd5b49-c323-452f-a1c3-2e2e97b5191a',
'7ef6dcc0-2aa2-47ad-80c9-8116190cfdcc', 'aa92a2c6-4f4e-4709-9f70-574570692403',
'a7639907-70c9-4f92-9f52-72a8d6b4ac04', 'ed083aef-8385-4361-ab6b-fcba83b289bb',
'c44713be-7988-4137-a934-e4a997ec9542', '9b1a5e74-cb84-4565-a2e9-3b1cf7219004',
'1cb38ce4-9c10-4243-a28e-97917910e81e', '1bd78383-fa00-4cdb-b8b8-daaef8731364',
'a22039de-5c4d-459a-a899-bb4fc8eb4bc0', '2ef8224f-bc83-4751-a7aa-70091a597e16',
'fe64591d-77ff-46ef-ae9d-c0143f67f149', '1534c358-5bb3-4868-aeb0-dbdf934fff5a',
'35cb5a10-aba4-43dd-9010-616bed257b3f', '19cc4b3b-7a64-441d-8e90-c68f9ceb6022',
'ff55dd00-08fd-4424-9b46-4a0c3accdee7', 'd2bda2e4-675a-4a9a-a16b-770e3bcccd28',
'897e0b3f-a3c5-4aa5-9c3c-f9c3e238a979', 'bf2cf129-2346-4c01-a49d-92759450c2d0',
'bb0d6fd6-3b70-4afc-aca3-7fc58e6455d0', 'f7969daa-41fc-42f7-b92f-9929ec5b0b1a',
```

```
'629730ef-a470-4626-b17e-29cb3d1ce350', '51a7c2f6-2588-4b3f-a084-53b422f74a1e',
'ae7b5a7c-80ec-4614-826b-190184f13573', 'eb6e8e18-93d6-480f-b63d-f256b3300538',
'e140fd2b-a199-4517-8598-5369c7bb08be', '2f17a508-af95-4f66-9206-1903768ef0a5',
'6844399e-37ae-4e8d-9c04-60f6e4a44881', '7f61564b-4203-4fb6-ac6a-2cc555465ed6',
'a1c6f6a9-eaa1-442e-8b4e-6b3454389111', 'e04ae533-ffff-4e12-bc0c-7e4c5ca26f59',
'411eb839-aa72-4e81-8c62-b9ee8c3ebc71', '48c2872c-a3c2-450a-9c85-751a030ae676',
'854d6f69-0d04-422d-9b24-b130cf59d138', '08325c31-6a20-475e-a4d6-3f17e144479c']

Class 8:
 ['21d31dbe-71ad-4b58-b607-98db26c9b82c',
'5e66758e-a947-46ec-a90e-5f56f136731a', '55452468-1438-4bd8-bbde-65ce0572db4e',
'92f97209-748c-41e8-b92a-fafde75023a6', '91cab205-c0d4-4989-9b1a-74fb2ce97e07',
'7f1af88b-cf1f-4e0d-af9c-ce69612c9da4', 'ffcb797d-bf0f-4094-89ce-ba575d5fbb33',
'd8165bb0-a719-43be-a1d8-3e21335feb67', '89872db9-f12d-4c4d-98cd-d60d6350745c',
'a2c6aace-9c42-4384-8a4e-f68728479821', 'e2e815f6-6eb8-402a-9564-dd8ef9ab04d4',
'722c0c97-6ff8-4c72-974c-f770e8d62cdb', 'bd2c9f3a-db57-43de-84a5-7ee210ddb53a',
'4f6abe43-ead9-4599-bca4-d644b41789bd', '64f21bd3-b103-43f2-8731-b1844fb4933c',
'2b5fedaf-8ea8-4978-8975-2e3d7468e95f', '976abfc7-cf69-4415-b5b9-2af6766ec358',
'84077486-4fd4-4e85-9151-8564186887fd', '6270a43b-9676-46fa-bd92-83ed49de679f',
'7278baea-af4d-4ddd-8450-0c41152f4bbc', 'c4119bfb-5b2b-4deb-8c0d-0efe36989b76',
'191bb8ce-3bf5-4196-8f05-f563285d0446', '01c8518f-0b5c-4bf3-813d-c4a704d64e9c',
'61ea0de1-f3d7-43cb-9a69-2383b9c9a15f', '93f73375-7287-4112-8035-c937dee14213',
'970db19f-b120-4adc-8c68-4a2f72944f6e', 'c3004d25-432f-4293-a617-f24de091c35e',
'8fc047ed-639f-4495-97e1-e19964e80218', 'd9c17459-af37-45ea-abb5-3a3819bcf89a',
'eebdf5df-05eb-4b5a-b75f-c1cfb3b46430', 'a17b1f87-ff73-4654-b3bb-2f3a67456939',
'0e9eb780-b556-49c6-adfc-d1656f9a9a70', '8505d238-714f-4485-8891-0e9bf4812e37',
'9a75d18e-481a-477b-a8ce-5d7b06274b78', 'a21a5648-7289-4b72-b8d1-e43315a45dd6',
'33749d7b-0d1c-4151-95e7-ade452e66139', 'bb8bf9f9-f0b6-460f-904b-2e1b43567ec7',
'fe6dfa5f-a311-4433-97e3-19de40e8b8dc', '3d0c3336-809a-4ed8-bf7e-64fcd1c24836',
'44a1d39f-fd31-434d-8dc1-cb8101741fd9', '64ca5f72-5c05-4c62-838a-06b49de67339',
'55cc7dd9-639c-46e1-b53c-7d02a29f7dce', '2fab933c-8c16-4203-ac44-91712016e656',
'5d9b1b21-e1e1-4ead-9a49-3c99b67004f4', 'ac5b9bb6-ad71-4fd0-ba45-1b470eeceb8a',
'd8dfc884-8ba2-4dba-89fc-3b2527e68c83', 'c3e06c8e-b1f3-4fdb-9e11-c9d67c202d0d',
'f572f448-6e83-491f-b793-1335f97203e4', 'd3f7be94-e4d1-4878-bce9-361caea20478',
'9960623f-c7c5-413c-a2b1-1d09b38ff1fa', '5319920c-f1d7-4b95-845e-f52803d2acf8',
'b4130497-cc42-4823-b3d2-0c092d4d8d2d', 'e658bca7-ad24-4d5d-8c00-4342966f1cc0',
'1bc2355a-71b3-4fda-b714-b076d48a3348', '81419f60-2e9b-497e-bcf9-49c613224162']

Class 9:
 ['4ee6b228-e993-4380-89cf-ad6776fe3af6',
'8954d7d1-c0db-4a55-871c-a0b77fa4c868', 'ffe5c654-2c3d-4098-abbd-e916ef0a7b50',
'29990497-dc37-4ec6-9c5a-81e8201e6d32', 'ebfe7e5d-5619-4591-8b63-741e3a5d5342',
'cc40d145-7990-4c06-a153-2e8c2acce24a', '716e14db-e095-4152-a8e8-f0622709fde8',
'd19dbdca-0541-4400-86f3-d251ebc86615', 'a7b1caf2-88d3-42c4-b536-f915e3389b9d',
'c3e83d12-c218-465c-a3a1-3f9ba277bfab', 'fe7a25ec-2b1f-480a-ac1f-d04bce992b61',
'f0add804-6248-4033-aecb-c994fda5a84f', 'caf3b25a-be58-47b7-9122-5c6f7dc56e3a',
'd5a27ac1-86c0-4931-9658-972c7f43f80a', 'e683d0b3-2f98-40ac-9654-bba9a995e674',
'806abbc1-45f7-40b1-8dd7-3b1d4f63ed5e', '911717df-55b3-4f93-8e86-da2484fdf453',
```

```
'b804b490-3837-42cc-83bf-aad82611b8e9', 'd3f3aab0-f5d5-492d-8607-d49448e38f60',
'5e2b4bbc-04f4-48cc-b223-1abc001f92f0', '9799ac09-ae60-42d7-aa3d-ca9b2c9d35b7',
'b6a81802-0a7e-4141-938e-6b233f9c6e62', '9d25e79b-edb2-47b9-985a-0020a1082c3f',
'75d308d5-348b-449b-9910-dd34fcb2e8cf', '33075ede-abd6-4e12-9f05-377fe4e40227',
'd7a15da3-8d61-44dc-b4ca-265bd98b4839', 'a852fe1e-e29d-4ef6-a444-437dab5ee175',
'66c89729-68e2-4f4d-9273-3b8db4f2f3ff', 'd457c4f9-28e2-4e27-921f-3a3c6c752c7d',
'eeb2d88d-4de5-4c4f-a03c-7e381c75243a', 'da1a085e-0806-4746-94a7-196cadc75179',
'2092253d-9e06-4415-bfc9-6510acd629f7', '42884ed0-e2ff-4d77-bc7c-cd410e7a911c',
'25b6f20b-92b2-4df8-b395-eb5a1ee324e9', '8016d857-938c-4437-a905-da05145f94b4',
'70ebd416-f0a4-4386-a978-297eafcf9e25', '1f40952f-02d7-4e90-a81d-1b769f1f1634',
'8d6b8236-aa8e-4a47-9d6a-94931019340b', '38c282db-3db5-4417-9bd4-f90a179c371f',
'8558687c-f0a7-4757-8846-23d468442023', '95d23a29-c388-4e12-9867-2318ca2e3dc1',
'2867688c-43c4-4317-812b-d7397962fa70', 'c4ea64d6-5528-42cc-ab1d-ccdeb188257e',
'0a15ef66-d59e-42bb-bf5d-c8869379b0a4', 'f5135d5a-34e9-4470-943c-b86b1ecdd789',
'81adc7ec-5a92-43dd-95db-f49df54b672e', 'b3aefff6-ac2b-4afd-825a-0be6378d6eef',
'3be27d52-7c5e-428c-a020-d590500fc3b0', 'b8dcac2a-b6c6-4a1b-9e27-d6236bbaa497',
'be087da6-152d-4952-9ff6-facc71b10320', '194bbcd4-808d-4352-a1b3-f93591a3cc2e',
'85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0', '835e846d-4738-471b-81cd-28224d8d0a9a',
'ab9a0c13-f933-4e65-8ced-45ee241dac68', '512f333a-fc03-4f15-a494-a9e8a235381a']

Class 10:
 ['08bf07a3-6007-49a8-ad13-a31db88f8146',
'93083cff-8775-4452-9901-d11ec9a7b7c1', 'ae42e66f-a525-42a9-96da-993fce68bbbf',
'f13a9a5a-1b79-4d58-9243-106abaf1fc0d', '0d4ee39c-babc-430f-80d6-dad3e76fea1c',
'6fe4df75-9585-4232-9860-a786df02c10d', 'af738f26-16c3-447a-9f86-15c7d461e0ec',
'254593fd-a178-48f2-af7b-6488be1e142d', '14b8bbd2-eb47-4138-a262-e8f09b775f33',
'3c2779ef-93c5-4fd5-b023-aa114b96962e', '8a719042-f1d2-46db-8cec-e16f1dfe07a1',
'4289798f-60bc-4c2e-8834-3934db1db740', 'cf7472db-7fdf-4056-be00-b297565773dd',
'65327ae1-57cb-4e45-90d1-82f3873253ef', '3b08c894-4c48-40de-a48b-00d68c4d1650',
'41ba54c8-8376-49d5-bd51-6754e1018cc9', '6b493db6-e760-41d1-8237-57b90cbc35d6',
'11fd0ccd-8d01-41bf-9281-e1ac60d3748d', '75b4ba29-5c8b-4c67-bda4-45d90ccabe1c',
'04f7b9d1-3c3d-4981-bffb-3bfa3d731cbb', 'e93aeb48-5e84-4f24-b36b-9c741a8d5d53',
'381a8c4c-9e37-4ed0-85c9-3dd56d702629', 'd24b0821-289c-4e65-9217-514d954640cf',
'7d14ebad-4689-450d-83d2-c3e67b5b6dfd', '9de85ce0-9fc8-4e33-b786-f978ec804b70',
'c038bfbb-63df-44ed-8404-0b2c8ef27666', '79b0ffb0-e30e-4b71-b210-346bbd09c2b8',
'e53648e2-4bb7-4a7b-95da-b0e864b6c961', '89d0adf4-7803-4a4e-bbfe-b4764185df1c',
'fe016223-b4f4-4a51-8f60-a7cd034206fd', '19cef079-5ce1-4f82-bc37-4639f1de2615',
'0586e724-8e8c-4bf5-a003-b5f5a07b2b4a', '5261056e-9705-45e8-a2f4-2c05595a4bcf',
'20afdbd8-d0c2-40ea-bf0c-9302a0c9e236', 'c08f9900-051c-401d-bbb1-04977db09e9e',
'934be4e4-9110-42a3-a1d0-70373981e59c', '83069c8e-7fc7-4307-8803-10cae8e310bb',
'fda43583-391a-4af0-ad17-954801357d3a', 'b84aae7e-352d-49ca-8db2-448cef043a1b',
'0e43309d-17fd-4ba0-ab6a-79e83bbc4d03', 'd0dc5bd2-aa3d-4098-8e71-eed03d7aae41',
'd2bd584c-fb99-4a82-97b7-42768037c65b', 'bdbd972f-834c-40f4-92e7-b96416ff4ffe',
'29256ff6-fdf0-4ea3-ab26-f9e4fac9be7d', 'b14b72c7-fc08-4a87-ae4a-fccca09cc807',
'6235aba8-20cd-4bca-b8f5-bec7b60add5e']

Class 11:
 ['e952d596-3c05-4726-9d5f-4de6e2ea73c0',
```

```
'370fa4ed-07a5-466c-80bb-8751c1bf4f5b', '20c91761-9c44-4af2-83d6-9ce643e975a2',
'2b98f54c-7a4f-4b5f-aa0d-4e18bbb1b47c', '0afd356b-14df-4d71-85c3-c1e5c97f8d3e',
'c693357c-7108-448c-8a0e-a10a1530209b', '1e9cf16a-66f8-449b-9fe9-622434fe969b',
'c9e257b9-c943-471a-90e3-7d94dd537724', '3cb7555b-9fb7-492f-bb19-b1161fb35899',
'0f223f51-30c0-4bbf-ae5e-36350e6fa63d', 'a3787904-6153-4def-afc9-6af62080abe7',
'8a8f24ab-4a6b-4c06-ab09-c4f43fd2b453', '083f9756-e8a5-4bf8-9c99-bd587f499ab6',
'f38717e5-26b0-4c77-b2f2-3d2735efbabb', 'd0819c5d-7b8e-47f5-a5e2-09ceb9c1744d',
'2427bcbc-d904-4602-b75a-31c43c0b0225', 'fa1214dd-4839-4e28-af54-08ee4c267ebc',
'50e751f5-3645-49d1-9d09-a5aa12122974', '195e37db-2b01-4e32-95ea-31e828ae1a41',
'd7d2506c-e929-4258-8c88-e58daa825e52', 'ad206a41-8e52-4a60-80e2-ae373811e113',
'cfad3a5d-8781-40bd-885b-f93f29ce16f2', 'a541ea81-3539-4ba4-83ec-011fdb20e31d',
'e3f579d7-f0f7-4f67-ad3f-593f81d58369', '81aaa085-b98d-4d60-8801-86cd3c5beb6b',
'85a807ba-1bc6-48be-bd3e-9aaee84e02f2', 'ee55cd06-4eaf-4df8-8239-aa73871a8b63',
'475f7ac2-3376-478a-b33b-dc96e3da1ca6', '7486a910-3f70-4227-bbc0-6e578e23ad3d',
'35437988-399f-49fb-9a35-fde222a168c6', '9753eea0-5aa8-4b32-b918-22362ed899aa',
'e745e365-208d-4272-8a44-e79e4f175e3d', '0a31bb5d-4e12-4e3a-b402-5b82b64724b4',
'7972e34b-a375-4246-86c8-41712be266a4', '789618bd-e9b5-46a1-88ba-4cf4b9f322ff',
'3607d551-5b68-4396-aa8f-e251b80e8494', '18ad9a45-bdf8-45e4-bb2c-f0bb2b427ba0',
'77d8b80b-d3a4-49e6-afa1-9cf270022fbf', '4a3594e8-53dc-4fe5-95c2-9be409b018b6',
'b677070a-244c-4462-9d9b-2de24a13df81', 'acda4224-0668-48a1-a292-9af780829dc6',
'8038b898-aa9d-461d-b68c-1ad4ea52f31f']

Class 12:
 ['a68523a2-1c78-463e-bdd3-806305565f0a',
'811aa4de-9fec-4c09-88fa-192f88ed697e', '880ced31-c7ce-4391-9ed3-6df9000ed1f7',
'46920950-9c87-45b5-91c3-6eda34f6b4c9', '7d2cfa41-d4ce-403e-9d73-0e1b634ffc41',
'da520d33-35d6-4114-bd7e-6a0ebefe61ce', 'f5602717-0ece-475a-ab77-c5c0e822ff53',
'e007667f-29fb-49c5-afab-86e3cd93a9ef', '28eeb457-8e0c-467e-a471-f72b065fc285',
'9c079434-ae3a-4b78-bd5d-cff120f13d25', 'd838fc2e-8e5b-4c22-ba39-b58cee8afae9',
'c3b88349-27ef-4929-8ca8-61bb5cd63873', 'c34f8a2e-6329-41d4-bd9c-e69f050048c3',
'61e734fd-be4d-4c20-bade-cf00040c82d5', '272ea957-0ef4-40dc-8bbf-56eca5bf18a4',
'd8480b74-b242-4656-9aab-9b5db126625e', 'bd44750a-7c9b-4ae5-8e02-3d52c8d8c3f8',
'ae263399-2957-4e0c-b7cf-e39328fed258', '489d19b8-4e9d-455a-a685-c5469b96f04b',
'f3a873e6-f522-4c8e-8392-ed9a221b22bb', '7f028ec1-f4e2-4414-bcd4-25b158419825',
'fa336870-b077-4ccb-8a27-1712dbc9977f', '360b6096-c375-4da8-8add-cf012fafbab9',
'9f574ca8-e8c1-4f4e-acc3-259c59105945', '114a3247-5226-460f-850f-3d53ad112ed1',
'c7cd4510-a5ae-4dd5-9bdc-989235363ba1', '3caacc7d-1546-43b3-b472-4e6c05a17a19',
'723a7c14-ad2d-46f8-bd55-08a61f4bae2c']

Class 13:
 ['3e169c21-54bc-4e8d-bb94-8a212bb0208d',
'2098ef7a-9215-4d2e-8e95-0ab7e745392f', '0f453862-9311-4650-af4e-e3b2374a3885',
'18566d57-99ea-4f5d-aa23-7769f1b0dcc2', '99c9bbe5-19ad-430c-a24d-1ba0a3fedca7']

Class 14:
 ['90f5a899-c8e0-462c-8dd0-56e1dd748a67',
'432644e2-db33-4302-8c53-cfd7d7320c48', '8eb108a8-f7b1-465f-9b13-fb575a294c6a',
'96732e07-26d5-4cba-8239-9a637e0233f2', '546e9145-0bd1-479a-aa07-7ed8fbec79b2']
```

```
Class 15:
 ['7a0bf0dc-010b-4baf-818a-3327c79ff21c',
'4f61dcd4-8bf9-4318-a15d-eafc97405d2f', '11847e1c-8f11-4bcd-ad4a-e8a719255d92',
'14782c33-0fe5-4c80-bb74-4783fde4d8aa']

Class 16:
 ['319bfd09-fbd0-4961-9def-6b79ba0d1a54',
'b32c75ca-7234-4e84-82e8-e334a591c348', 'cf1db469-e2b9-4dc6-9532-c4d8319d82da',
'e99c9861-07d0-4646-bd04-05d281dd8702']

Class 17:
 ['7c0a441e-7e5a-4e30-b0fd-a43edb903db0',
'36843f90-5f68-4220-acfd-9b6084ffbb8f', 'e45f0ad7-b3cf-49b6-ad3a-eb40fce1fdb9']

Class 18:
 ['af42f628-9ec1-4bf1-a4d4-c1f8da6039fe',
'e17b4553-2039-44d7-af8c-5f3c1af643b7', 'a9bf3964-edab-4953-94af-bd64bee006e2']

Class 19:
 ['e78dea6d-311f-4f6e-8e03-40c2ff58f677',
'7fdb2929-de7e-42ed-a901-7bc882b31d9e', 'ce1b7a97-e6ac-4e2c-9251-9e69e7331276']
```

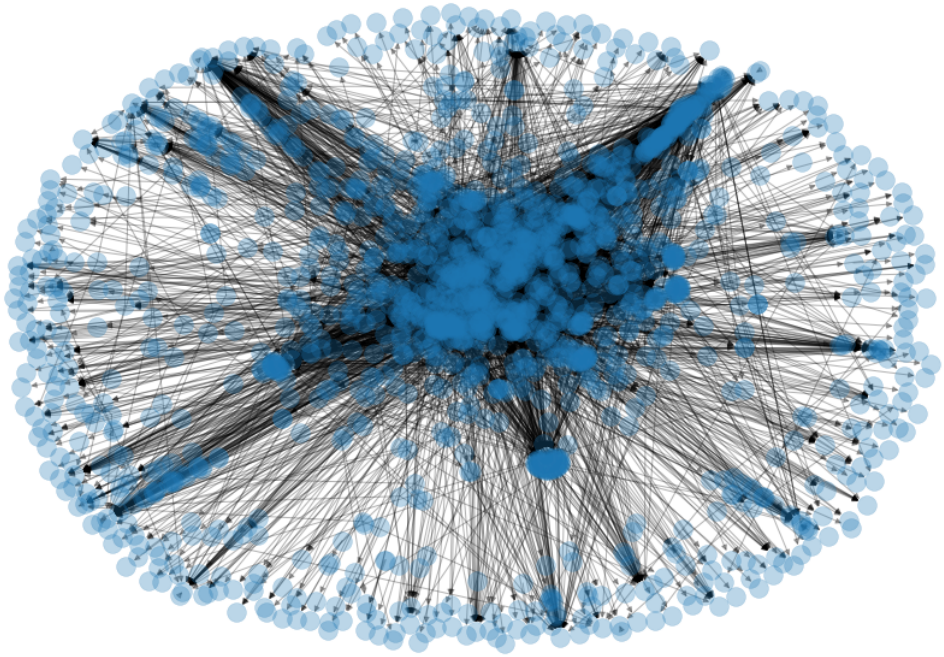
I filtering out any modularity classes with two or fewer nodes, in the line `if len(c) > 2`. Remember from the visualization that there were lots of small components of the network with only two nodes. Modularity will find these components and treat them as separate classes (since they're not connected to anything else). By filtering them out, we get a better sense of the larger modularity classes within the network's main component.

[59]:
```
# export  data into Gephi's GEXF format for visualization
nx.write_gexf(G, 'follow_network.gexf')
```

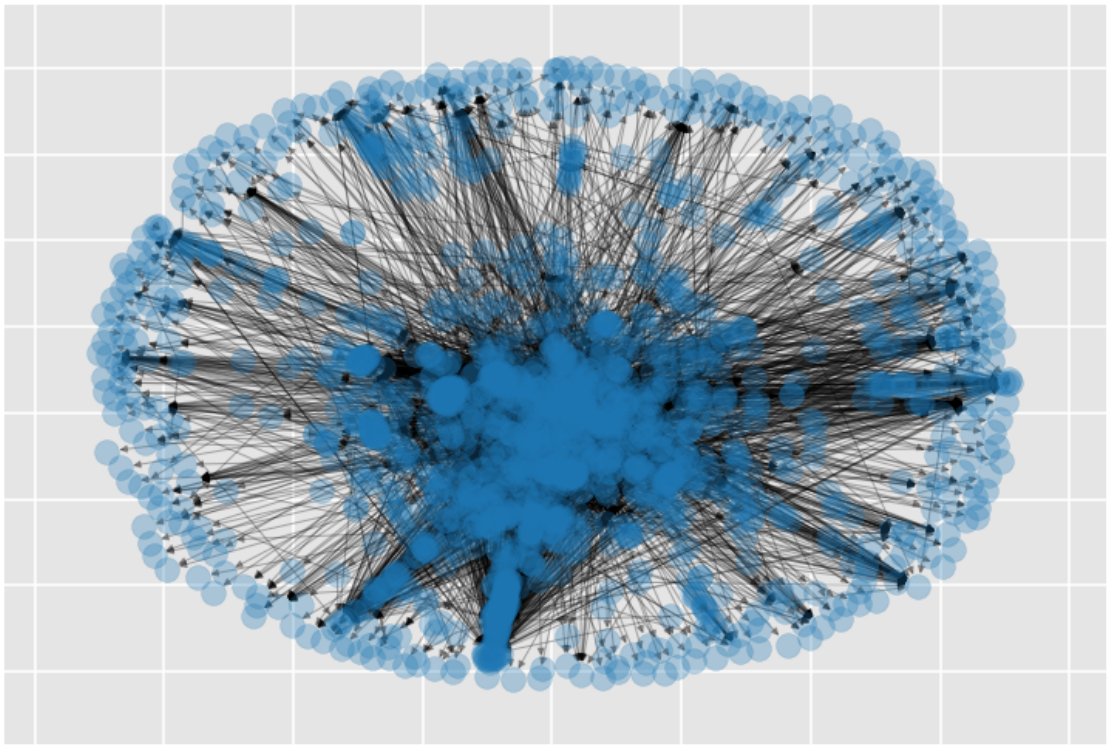
0.3.1 Visualization Graph using python networkx

Make graph from G from previous step

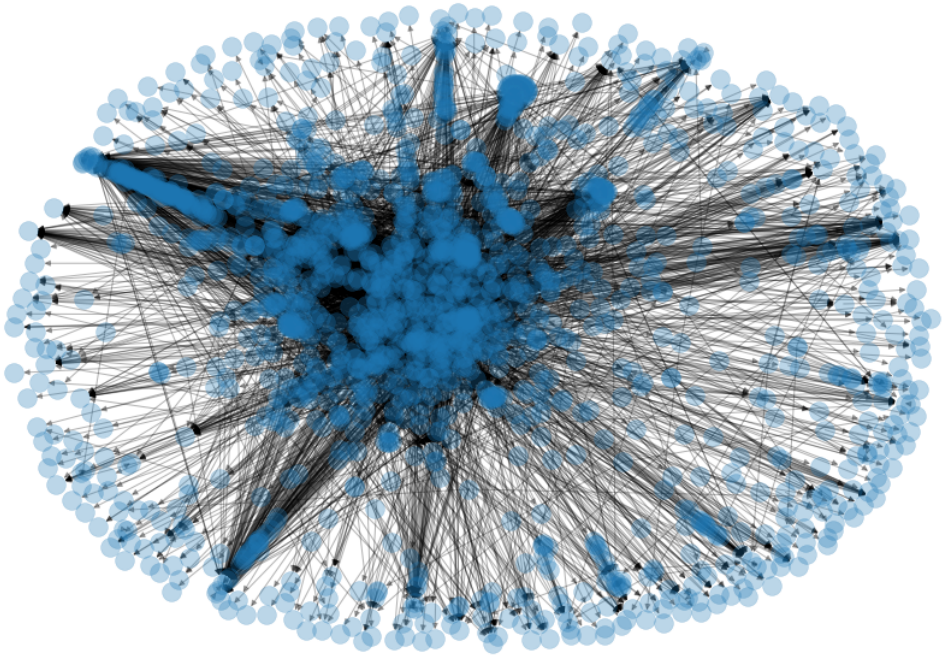
[60]:
```
nx.draw(G, with_labels=False, node_size=250, alpha=0.3, arrows=True,␣
↪font_weight='bold')
```



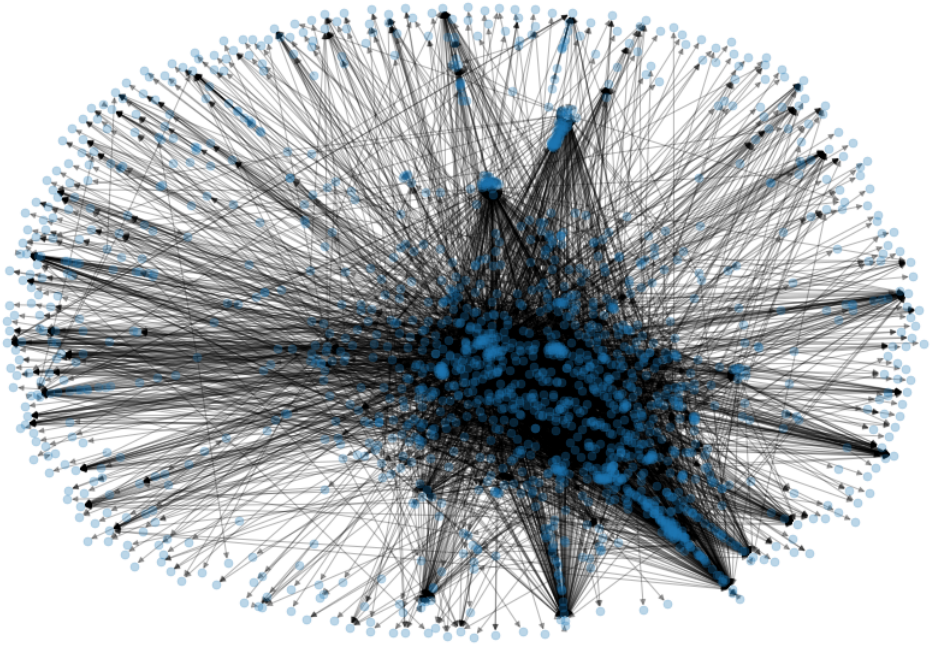
[61]:
```
nx.draw_networkx(G, with_labels=False, node_size=250, alpha=0.3, arrows=True,␣
↪font_weight='bold')
```



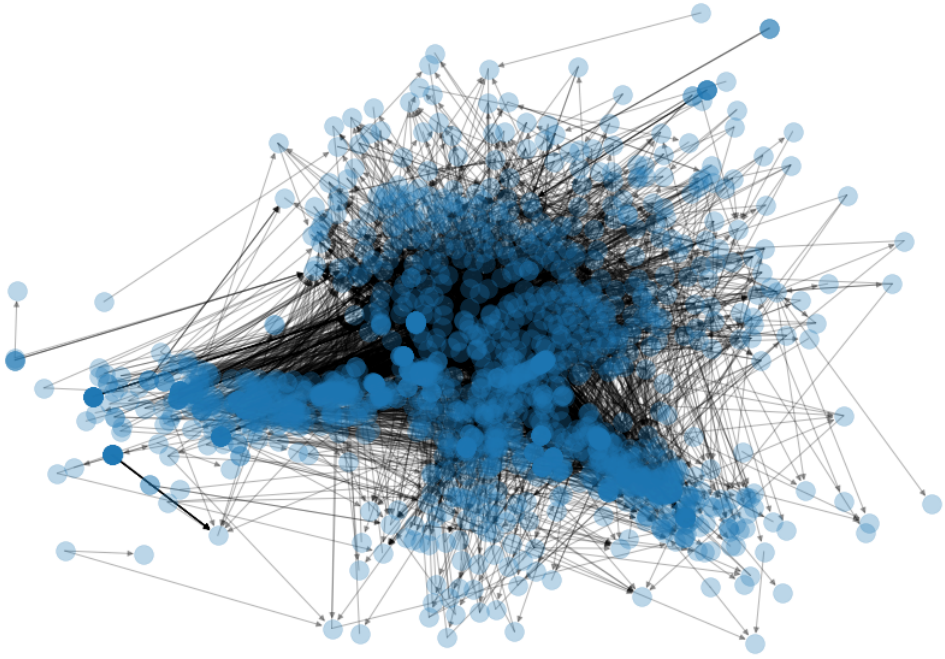
[62]:
```
nx.draw_spring(G, with_labels=False, node_size=250, alpha=0.3, arrows=True,␣
↪font_weight='bold')
```



[63]:
```
pos = nx.spring_layout(G, k=15)
nx.draw_spring(G, with_labels=False, node_size=50, alpha=0.3, arrows=True,␣
↪font_weight='bold')
```



[64]:
```
nx.draw_kamada_kawai(G, with_labels=False, node_size=250, alpha=0.3,␣
↪arrows=True, font_weight='bold')
```



let's plot the giant connected component (GCC) instead Connected component in a directed graph:

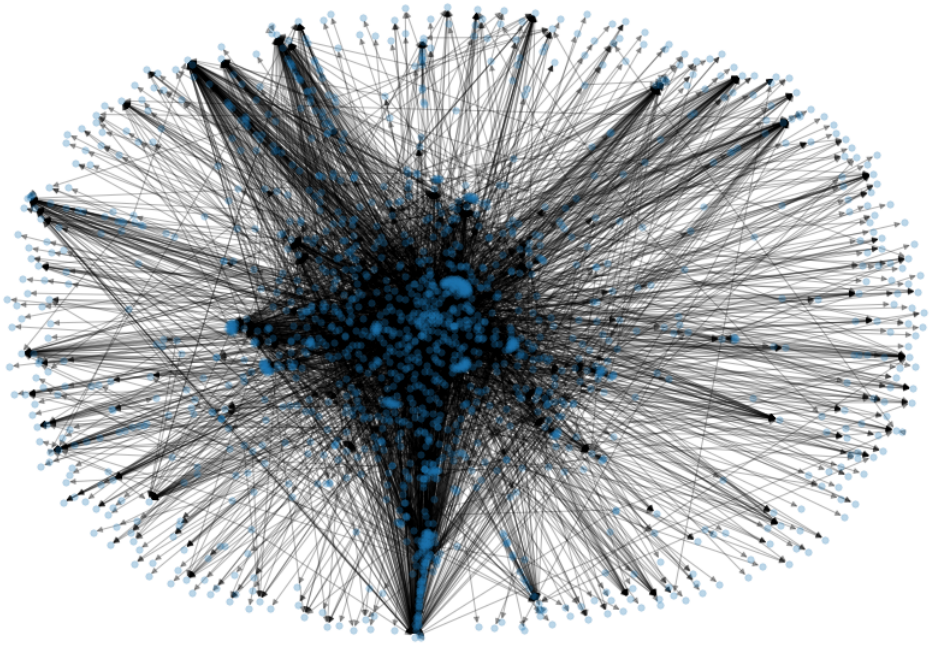
1. Strongly connected components:

   There is a directed path between node A to node B and another from node B to node A.

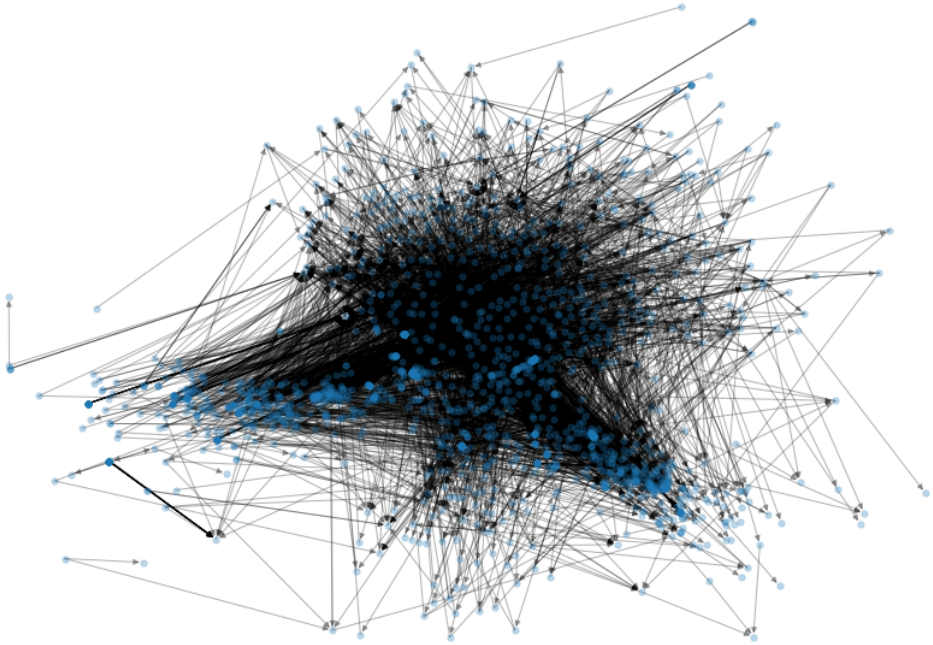
2. Weakly connected components:

   There is a directed path from node A to node B but not necessarily from node B to node A

[65]:
```
gcc = max(nx.weakly_connected_components(G), key=lambda x: len(x))
H = G.subgraph(gcc)
pos = nx.spring_layout(G, k=5)
nx.draw_spring(H, with_labels=False, node_size=30 , alpha=0.3, arrows=True)
```

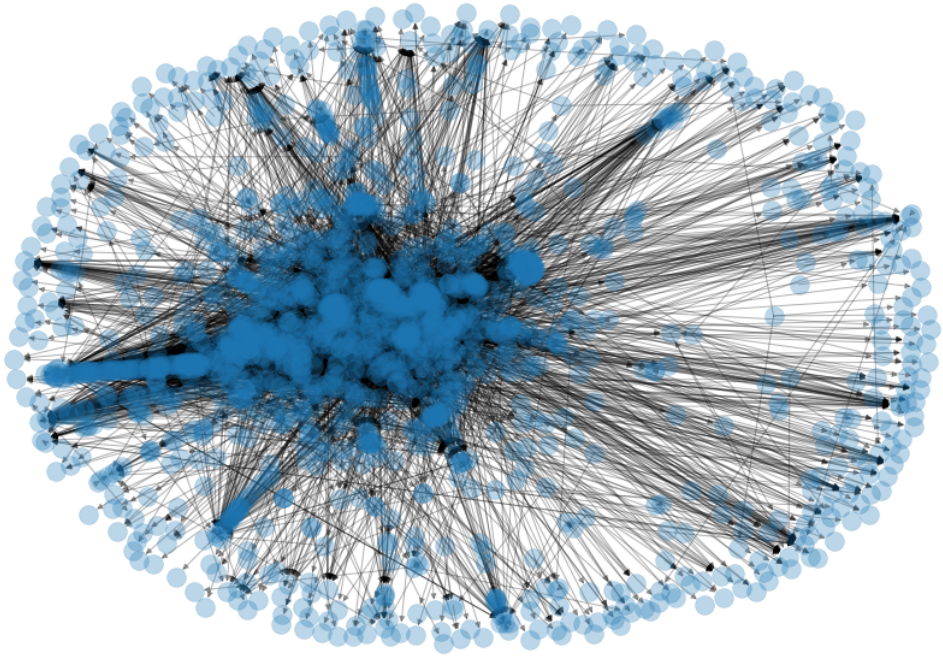


[66]:
```
nx.draw_kamada_kawai(H, with_labels=False, node_size=30, alpha=0.3, arrows=True)
```



Make from Pandas DF

[67]:
```
#DIRECTED
G=nx.from_pandas_edgelist(follows_df, source='user_uuid', target='follow_uuid',␣
↪create_using=nx.DiGraph() )

# Make the graph
nx.draw(G, with_labels=False, node_size=250, alpha=0.3, arrows=True)
```



Interactive graph

[68]:
```
import holoviews as hv
from holoviews import opts

hv.extension('bokeh')

defaults = dict(width=1000, height=1000)
hv.opts.defaults(opts.EdgePaths(**defaults), opts.Graph(**defaults), opts.
 ↪Nodes(**defaults))
```

[69]:
```
source = follows_df['user_uuid']
target = follows_df['follow_uuid']
simple_graph = hv.Graph(((source, target),))
simple_graph.relabel('Directed Graph').opts(directed=True, node_size=10,␣
 ↪arrowhead_length=0.03)
```

[69]:
```
:Graph   [start,end]
```

[ ]:
```

```

[70]:
```
# from_nx_graph= hv.Graph.from_networkx(G, nx.layout.circular_layout).
 ↪opts(tools=['hover'])
```

```
from_nx_graph = hv.Graph.from_networkx(G, nx.layout.
↪fruchterman_reingold_layout, k=1)

from_nx_graph.relabel('Directed Graph').opts(directed=True, node_size=10,␣
↪arrowhead_length=0.03)
```

[70]:
```
:Graph   [start,end]
```

[71]:
```
from_nx_graph_animating  = hv.HoloMap({i: hv.Graph.from_networkx(G, nx.
↪spring_layout, iterations=i, seed=10) for i in range(5, 30, 5)},
           kdims='Iterations')
from_nx_graph_animating.relabel('Directed Graph').opts(directed=True,␣
↪node_size=10, arrowhead_length=0.01)
```

[71]:
```
:HoloMap   [Iterations]
   :Graph   [start,end]
```

I tried different methods, tools and layout to get a better result. We need to be more specify to get the better visualization. Also, it is better to use Graph visualization tools such as **Gephi**.

0.4 Products DataFrame

To Do List

1. Users with Most Items
2. Most popular Category
3. Cities with most Item
4. New vs Old
5. Homemade vs Manufacture
6. Orginal Price Distribution
7. Price Distribution
8. Items with most Discount
9. Most Generous Cities
10. Most Generous users
11. Items with most Comments
12. Items with most bookmarks
13. Gender Distiribution of the items
14. Seasons and categories
15. Seasons and price

Read Data

[72]:
```
products_df = pd.read_csv('komodaa/products.csv')
products_df.head()
```

[72]:

	uuid	user_uuid	\
0	33949611-a3e4-440a-83b1-6cc1a2602b8b	06abe0e5-c408-4a33-9d81-5bfacc891f15	
1	a55fbc5e-b0b1-43c8-b25b-57685d2ee242	16848d64-b555-4182-a5b4-c0ae1e850e01	
2	d895881f-6570-4ab6-9756-367c12b1ab44	16848d64-b555-4182-a5b4-c0ae1e850e01	

```
3  c0a45353-2453-442f-ae45-d741ac5ecdc3  16848d64-b555-4182-a5b4-c0ae1e850e01
4  00871ece-294a-4b1e-be51-2d72ac1205f7  16848d64-b555-4182-a5b4-c0ae1e850e01

                          category_uuid                             city_uuid  \
0  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
1  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
2  8ff26775-1571-4a9e-904c-cd7951ba93ea  22972566-8739-40ca-bf19-a270d320dc83
3  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
4  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83

   is_new  is_homemade  original_price   price  comments_count  \
0       0            0               0   15000               0
1       1            0          550000  300000               0
2       1            0           92000   50000               1
3       1            0          310000  230000               6
4       1            0          310000  230000               4

   bookmarks_count  gender                 date
0                0       1  2019-12-19 02:42:28
1                1       1  2019-12-19 02:42:28
2                4       1  2019-12-19 02:42:28
3                1       1  2020-04-19 14:49:54
4                0       1  2020-04-19 14:53:57
```

Get familiar with data

[73]:
```
# Check for Dublicate
products_df.duplicated().value_counts()
```

[73]:
```
False    22893
dtype: int64
```

[74]:
```
print('Number of products: ', products_df.shape[0])
```

```
Number of products:  22893
```

[75]:
```
products_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22893 entries, 0 to 22892
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   uuid             22893 non-null  object
 1   user_uuid        22893 non-null  object
 2   category_uuid    22893 non-null  object
 3   city_uuid        22893 non-null  object
 4   is_new           22893 non-null  int64 
 5   is_homemade      22893 non-null  int64 
```

```
 6   original_price   22893 non-null  int64
 7   price            22893 non-null  int64
 8   comments_count   22893 non-null  int64
 9   bookmarks_count  22893 non-null  int64
 10  gender           22893 non-null  int64
 11  date             22893 non-null  object
dtypes: int64(7), object(5)
memory usage: 2.1+ MB
```

[76]:
```
products_df.describe()
```

[76]:

	is_new	is_homemade	original_price	price	comments_count	bookmarks_count	gender
count	22893.000	22893.000	22893.000	22893.000	22893.000	22893.000	22893.000
mean	0.505	0.040	1516705.384	226696.368	1.747	4.980	1.000
std	0.500	0.195	49605537.705	15531095.684	4.348	11.320	0.000
min	0.000	0.000	0.000	0.000	0.000	0.000	1.000
25%	0.000	0.000	60000.000	27000.000	0.000	0.000	1.000
50%	1.000	0.000	120000.000	50000.000	0.000	2.000	1.000
75%	1.000	0.000	200000.000	90000.000	2.000	5.000	1.000
max	1.000	1.000	2147483647.000	2147483647.000	115.000	272.000	1.000

[77]:
```
missing_percentage = products_df.isnull().sum() / products_df.shape[0] * 100
missing_percentage
```

[77]:
```
uuid              0.000
user_uuid         0.000
category_uuid     0.000
city_uuid         0.000
is_new            0.000
is_homemade       0.000
original_price    0.000
price             0.000
comments_count    0.000
bookmarks_count   0.000
gender            0.000
date              0.000
dtype: float64
```

Users with Most Items

[78]:
```
user_item = products_df['user_uuid'].value_counts().to_frame()
user_item.rename(columns = {'user_uuid': 'count' }, inplace = True)
user_item.head()
```

[78]:

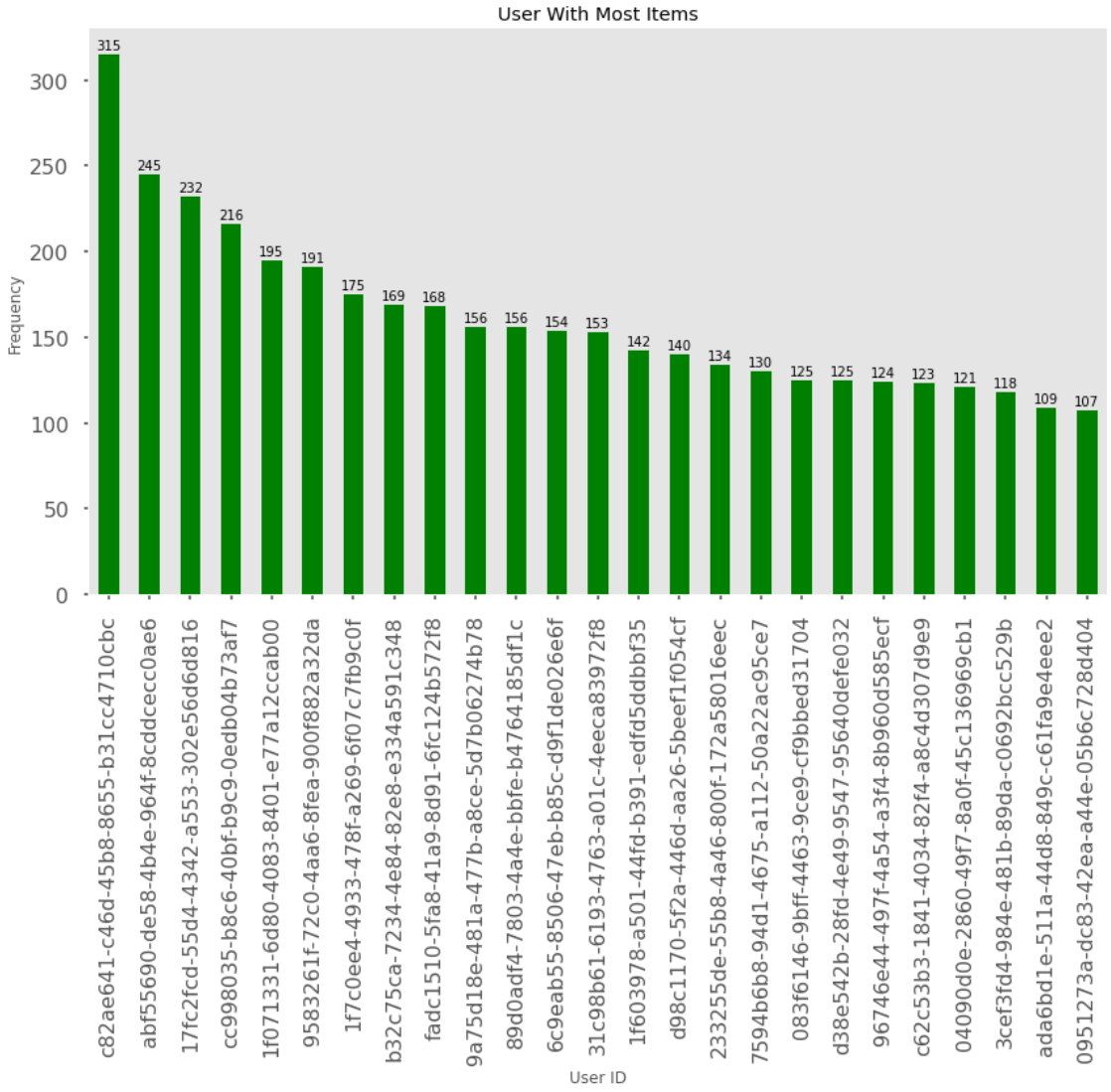
	count
c82ae641-c46d-45b8-8655-b31cc4710cbc	315
abf55690-de58-4b4e-964f-8cddcecc0ae6	245
17fc2fcd-55d4-4342-a553-302e56d6d816	232
cc998035-b8c6-40bf-b9c9-0edb04b73af7	216
1f071331-6d80-4083-8401-e77a12ccab00	195

[79]:
```
print('{} Users put their product online for sale.'.format(len(user_item)))
print('{} Users put more than one product online for sale.'.
 ↪format(len(user_item[user_item['count']>1])))
print('{} Users put Just one product online for sale.'.
 ↪format(len(user_item[user_item['count']<=1])))
print('{} is the maximum number of product by just a user for sale.'.
 ↪format(user_item['count'].max()))
```

```
2546 Users put their product online for sale.
1907 Users put more than one product online for sale.
639 Users put Just one product online for sale.
315 is the maximum number of product by just a user for sale.
```

[80]:
```
ax = products_df['user_uuid'].value_counts().nlargest(25).plot(kind='bar',
                                    figsize=(14,8),
                                    color = 'green',
                                    title = 'User With Most Items')

for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

# Turns off grid on the left Axis.
ax.grid(False)

ax.set_xlabel("User ID")
ax.set_ylabel("Frequency")
```

[80]:
```
Text(0, 0.5, 'Frequency')
```



Most popular Category

[81]:
```
products_df.head(2)
```

[81]:

	uuid	user_uuid	category_uuid	city_uuid	is_new	is_homemade	original_price	price	comments_count
0	33949611-a3e4-440a-83b1-6cc1a2602b8b	06abe0e5-c408-4a33-9d81-5bfacc891f15	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	0	0	0	15000	0
1	a55fbc5e-b0b1-43c8-b25b-57685d2ee242	16848d64-b555-4182-a5b4-c0ae1e850e01	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83					

```
1       1             0          550000  300000               0

   bookmarks_count  gender                 date
0                0       1  2019-12-19 02:42:28
1                1       1  2019-12-19 02:42:28
```

[82]:
```
products_df['category_uuid'].value_counts().to_frame()
```

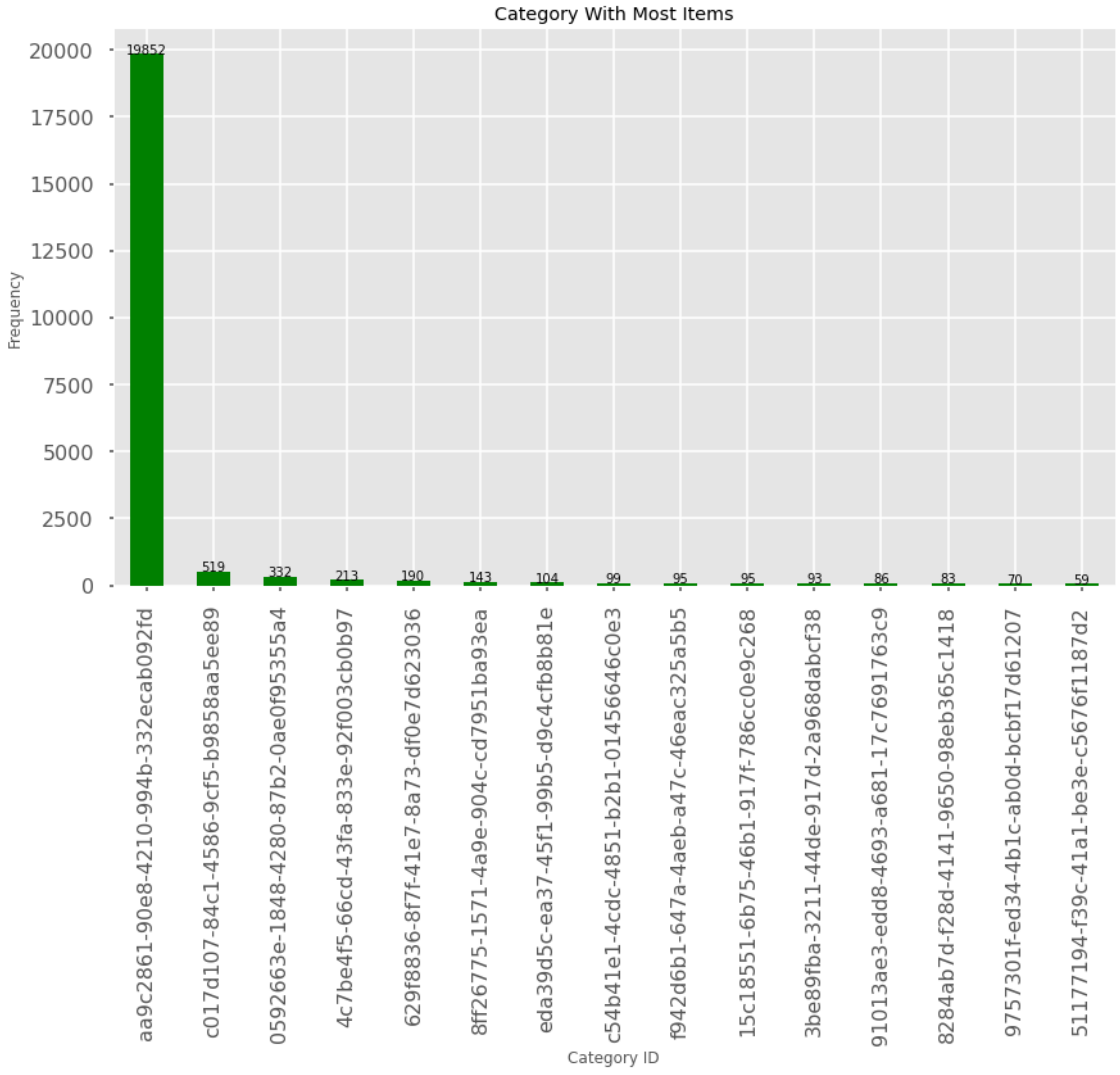
[82]:
```
                                     category_uuid
aa9c2861-90e8-4210-994b-332ecab092fd          19852
c017d107-84c1-4586-9cf5-b9858aa5ee89            519
0592663e-1848-4280-87b2-0ae0f95355a4            332
4c7be4f5-66cd-43fa-833e-92f003cb0b97            213
629f8836-8f7f-41e7-8a73-df0e7d623036            190
…                                               …
6ccc9a5b-2408-416d-885b-5cf5965b17e9              1
f652a45f-3cdb-4128-bcc7-f80c18c280c0              1
f72236d2-2362-412d-93d8-2da09203a3d4              1
07efeb2c-bef3-4ef6-af9d-ad9ed6a457b2              1
ce5873df-92d9-4269-a910-4ac498d74915              1

[95 rows x 1 columns]
```

[83]:
```
print('We have {} Categoty with items for sale.'.
 ↪format(products_df['category_uuid'].nunique()))
```

```
We have 95 Categoty with items for sale.
```

[84]:
```
ax = products_df['category_uuid'].value_counts().nlargest(15).plot(kind='bar',
                                    figsize=(14,8),
                                    color = 'green',
                                    title="Category With Most Items")

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("Category ID")
ax.set_ylabel("Frequency")
```

[84]:
```
Text(0, 0.5, 'Frequency')
```



Cities with most Item

[85]:
```
products_df['city_uuid'].value_counts().to_frame()
```

[85]:

	city_uuid
d29f1e68-56ce-476f-b09b-c960a4f5674f	6531
22972566-8739-40ca-bf19-a270d320dc83	5454
b4716b80-6d6b-4bdf-82f3-8a735760806c	1506
308873a5-249c-4bd3-b721-9e5aa54cd0ce	1007
017fea31-2ded-4a6f-b9cf-3e4a9d0fb799	886
…	…
7bee8007-467f-433c-9031-dec06f0e78f5	1
b1482ec4-90c2-4960-a9f2-b9326059b883	1
5e2dc764-d3b4-4bf1-8340-025e52924b77	1
7a66c2c3-7a93-46d0-b9b1-d805ca2eb0eb	1

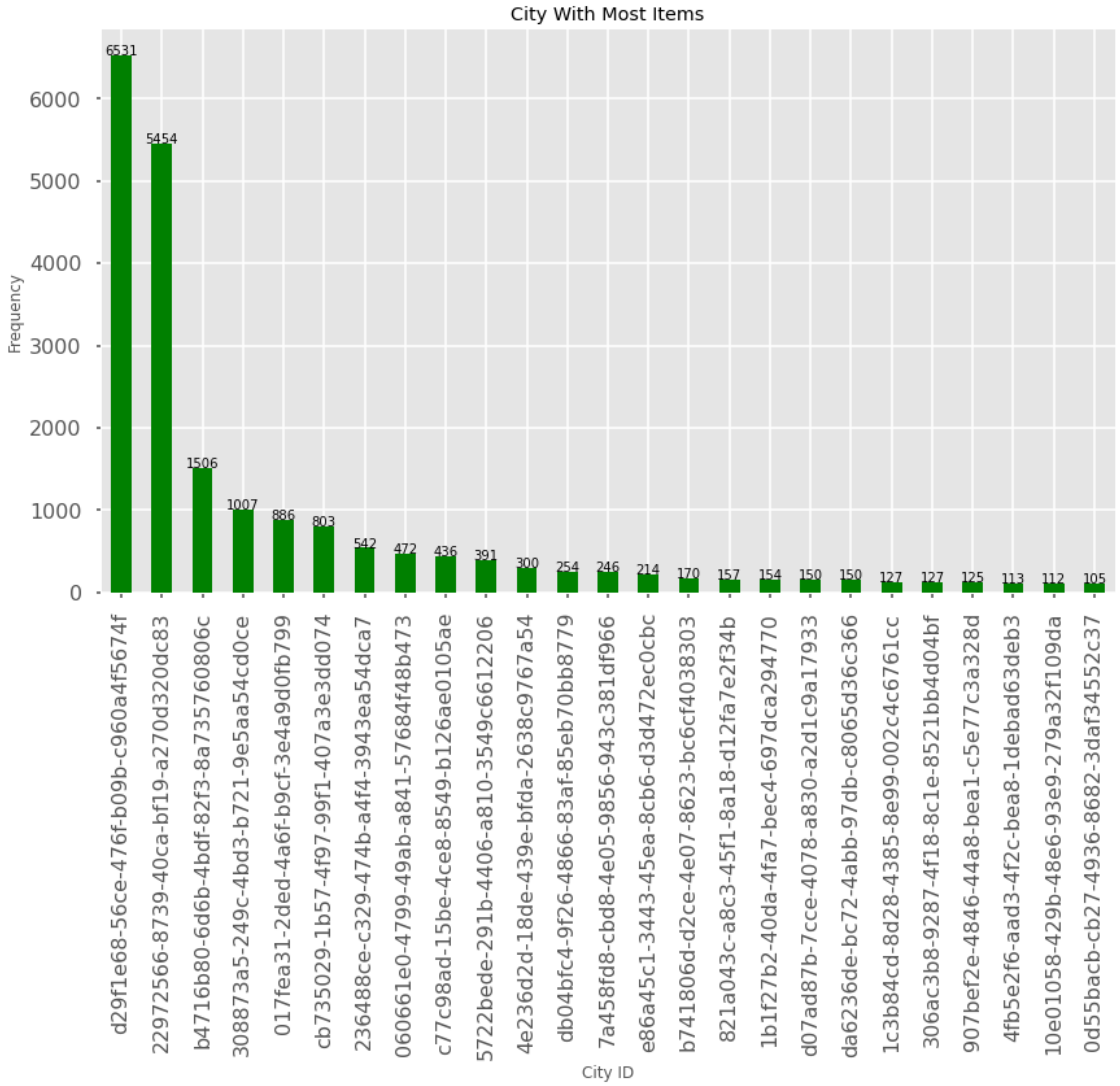
```
5da589dd-1647-4699-bc09-10ea7a13ed64          1

[193 rows x 1 columns]
```

[86]:
```
print('We have {} cities with items for sale.'.format(products_df['city_uuid'].
 ↪nunique()))
```

```
We have 193 cities with items for sale.
```

[87]:
```
ax = products_df['city_uuid'].value_counts().nlargest(25).plot(kind='bar',
                                    figsize=(14,8),
                                    color = 'green',
                                    title="City With Most Items")

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")


ax.set_xlabel("City ID")
ax.set_ylabel("Frequency")
```

[87]:
```
Text(0, 0.5, 'Frequency')
```

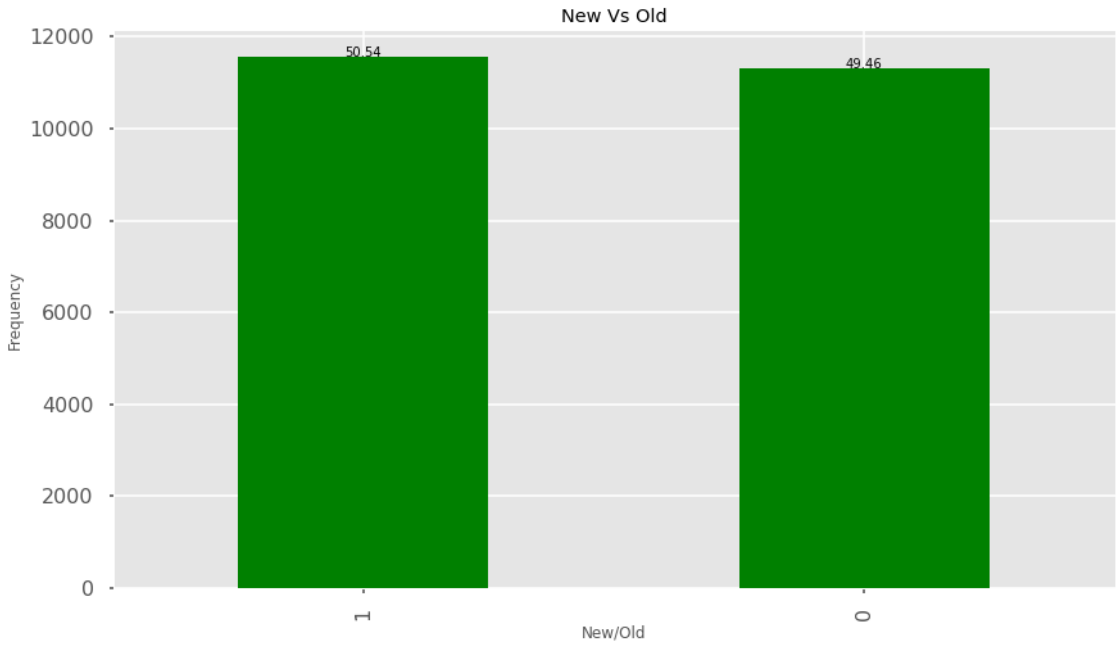


New vs Old

[88]:
```
ax = products_df['is_new'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    color = 'green',
                                    title="New Vs Old")
#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format((height/products_df.shape[0])*100),
            ha="center")
```

```
ax.set_xlabel("New/Old")
ax.set_ylabel("Frequency")
```

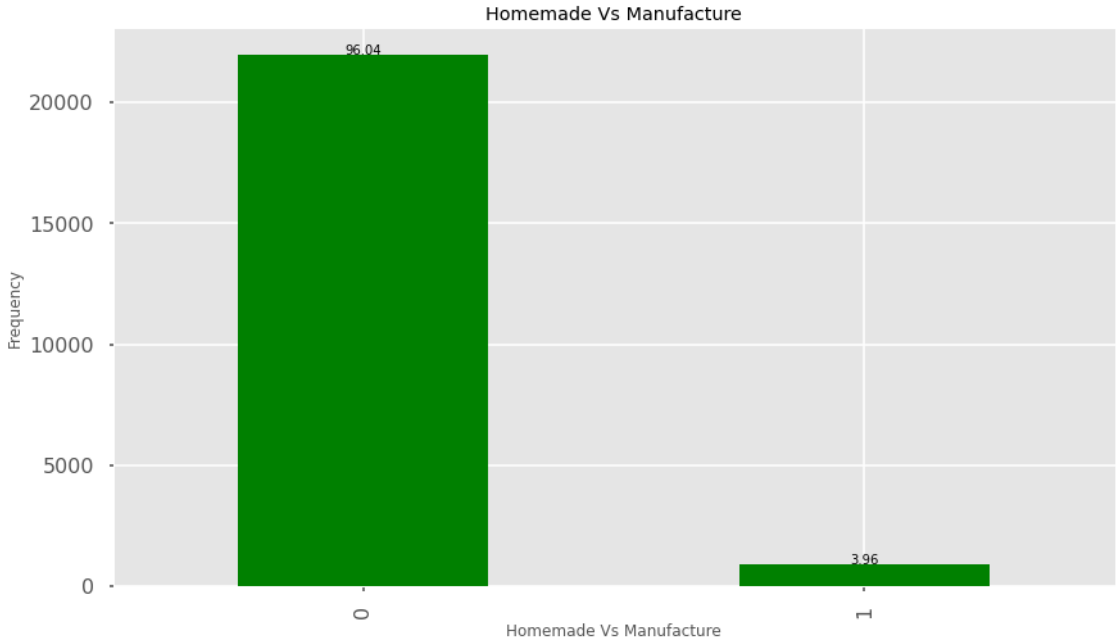
[88]: Text(0, 0.5, 'Frequency')



Homemade vs Manufactured

[89]:
```
ax = products_df['is_homemade'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    color = 'green',
                                    title="Homemade Vs Manufacture")

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format((height/products_df.shape[0])*100),
            ha="center")


ax.set_xlabel("Homemade Vs Manufactured")
ax.set_ylabel("Frequency")
```

[89]: Text(0, 0.5, 'Frequency')



Orginal Price Distribution

[90]:
```
products_df.original_price.describe()
```

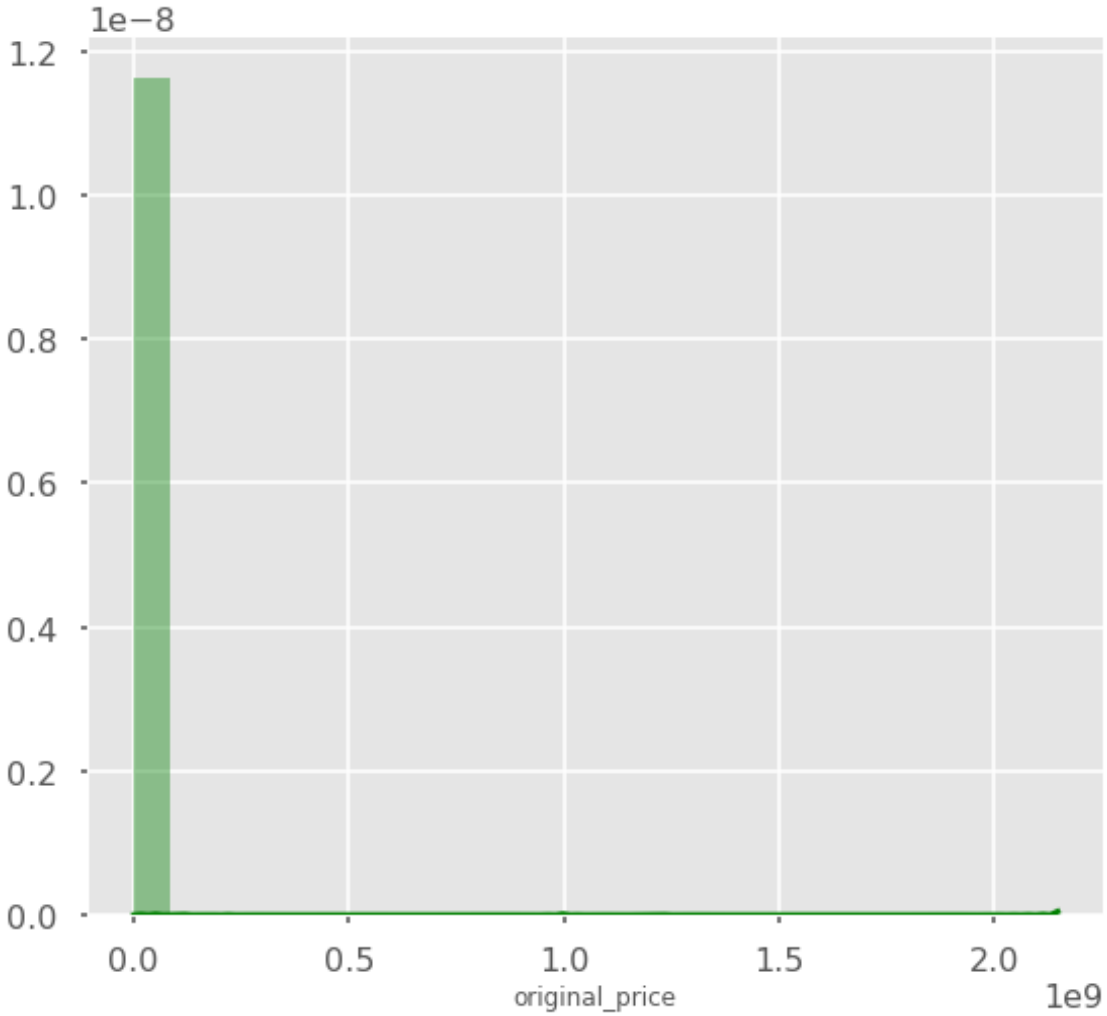
[90]:
```
count        22893.000
mean       1516705.384
std       49605537.705
min              0.000
25%          60000.000
50%         120000.000
75%         200000.000
max     2147483647.000
Name: original_price, dtype: float64
```

[91]:
```
product_p = products_df['original_price'].value_counts().to_frame().
 ↪reset_index()
product_p.rename(columns = {'index' : 'original_price', 'original_price':
 ↪'count'}, inplace = True)
product_p.head(5)
```

[91]:

	original_price	count
0	150000	1218
1	100000	1182
2	0	1143
3	120000	1093
4	50000	930

[92]:
```
plt.figure(figsize=(9, 8))
sns.distplot(products_df['original_price'], color='g', bins=25,␣
 ↪hist_kws={'alpha': 0.4});
```



[93]:
```
print('The highest price: ',products_df['original_price'].max())
print('The highest price: ',products_df['original_price'].min())
```

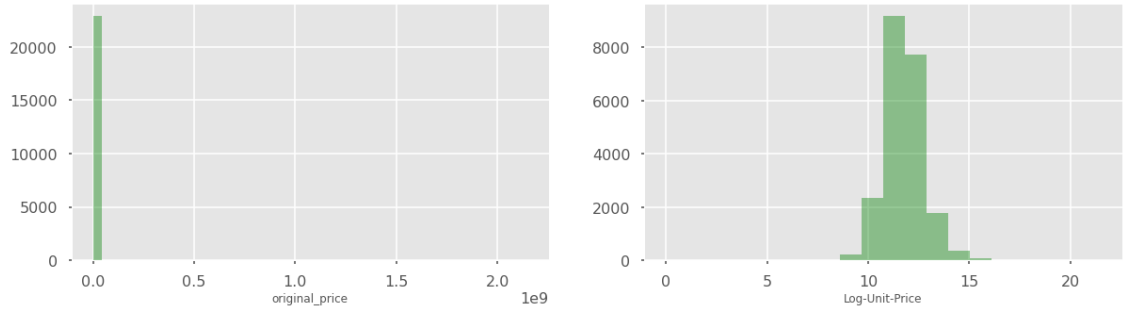
```
The highest price:  2147483647
The highest price:  0
```

It seems your histogram is heavily Long-Tailed. As we have prices up to 2*1e9 while the majority of data are much smaller, in the order of 1e6. So the bin=25 parameter does such that the first bin includes almost all of the data. We should use logarithmic bins

[94]:
```
price = products_df[['original_price','price']]
price = price[price['original_price']!=0]
```

[95]:
```
fig, ax = plt.subplots(1,2,figsize=(20,5))
sns.distplot(products_df.original_price, ax=ax[0], kde=False, color="g")
sns.distplot(np.log(price['original_price']), ax=ax[1], bins=20, color="g",␣
↪kde=False)
ax[1].set_xlabel("Log-Unit-Price")
```

[95]:
```
Text(0.5, 0, 'Log-Unit-Price')
```



Show Orginal Price distribution using custom bins

[96]:
```
products_df.head(2)
```

[96]:
```
                                   uuid                                 user_uuid  \
0  33949611-a3e4-440a-83b1-6cc1a2602b8b  06abe0e5-c408-4a33-9d81-5bfacc891f15
1  a55fbc5e-b0b1-43c8-b25b-57685d2ee242  16848d64-b555-4182-a5b4-c0ae1e850e01

                          category_uuid                                 city_uuid  \
0  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
1  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83

   is_new  is_homemade  original_price   price  comments_count  \
0       0            0               0   15000               0
1       1            0          550000  300000               0

   bookmarks_count  gender                 date
0                0       1  2019-12-19 02:42:28
1                1       1  2019-12-19 02:42:28
```

[97]:
```
print(' Number of Item with price of 0         : ' ,␣
↪len(products_df[products_df['original_price']==0]))
print(' Number of Item with price other than 0 : ' ,␣
↪len(products_df[products_df['original_price']!=0]))
```

```
Number of Item with price of 0         :  1143
Number of Item with price other than 0 :  21750
```

[98]:
```
product_non_zero_org_price = products_df[products_df['original_price']!=0]
product_non_zero_org_price.head(2)
```

[98]:
```
                                   uuid                                 user_uuid  \
1  a55fbc5e-b0b1-43c8-b25b-57685d2ee242  16848d64-b555-4182-a5b4-c0ae1e850e01
2  d895881f-6570-4ab6-9756-367c12b1ab44  16848d64-b555-4182-a5b4-c0ae1e850e01

                          category_uuid                                 city_uuid  \
1  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
2  8ff26775-1571-4a9e-904c-cd7951ba93ea  22972566-8739-40ca-bf19-a270d320dc83

   is_new  is_homemade  original_price   price  comments_count  \
1       1            0          550000  300000               0
2       1            0           92000   50000               1

   bookmarks_count  gender                 date
1                1       1  2019-12-19 02:42:28
2                4       1  2019-12-19 02:42:28
```

Unfortunately, We have **1143** item with price of zero! I believe this would make problem when we want to visualize our data, so I eliminate them for the visualization. We want to know the distribution of the item with prices and not zero price right?!

[99]:
```
product_non_zero_org_price['bucket_orginal'] = pd.
 ↪cut(product_non_zero_org_price.original_price, 100000 , include_lowest= True)
product_non_zero_org_price.head()
```

```
<ipython-input-99-a00262e7e817>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  product_non_zero_org_price['bucket_orginal'] =
pd.cut(product_non_zero_org_price.original_price, 100000 , include_lowest= True)
```

[99]:
```
                                   uuid                                 user_uuid  \
1  a55fbc5e-b0b1-43c8-b25b-57685d2ee242  16848d64-b555-4182-a5b4-c0ae1e850e01
2  d895881f-6570-4ab6-9756-367c12b1ab44  16848d64-b555-4182-a5b4-c0ae1e850e01
3  c0a45353-2453-442f-ae45-d741ac5ecdc3  16848d64-b555-4182-a5b4-c0ae1e850e01
4  00871ece-294a-4b1e-be51-2d72ac1205f7  16848d64-b555-4182-a5b4-c0ae1e850e01
5  87857c5d-e2df-4118-a822-e7285c78b814  16848d64-b555-4182-a5b4-c0ae1e850e01

                          category_uuid                                 city_uuid  \
1  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
2  8ff26775-1571-4a9e-904c-cd7951ba93ea  22972566-8739-40ca-bf19-a270d320dc83
3  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
4  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83
```

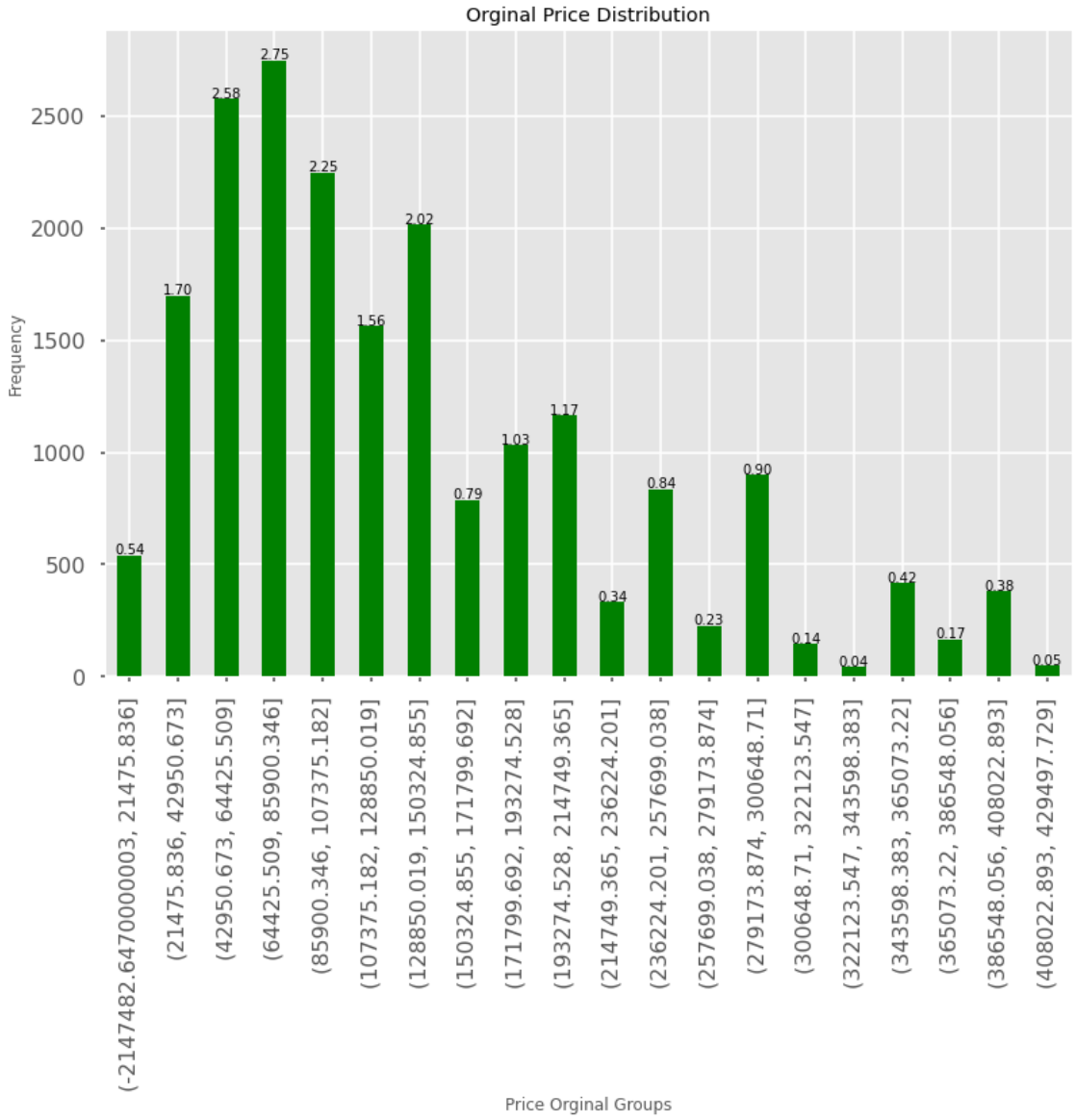
```
5  c017d107-84c1-4586-9cf5-b9858aa5ee89  22972566-8739-40ca-bf19-a270d320dc83

   is_new  is_homemade  original_price   price  comments_count  \
1       1            0          550000  300000               0
2       1            0           92000   50000               1
3       1            0          310000  230000               6
4       1            0          310000  230000               4
5       1            0          150000   98000               0

   bookmarks_count  gender                 date            bucket_orginal
1                1       1  2019-12-19 02:42:28  (536871.911, 558346.748]
2                4       1  2019-12-19 02:42:28   (85900.346, 107375.182]
3                1       1  2020-04-19 14:49:54   (300648.71, 322123.547]
4                0       1  2020-04-19 14:53:57   (300648.71, 322123.547]
5                0       1  2020-05-15 14:23:33  (128850.019, 150324.855]
```

[100]:
```
product_price_buckets = product_non_zero_org_price[['bucket_orginal','uuid']].
↪groupby('bucket_orginal').count()
product_price_buckets.head()
```

[100]:
```
                                  uuid
bucket_orginal
(-2147482.6470000003, 21475.836]   541
(21475.836, 42950.673]            1700
(42950.673, 64425.509]            2578
(64425.509, 85900.346]            2750
(85900.346, 107375.182]           2249
```

[101]:
```
ax = product_price_buckets.head(20).plot(kind='bar',title="Orginal Price␣
↪Distribution", legend = False , color = 'green')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format((height/product_price_buckets.shape[0])*100),
            ha="center")

ax.set_xlabel("Price Orginal Groups")
ax.set_ylabel("Frequency")
```

[101]:
```
Text(0, 0.5, 'Frequency')
```



Price Distribution

[102]:
```
products_df.price.describe()
```

[102]:
```
count        22893.000
mean        226696.368
std       15531095.684
min              0.000
25%          27000.000
50%          50000.000
75%          90000.000
max     2147483647.000
```

```
Name: price, dtype: float64
```

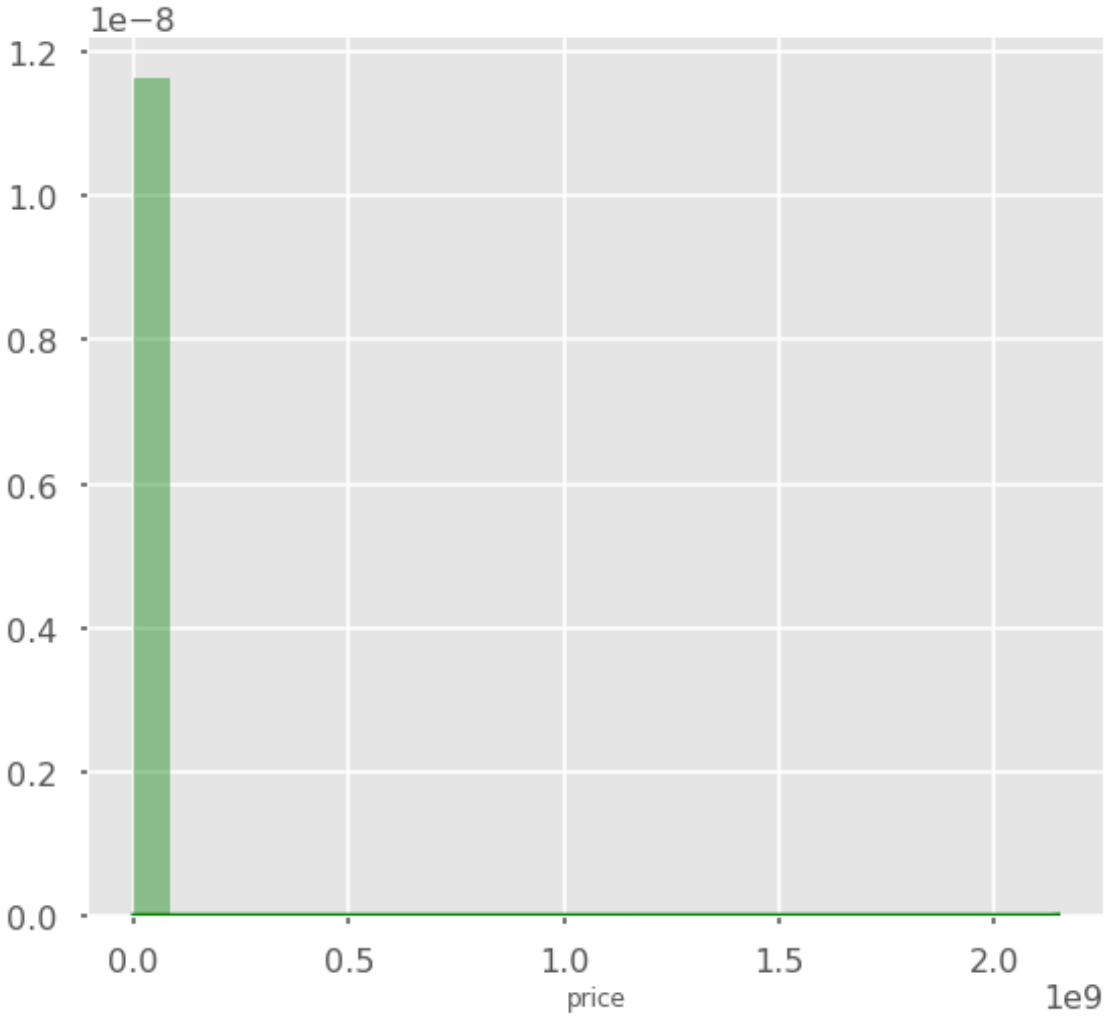
[103]:
```
product_price = products_df['price'].value_counts().to_frame().reset_index()
product_price.rename(columns = {'index' : 'price', 'price':'count'}, inplace =␣
↪True)
product_price
```

[103]:

	price	count
0	50000	1486
1	30000	1396
2	40000	1158
3	20000	1142
4	25000	1041
..	…	…
381	740000	1
382	55001	1
383	101000	1
384	1450000	1
385	75555	1

[386 rows x 2 columns]

[104]:
```
plt.figure(figsize=(9, 8))
sns.distplot(products_df['price'], color='g', bins=25, hist_kws={'alpha': 0.4});
```



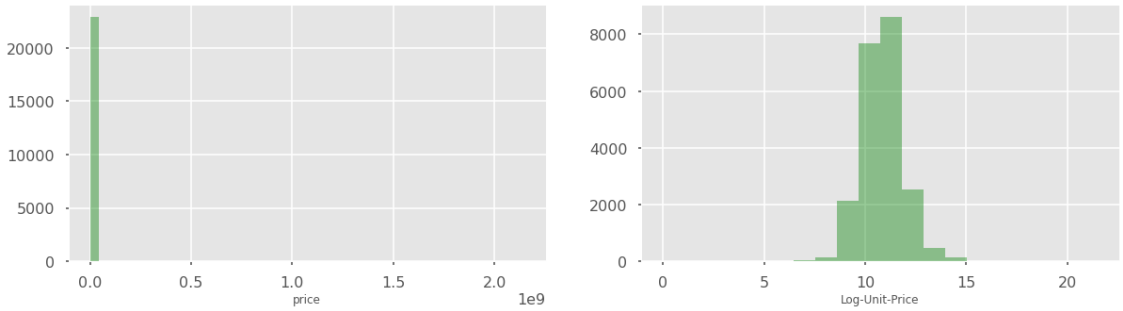
[105]:
```
print('The highest price: ',products_df['price'].max())
print('The highest price: ',products_df['price'].min())
```

```
The highest price:  2147483647
The highest price:  0
```

[106]:
```
price = products_df[['original_price','price']]
price = price[price['original_price']!=0]
```

[107]:
```
fig, ax = plt.subplots(1,2,figsize=(20,5))
sns.distplot(products_df.price, ax=ax[0], kde=False, color="g")
sns.distplot(np.log(price['price']), ax=ax[1], bins=20, color="g", kde=False)
ax[1].set_xlabel("Log-Unit-Price")
```

[107]:
```
Text(0.5, 0, 'Log-Unit-Price')
```



[108]:
```
print(' Number of Item with discounted price of 0         : ' ,␣
 ↪len(products_df[products_df['price']==0]))
print(' Number of Item with discounted price other than 0 : ' ,␣
 ↪len(products_df[products_df['price']!=0]))
```

```
 Number of Item with discounted price of 0         :  63
 Number of Item with discounted price other than 0 :  22830
```

[109]:
```
product_non_zero_dis_price = products_df[products_df['price']!=0]
product_non_zero_dis_price.head(2)
```

[109]:

	uuid	user_uuid	category_uuid	city_uuid	is_new	is_homemade	original_price	price	comments_count	bookmarks_count	gender	date
0	33949611-a3e4-440a-83b1-6cc1a2602b8b	06abe0e5-c408-4a33-9d81-5bfacc891f15	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	0	0	0	15000	0	0	1	2019-12-19 02:42:28
1	a55fbc5e-b0b1-43c8-b25b-57685d2ee242	16848d64-b555-4182-a5b4-c0ae1e850e01	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	1	0	550000	300000	0	1	1	2019-12-19 02:42:28

[110]:
```
product_non_zero_dis_price['bucket_discounted'] = pd.
 ↪cut(product_non_zero_dis_price.price, 100000 , include_lowest= True)
product_non_zero_dis_price.head()
```

```
<ipython-input-110-ace414813ee7>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  product_non_zero_dis_price['bucket_discounted'] =
pd.cut(product_non_zero_dis_price.price, 100000 , include_lowest= True)
```

[110]:

	uuid	user_uuid	category_uuid	city_uuid	is_new	is_homemade	original_price	price	comments_count	bookmarks_count	gender	date	bucket_discounted
0	33949611-a3e4-440a-83b1-6cc1a2602b8b	06abe0e5-c408-4a33-9d81-5bfacc891f15	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	0	0	0	15000	0	0	1	2019-12-19 02:42:28	(-2147482.6470000003, 21475.836]
1	a55fbc5e-b0b1-43c8-b25b-57685d2ee242	16848d64-b555-4182-a5b4-c0ae1e850e01	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	1	0	550000	300000	0	1	1	2019-12-19 02:42:28	(279173.874, 300648.71]
2	d895881f-6570-4ab6-9756-367c12b1ab44	16848d64-b555-4182-a5b4-c0ae1e850e01	8ff26775-1571-4a9e-904c-cd7951ba93ea	22972566-8739-40ca-bf19-a270d320dc83	1	0	92000	50000	1	4	1	2019-12-19 02:42:28	(42950.673, 64425.509]
3	c0a45353-2453-442f-ae45-d741ac5ecdc3	16848d64-b555-4182-a5b4-c0ae1e850e01	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	1	0	310000	230000	6	1	1	2020-04-19 14:49:54	(214749.365, 236224.201]
4	00871ece-294a-4b1e-be51-2d72ac1205f7	16848d64-b555-4182-a5b4-c0ae1e850e01	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	1	0	310000	230000	4	0	1	2020-04-19 14:53:57	(214749.365, 236224.201]

[111]:
```
product_dis_price_buckets =␣
↪product_non_zero_dis_price[['bucket_discounted','uuid']].
↪groupby('bucket_discounted').count()
product_dis_price_buckets.head()
```

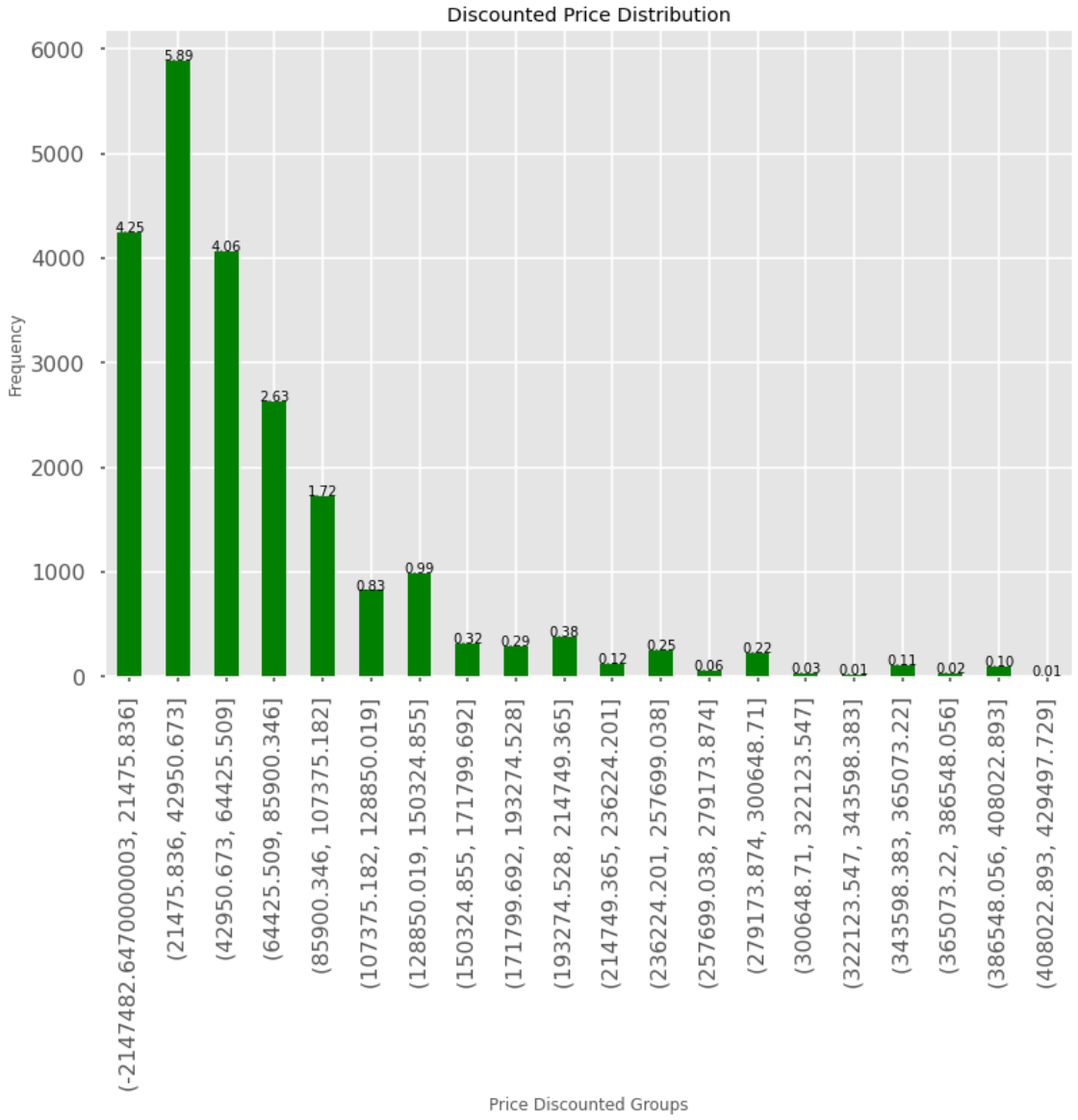
[111]:

bucket_discounted	uuid
(-2147482.6470000003, 21475.836]	4245

```
(21475.836, 42950.673]          5891
(42950.673, 64425.509]          4059
(64425.509, 85900.346]          2634
(85900.346, 107375.182]         1724
```

[112]:
```
ax = product_dis_price_buckets.head(20).plot(kind='bar',title="Discounted Price␣
↪Distribution", legend = False , color = 'green')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format((height/product_dis_price_buckets.shape[0])*100),
            ha="center")

ax.set_xlabel("Price Discounted Groups")
ax.set_ylabel("Frequency")
```

[112]:
```
Text(0, 0.5, 'Frequency')
```



Items with most Discount

[113]:
```
products_df['discount']= products_df['original_price'] - products_df['price']
```

[114]:
```
products_df.head(2)
```

[114]:

	uuid	user_uuid	\
0	33949611-a3e4-440a-83b1-6cc1a2602b8b	06abe0e5-c408-4a33-9d81-5bfacc891f15	
1	a55fbc5e-b0b1-43c8-b25b-57685d2ee242	16848d64-b555-4182-a5b4-c0ae1e850e01	

	category_uuid	city_uuid	\
0	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	

```
1  aa9c2861-90e8-4210-994b-332ecab092fd  22972566-8739-40ca-bf19-a270d320dc83

   is_new  is_homemade  original_price   price  comments_count  \
0       0            0               0   15000               0
1       1            0          550000  300000               0

   bookmarks_count  gender                 date  discount
0                0       1  2019-12-19 02:42:28    -15000
1                1       1  2019-12-19 02:42:28    250000
```

[115]:
```
products_df.sort_values('discount', inplace = True , ascending=False)
products_df.head()
```

[115]:
```
                                     uuid  \
10814  56d54b87-3a0f-4164-94d5-0217aa7bb8de
638    f7858eea-4370-410f-a77d-1260376e9f9f
8514   c619c0ae-8f1f-412d-ab85-a54145a4cbe0
8513   87a5959f-8fdb-4966-9989-311c805932b2
639    82fbd4f9-47a1-445a-bbe8-72b687d2ac79

                                  user_uuid  \
10814  a9502826-05f5-4551-84b7-4ab459715245
638    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0
8514   0d5f59d7-54db-46e5-866b-96ddf97f69db
8513   0d5f59d7-54db-46e5-866b-96ddf97f69db
639    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0

                             category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd
8514   aa9c2861-90e8-4210-994b-332ecab092fd
8513   aa9c2861-90e8-4210-994b-332ecab092fd
639    aa9c2861-90e8-4210-994b-332ecab092fd

                                  city_uuid  is_new  is_homemade  \
10814  1b1f27b2-40da-4fa7-bec4-697dca294770       1            0
638    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0
8514   308873a5-249c-4bd3-b721-9e5aa54cd0ce       1            0
8513   308873a5-249c-4bd3-b721-9e5aa54cd0ce       1            0
639    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0

       original_price  price  comments_count  bookmarks_count  gender  \
10814      2147483647  10000               0                1       1
638        2147483647  25000               0                1       1
8514       2147483647  35000               0                9       1
8513       2147483647  35000               0                0       1
639        2147483647  40000               0                3       1
```

```
                        date    discount
10814  2019-12-19 02:42:28  2147473647
638    2019-12-19 02:42:28  2147458647
8514   2019-12-19 02:42:28  2147448647
8513   2019-12-19 02:42:28  2147448647
639    2019-12-19 02:42:28  2147443647
```

[116]:
```
products_df['category_uuid'].nunique()
```

[116]:
```
95
```

[117]:
```
print('Maximum Discount: ' , products_df.discount.max())
print('Minimum Discount: ' , products_df.discount.min())
```

```
Maximum Discount:  2147473647
Minimum Discount:  -1400000
```

[118]:
```
discount_dis = pd.DataFrame(products_df.groupby([products_df["category_uuid"],␣
↪products_df['discount']])['discount'].size())
```

[119]:
```
discount_dis.rename(columns={'discount' : 'count'}, inplace = True)
```

[120]:
```
discount_dis.head()
```

[120]:
```
                                                 count
category_uuid                         discount
00f9ebd7-76fa-46aa-ba7e-a6eea5bc8f28  20000        1
                                      30000        1
                                      40000        1
                                      130000       1
0592663e-1848-4280-87b2-0ae0f95355a4 -130000       1
```

Most Generous Cities

[121]:
```
products_df.head(2)
```

[121]:
```
                                     uuid  \
10814  56d54b87-3a0f-4164-94d5-0217aa7bb8de
638    f7858eea-4370-410f-a77d-1260376e9f9f

                                  user_uuid  \
10814  a9502826-05f5-4551-84b7-4ab459715245
638    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0

                                category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd
```

```
                                    city_uuid  is_new  is_homemade  \
10814  1b1f27b2-40da-4fa7-bec4-697dca294770       1            0
638    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0

       original_price  price  comments_count  bookmarks_count  gender  \
10814      2147483647  10000               0                1       1
638        2147483647  25000               0                1       1

                     date    discount
10814  2019-12-19 02:42:28  2147473647
638    2019-12-19 02:42:28  2147458647
```

[122]:
```
df = products_df[["category_uuid", "city_uuid" , "discount"]]
```

[123]:
```
df.head()
```

[123]:
```
                                category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd
8514   aa9c2861-90e8-4210-994b-332ecab092fd
8513   aa9c2861-90e8-4210-994b-332ecab092fd
639    aa9c2861-90e8-4210-994b-332ecab092fd

                                    city_uuid    discount
10814  1b1f27b2-40da-4fa7-bec4-697dca294770  2147473647
638    d29f1e68-56ce-476f-b09b-c960a4f5674f  2147458647
8514   308873a5-249c-4bd3-b721-9e5aa54cd0ce  2147448647
8513   308873a5-249c-4bd3-b721-9e5aa54cd0ce  2147448647
639    d29f1e68-56ce-476f-b09b-c960a4f5674f  2147443647
```

[124]:
```
df.groupby(['city_uuid','category_uuid']).sum()
```

[124]:
```
discount
city_uuid                               category_uuid
00a6e24d-57b3-4284-97ad-82ab78f7e97d aa9c2861-90e8-4210-994b-332ecab092fd
180000
017fea31-2ded-4a6f-b9cf-3e4a9d0fb799 0592663e-1848-4280-87b2-0ae0f95355a4
2302400
                                     15c18551-6b75-46b1-917f-786cc0e9c268
3915000
                                     37978d09-0fc2-466e-b9d6-081eea95c8d5
191000
                                     3be89fba-3211-44de-917d-2a968dabcf38
470000
…
…
f9bedfa1-deaf-41cd-8c06-f6e010a3b487 4c7be4f5-66cd-43fa-833e-92f003cb0b97
```

```
58000
                                     aa9c2861-90e8-4210-994b-332ecab092fd
1974000
                                     f02e4eb6-c0a5-4a2a-a48f-687ae8407c67
348000
fb52139d-8de9-44e9-b4e6-f0e7b87d6cdf aa9c2861-90e8-4210-994b-332ecab092fd
20000
fb728ded-0321-4a80-b0f5-6ff0ca16bf74 aa9c2861-90e8-4210-994b-332ecab092fd
2945000

[885 rows x 1 columns]
```

[125]:
```
city_dis = df.groupby(['city_uuid'])['discount'].mean().to_frame()
city_dis.head()
```

[125]:
```
                                      discount
city_uuid
00a6e24d-57b3-4284-97ad-82ab78f7e97d  90000.000
017fea31-2ded-4a6f-b9cf-3e4a9d0fb799 223799.758
02bcddec-e7ed-4163-a6b5-dc5fe1e98702  18333.333
02f205d0-ef0e-4a8a-af7f-2b6274b64d30 132402.439
04965e51-5904-4b18-bd27-66e58f2871c5 169684.211
```

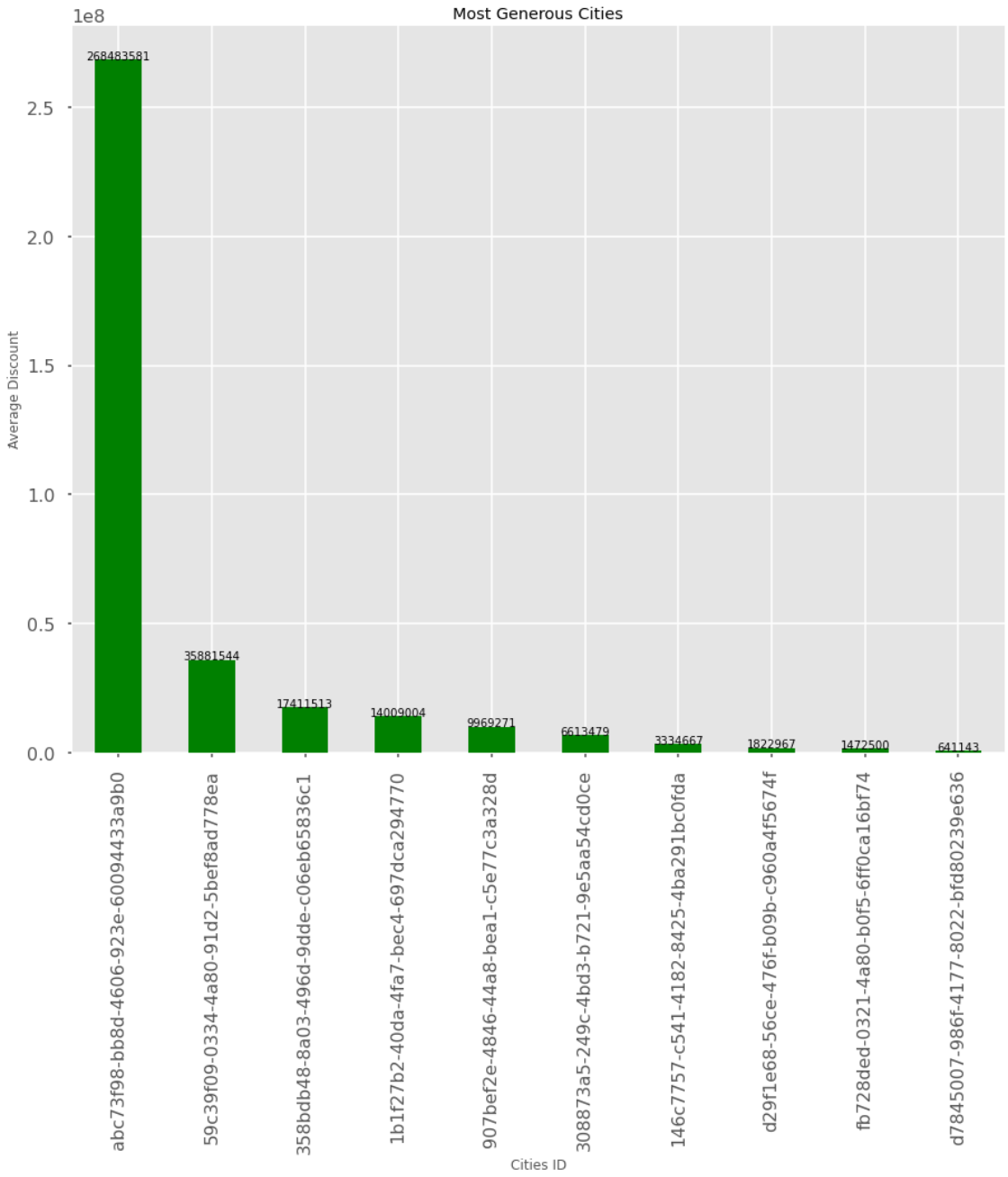
[126]:
```
city_dis.sort_values('discount', inplace = True , ascending=False)
city_dis.head(15)
```

[126]:
```
                                        discount
city_uuid
abc73f98-bb8d-4606-923e-60094433a9b0 268483580.875
59c39f09-0334-4a80-91d2-5bef8ad778ea  35881544.117
358bdb48-8a03-496d-9dde-c06eb65836c1  17411513.203
1b1f27b2-40da-4fa7-bec4-697dca294770  14009004.201
907bef2e-4846-44a8-bea1-c5e77c3a328d   9969271.136
308873a5-249c-4bd3-b721-9e5aa54cd0ce   6613479.234
146c7757-c541-4182-8425-4ba291bc0fda   3334666.667
d29f1e68-56ce-476f-b09b-c960a4f5674f   1822967.444
fb728ded-0321-4a80-b0f5-6ff0ca16bf74   1472500.000
d7845007-986f-4177-8022-bfd80239e636    641142.857
54006482-b391-4503-80a7-6669aab13fe8    625000.000
8aa7db48-54d8-48aa-bdd8-bc29e7a31090    512500.000
277bbddd-f4f0-4b2f-bb03-7b4bba54d55e    500000.000
cb735029-1b57-4f97-99f1-407a3e3dd074    406181.620
123d7827-de30-45bf-92b1-dbc191348682    405366.667
```

[127]:
```
ax = city_dis.head(10).plot(kind='bar', legend = False, color =␣
↪'green',figsize=(15,12), title='Most Generous Cities')
```

```
#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("Cities ID")
ax.set_ylabel("Average Discount")
```

[127]: Text(0, 0.5, 'Average Discount')



Category with most discount

[128]:
```
category_dis = df.groupby(['category_uuid'])['discount'].mean().to_frame()
category_dis.head()
```

[128]:

category_uuid	discount
00f9ebd7-76fa-46aa-ba7e-a6eea5bc8f28	55000.000

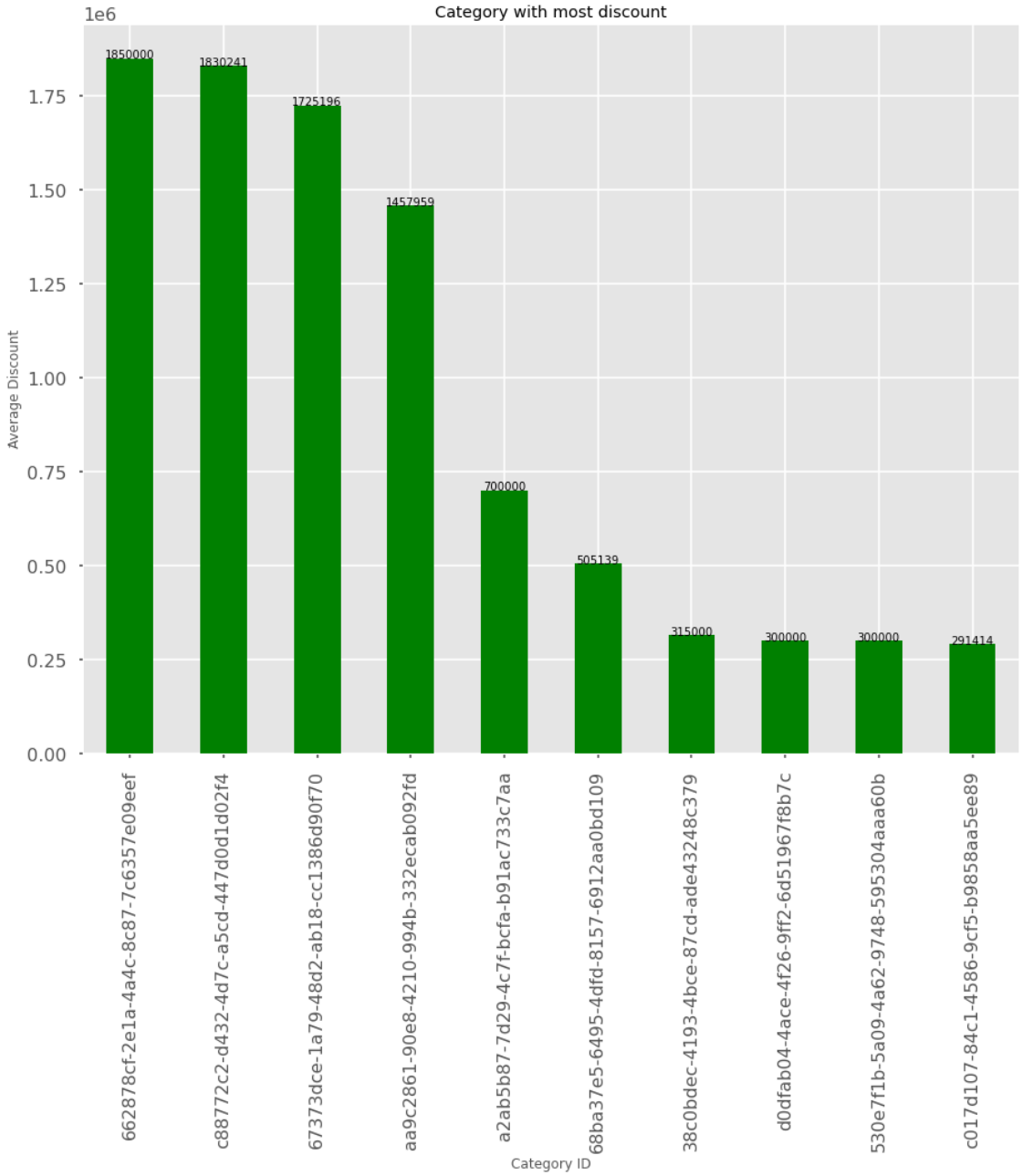
```
0592663e-1848-4280-87b2-0ae0f95355a4 134996.837
07efeb2c-bef3-4ef6-af9d-ad9ed6a457b2 150000.000
0da51c68-41fc-4d3f-9d7c-ed39829ab785  28428.571
109d7b9c-5f07-423c-8db8-8aa828d6989e 147191.489
```

[129]:
```
category_dis.sort_values('discount', inplace = True , ascending=False)
category_dis.head(15)
```

[129]:
```
                                        discount
category_uuid
662878cf-2e1a-4a4c-8c87-7c6357e09eef 1850000.000
c88772c2-d432-4d7c-a5cd-447d0d1d02f4 1830241.379
67373dce-1a79-48d2-ab18-cc1386d90f70 1725196.078
aa9c2861-90e8-4210-994b-332ecab092fd 1457958.763
a2ab5b87-7d29-4c7f-bcfa-b91ac733c7aa  700000.000
68ba37e5-6495-4dfd-8157-6912aa0bd109  505138.889
38c0bdec-4193-4bce-87cd-ade43248c379  315000.000
d0dfab04-4ace-4f26-9ff2-6d51967f8b7c  300000.000
530e7f1b-5a09-4a62-9748-595304aaa60b  300000.000
c017d107-84c1-4586-9cf5-b9858aa5ee89  291414.451
b6539b55-21dd-408d-8004-ec75c793cd6c  290625.000
6ccc9a5b-2408-416d-885b-5cf5965b17e9  250000.000
8284ab7d-f28d-4141-9650-98eb365c1418  217451.807
5391a538-1c30-4822-823f-f52653d36d45  210000.000
dec6f445-bd24-49db-8f50-e2dad31d5ce8  209280.000
```

[130]:
```
ax = category_dis.head(10).plot(kind='bar', legend = False, color =␣
↪'green',figsize=(15,12), title='Category with most discount')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("Category ID")
ax.set_ylabel("Average Discount")
```

[130]:
```
Text(0, 0.5, 'Average Discount')
```



Most Generous users

[131]:
```
user_dis = products_df.groupby(['user_uuid'])['discount'].mean().to_frame()
user_dis.head()
```

[131]:

user_uuid	discount
00016ef8-b6d3-4f67-8ba3-97146317f212	-50000.000
0008cf8e-57ec-4981-9c57-7dc6f817d215	64000.000

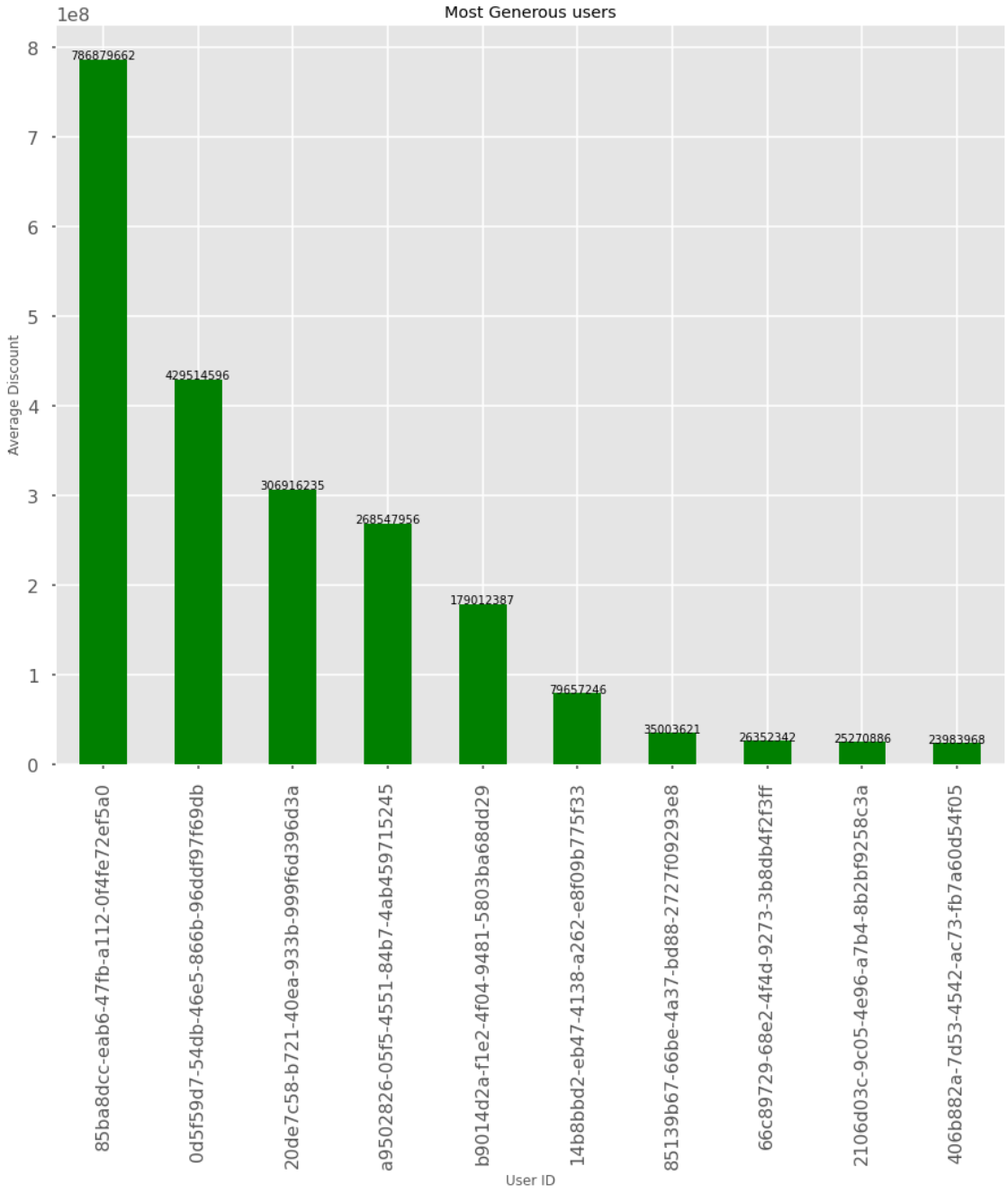
```
0023faeb-5abe-43fd-9c04-8f6701e55dfb  38333.333
004c68ec-c8e7-4e98-aab6-04fe47332881  20000.000
0081fc53-45e5-4422-a9bc-c1c9a9d0e0be   9000.000
```

[132]:
```
user_dis.sort_values('discount', inplace = True , ascending=False)
user_dis.head(15)
```

[132]:
```
                                        discount
user_uuid
85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0 786879661.750
0d5f59d7-54db-46e5-866b-96ddf97f69db 429514596.067
20de7c58-b721-40ea-933b-999f6d396d3a 306916235.286
a9502826-05f5-4551-84b7-4ab459715245 268547955.875
b9014d2a-f1e2-4f04-9481-5803ba68dd29 179012387.250
14b8bbd2-eb47-4138-a262-e8f09b775f33  79657246.185
85139b67-66be-4a37-bd88-2727f09293e8  35003620.690
66c89729-68e2-4f4d-9273-3b8db4f2f3ff  26352342.105
2106d03c-9c05-4e96-a7b4-8b2bf9258c3a  25270886.449
406b882a-7d53-4542-ac73-fb7a60d54f05  23983968.220
1c2b689d-ff45-4270-b180-dfca59a32d5b  23000000.000
ff8554be-c603-41b1-ae35-1b81b85510ee  11450000.000
0285176c-96b3-4a87-be59-34453e0fe0ff   9400000.000
dc77decd-1be3-49ee-a025-1c64cbb9a654   5448284.186
7ef9c89b-382e-4870-8c29-4c97e12e419d   4700000.000
```

[133]:
```
ax = user_dis.head(10).plot(kind='bar', legend = False, color =␣
↪'green',figsize=(15,12), title='Most Generous users')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("User ID")
ax.set_ylabel("Average Discount")
```

[133]:
```
Text(0, 0.5, 'Average Discount')
```



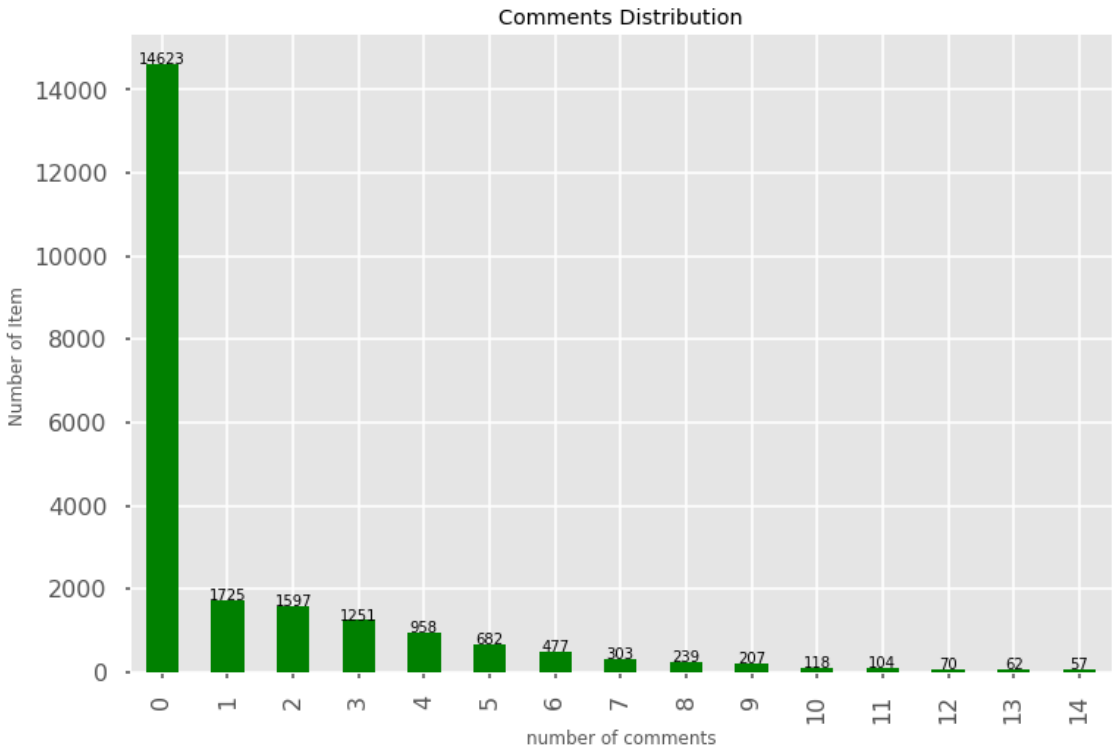
[ ]:

Items with most Comments and Items with most bookmarks (Hot Items)

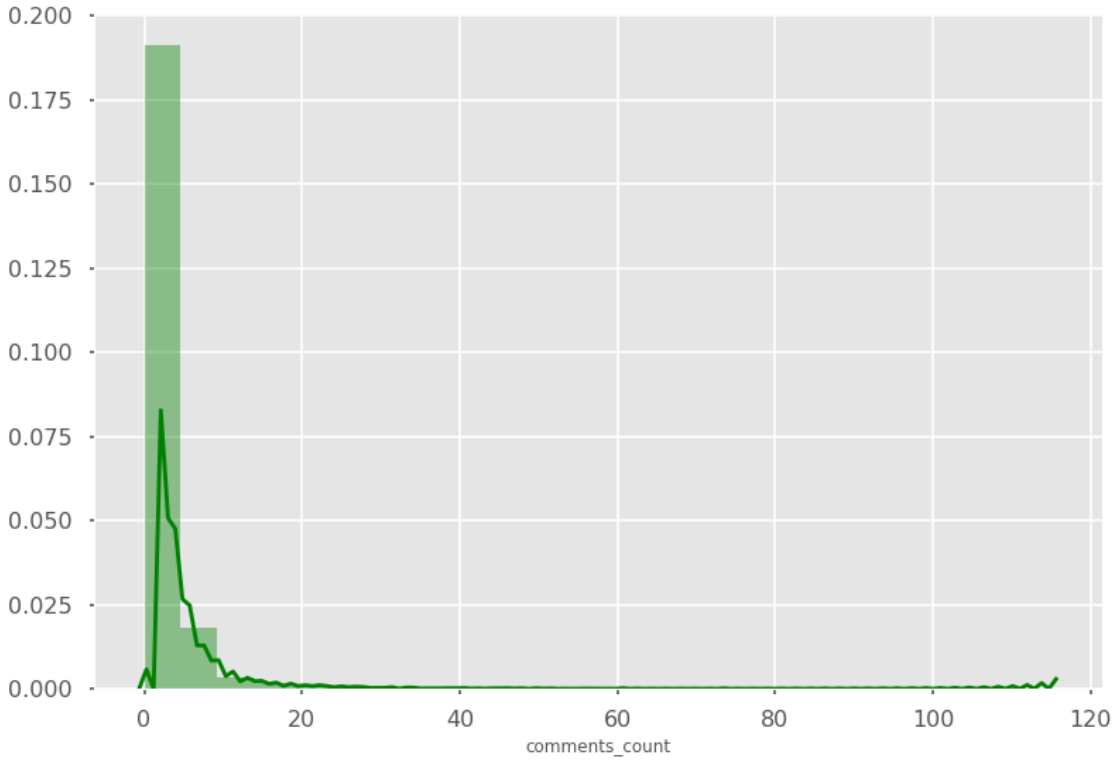
Comments

[134]:
```
ax = products_df['comments_count'].value_counts().head(15).plot(kind='bar',␣
↪legend = False, color = 'green',figsize=(12,8), title='Comments␣
↪Distribution')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("number of comments")
ax.set_ylabel("Number of Item ")
```

[134]:
```
Text(0, 0.5, 'Number of Item ')
```



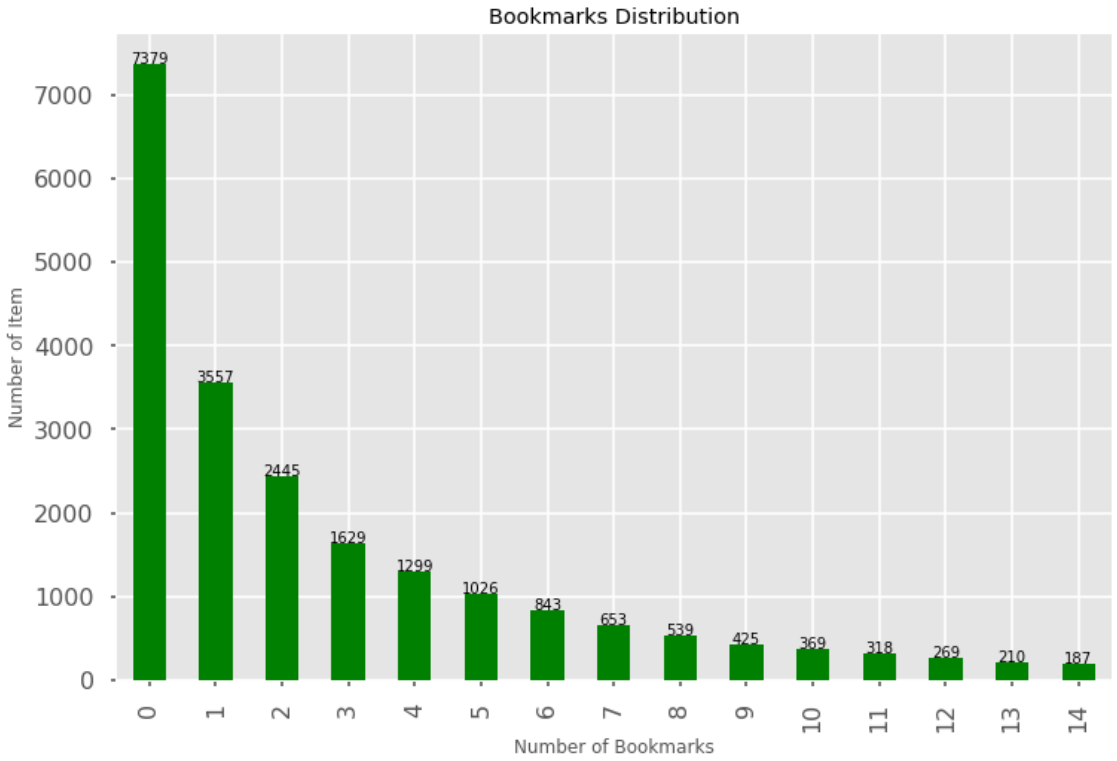
[135]:
```
plot = sns.distplot(products_df.comments_count,
                    bins=25,
                    color="green")
```



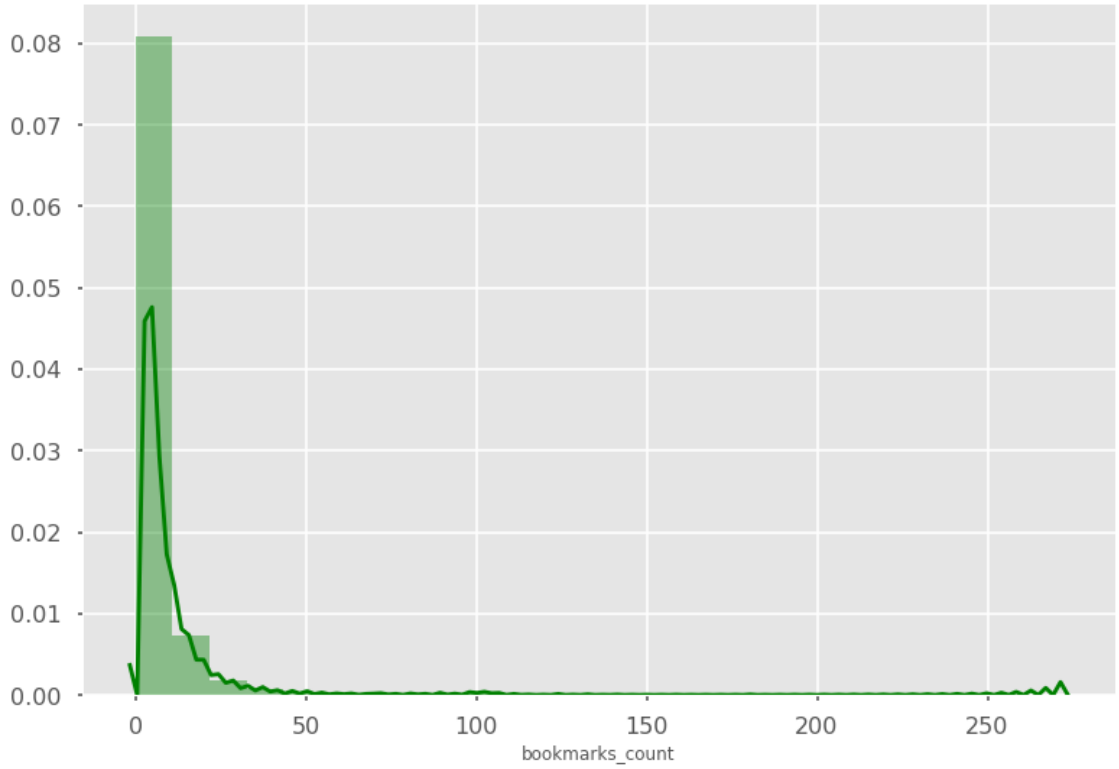
bookmarks

[136]:
```
ax = products_df['bookmarks_count'].value_counts().head(15).plot(kind='bar',␣
↪legend = False, color = 'green',figsize=(12,8), title='Bookmarks␣
↪Distribution')

#annotate axis = seaborn axis
for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.0f}'.format(height),
            ha="center")

ax.set_xlabel("Number of Bookmarks")
ax.set_ylabel("Number of Item")
```

[136]:
```
Text(0, 0.5, 'Number of Item')
```



[137]:
```
plot = sns.distplot(products_df.bookmarks_count,
                    bins=25,
                    color="green")
```



[ ]:

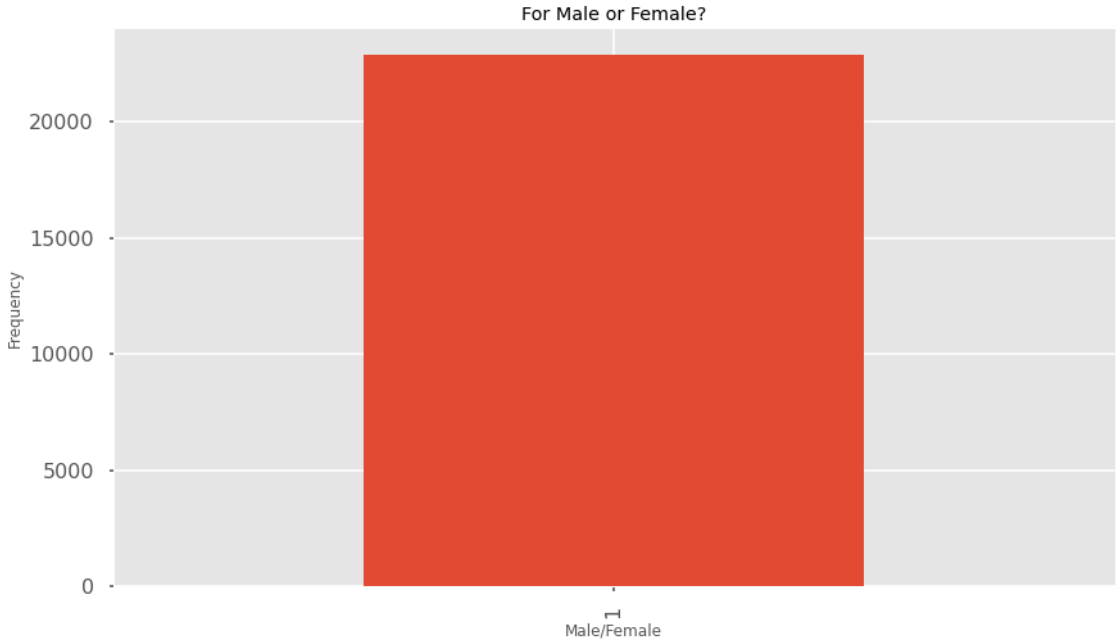
Gender Distiribution of the items

[138]:
```
products_df.columns
```

[138]:
```
Index(['uuid', 'user_uuid', 'category_uuid', 'city_uuid', 'is_new',
       'is_homemade', 'original_price', 'price', 'comments_count',
       'bookmarks_count', 'gender', 'date', 'discount'],
      dtype='object')
```

[139]:
```
ax = products_df['gender'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    title="For Male or Female?")
ax.set_xlabel("Male/Female")
ax.set_ylabel("Frequency")
```

[139]:
```
Text(0, 0.5, 'Frequency')
```



[140]:
```
products_df['gender'].value_counts()
```

[140]:
```
1    22893
Name: gender, dtype: int64
```

Wow! All of the clothes are for men only!

[ ]:
```

```

Month and Categories

[141]:
```
products_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22893 entries, 10814 to 12413
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   uuid             22893 non-null  object
 1   user_uuid        22893 non-null  object
 2   category_uuid    22893 non-null  object
 3   city_uuid        22893 non-null  object
 4   is_new           22893 non-null  int64
 5   is_homemade      22893 non-null  int64
 6   original_price   22893 non-null  int64
 7   price            22893 non-null  int64
 8   comments_count   22893 non-null  int64
 9   bookmarks_count  22893 non-null  int64
```

```
 10  gender           22893 non-null  int64
 11  date             22893 non-null  object
 12  discount         22893 non-null  int64
dtypes: int64(8), object(5)
memory usage: 2.4+ MB
```

[142]:
```
products_df["date"] = pd.to_datetime(products_df.date, cache=True)

print("Start Of The Period: " , products_df.date.min())
print("End Of The Period  : " , products_df.date.max() )
print("Time Period        : " , products_df.date.max() - products_df.date.min())
```

```
Start Of The Period:  2019-12-19 02:42:28
End Of The Period  :  2020-09-16 14:29:20
Time Period        :  272 days 11:46:52
```

[143]:
```
categories_month_count = pd.DataFrame(products_df.groupby([pd.
↪to_datetime(products_df["date"]).dt.to_period('M'), products_df.
↪category_uuid])['category_uuid'].
                                    count())
```

[144]:
```
categories_month_count.rename(columns = {'category_uuid' : 'count'},inplace =␣
↪True)
categories_month_count
```

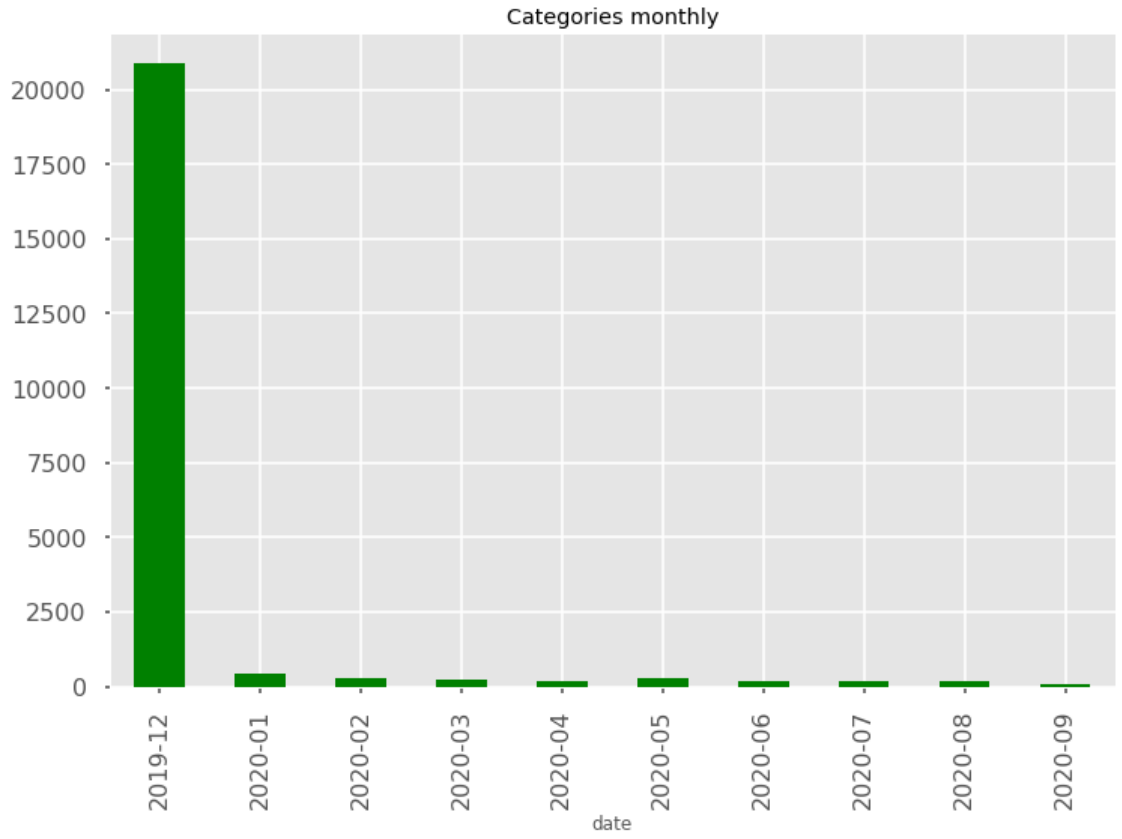
[144]:

		count
date	category_uuid	
2019-12	0592663e-1848-4280-87b2-0ae0f95355a4	148
	0da51c68-41fc-4d3f-9d7c-ed39829ab785	1
	109d7b9c-5f07-423c-8db8-8aa828d6989e	2
	109e52c9-b8ce-4c24-b693-56a76a361a0d	3
	15c18551-6b75-46b1-917f-786cc0e9c268	80
…		…
2020-09	c2d8bd72-3e93-4db8-8b2f-47825445928d	1
	c54b41e1-4cdc-4851-b2b1-01456646c0e3	7
	d06ba9f8-d3d8-403e-9807-ca9be9122ef7	3
	eda39d5c-ea37-45f1-99b5-d9c4cfb8b81e	1
	ffa7c9df-e9b8-44d8-a0a6-c1cd0de75132	2

[370 rows x 1 columns]

[145]:
```
products_df.groupby(pd.to_datetime(products_df["date"]).dt.
↪to_period('M'))['uuid'].count().plot(kind='bar', legend = False, color =␣
↪'green',figsize=(12,8),
                                                                        ␣
↪                title='Categories monthly')
```

[145]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x284df806370>
```

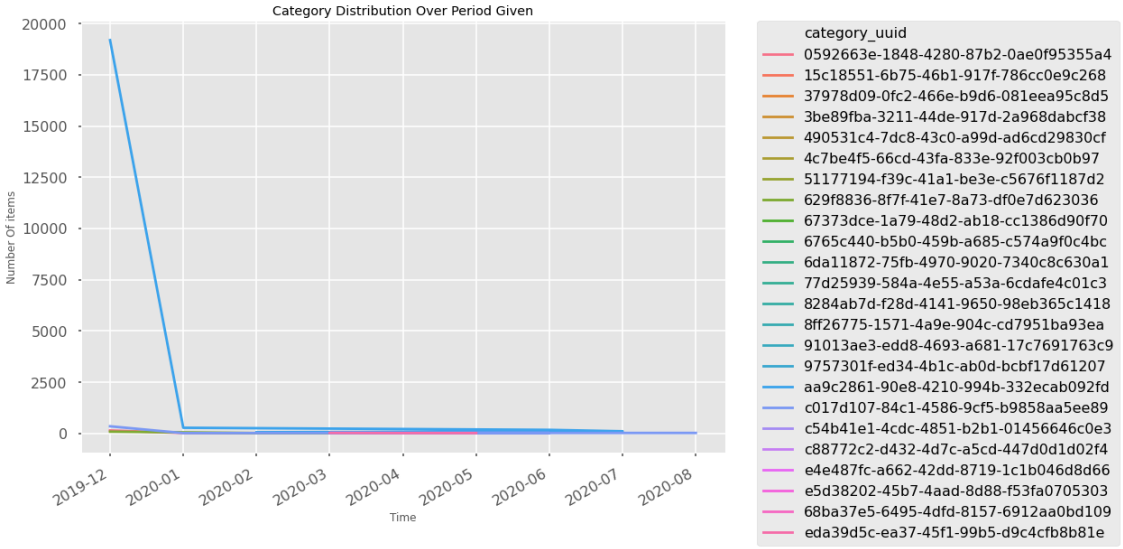


[146]:
```
categories_month_count.reset_index(inplace = True)
categories_month_count =␣
 ↪categories_month_count[categories_month_count['count']>15]
```

[147]:
```
li_date = categories_month_count.date.unique()
```

[148]:
```
plot = sns.lineplot(x=categories_month_count['date'].astype(str), y='count',␣
 ↪hue='category_uuid', data=categories_month_count)


plot.set_xticklabels(li_date,rotation=30, horizontalalignment='right')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

# Add title and axis names
plt.title('Category Distribution Over Period Given')
plt.xlabel('Time', fontsize='12')
plt.ylabel('Number Of items')
```

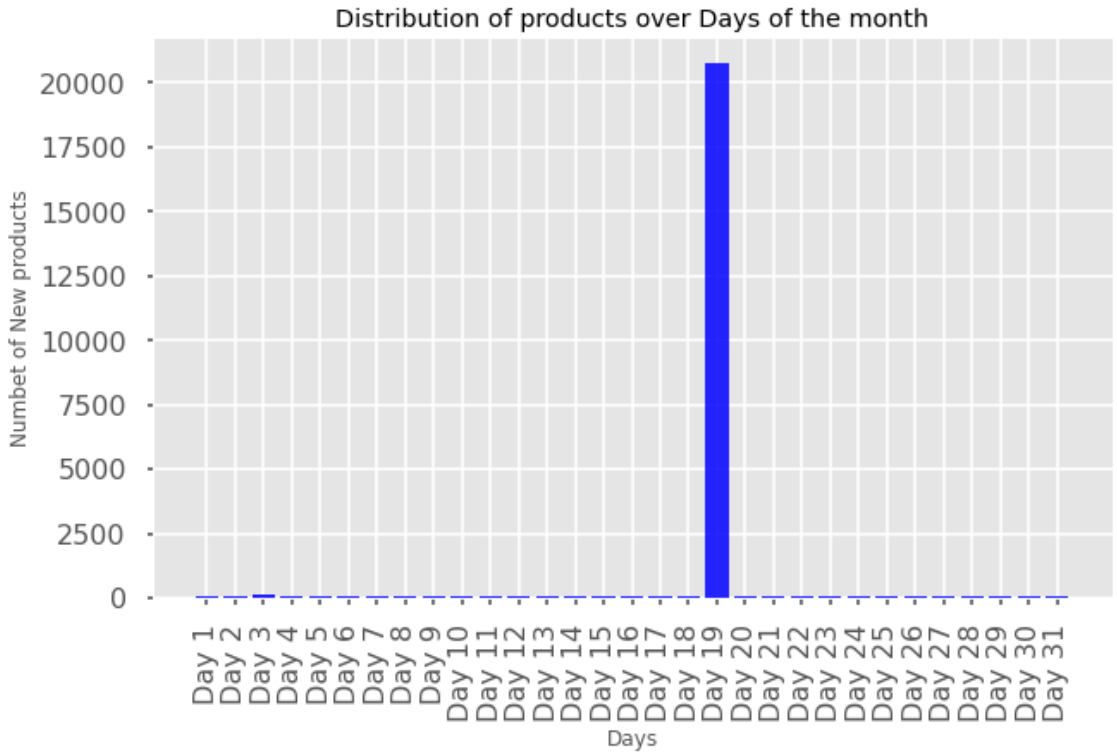
[148]:
```
Text(0, 0.5, 'Number Of items')
```



[149]:
```
# plot data
# fig, ax = plt.subplots(figsize=(20,10))
# pd.DataFrame(products_df.groupby([pd.to_datetime(products_df["date"]).dt.
 ↪to_period('M'), products_df.category_uuid],sort=True)['category_uuid'].
#              count()).unstack().plot(ax=ax, legend = False)
```

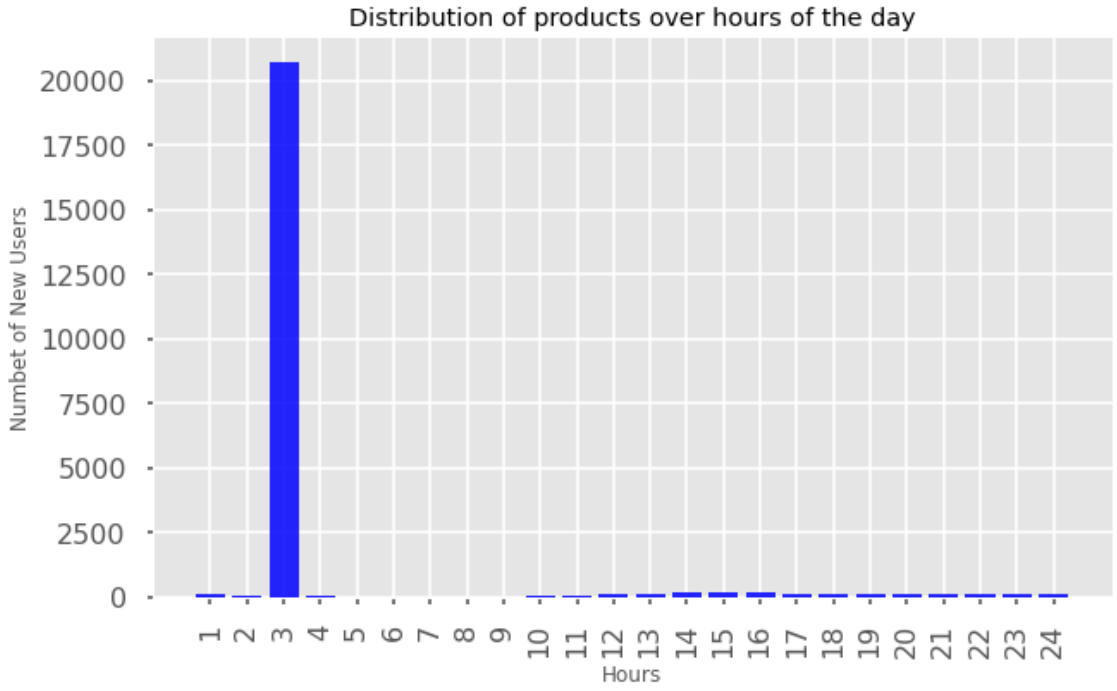
[150]:
```
# Daily distribution count
daily_count = products_df.groupby(products_df["date"].dt.day, as_index = False).
 ↪count()

fig= plt.figure(figsize=(10,6))

# Choose the names of the bars
bars = []
for i in range(1,32):
    bars.append('Day '+str(i))

y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, daily_count.date, color = 'Blue', alpha = 0.85)

# Create names on the x-axis
plt.xticks(y_pos, bars, rotation=90)
plt.yticks()

# Add title and axis names
plt.title('Distribution of products over Days of the month')
```

```
plt.xlabel('Days', fontsize='12')
plt.ylabel('Numbet of New products')


# Show graphic
plt.savefig('Images/Distribution of new products over days.png',bbox_inches =␣
 ↪'tight')
plt.show()
```



[151]:
```
# Monthly distribution count
hourly_count = products_df.groupby(products_df["date"].dt.hour, as_index =␣
 ↪False).count()

fig= plt.figure(figsize=(10,6))

# Choose the names of the bars
bars = []
for i in range(1,25):
    bars.append(str(i))

y_pos = np.arange(len(bars))
```

```
# Create bars
plt.bar(y_pos, hourly_count.date,color = 'Blue', alpha = 0.85)

# Create names on the x-axis
plt.xticks(y_pos, bars, rotation=90)
plt.yticks()

# Add title and axis names
plt.title('Distribution of products over hours of the day')
plt.xlabel('Hours', fontsize='12')
plt.ylabel('Numbet of New Users')


# Show graphic
plt.savefig('Images/Distribution of products over hours.png',bbox_inches =␣
 ↪'tight')
plt.show()
```

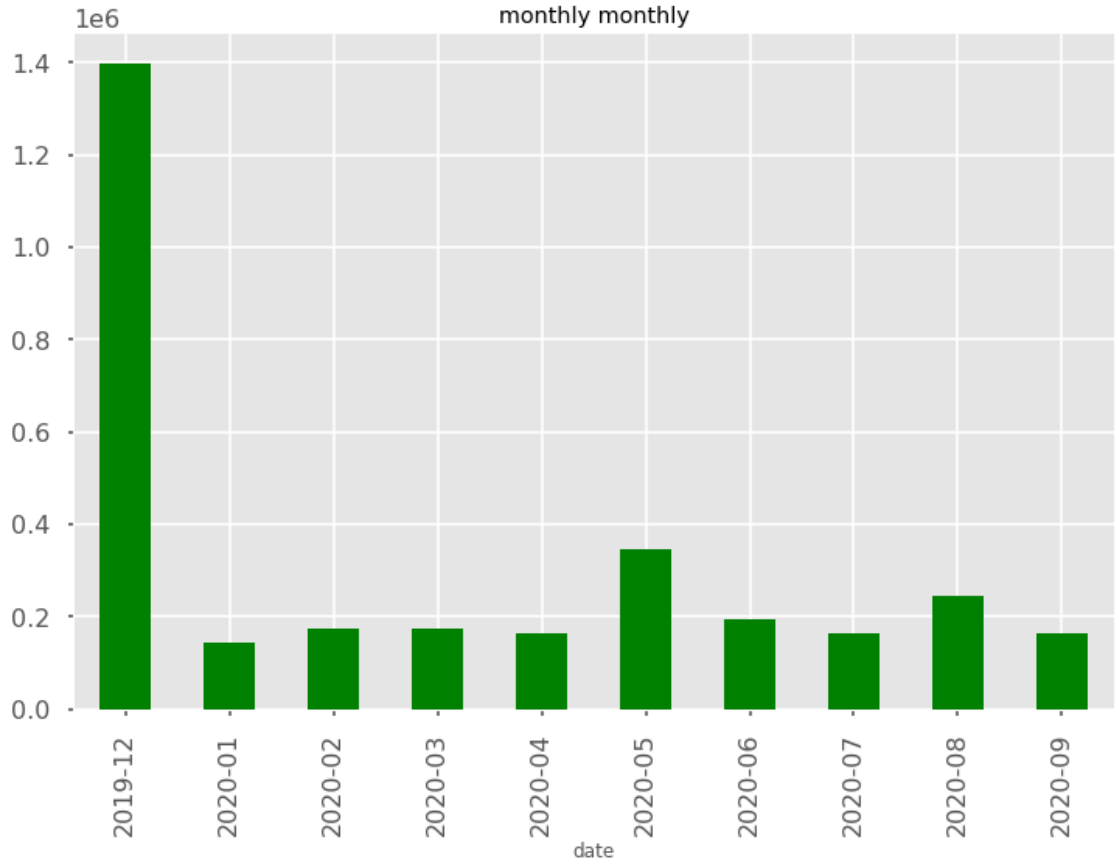


[ ]:

Month and Discount

[152]:
```
categories_month_count = pd.DataFrame(products_df.groupby(pd.
 ↪to_datetime(products_df["date"]).dt.to_period('M'))['discount'].mean()).
 ↪plot(kind='bar', legend = False,
```

```
                                                                                ␣
↪                                                          color =␣
↪'green',figsize=(12,8),
                                                                                ␣
↪                                                          title='monthly␣
↪monthly')
```



[153]:
```
products_df.groupby(pd.to_datetime(products_df["date"]).dt.
↪to_period('M'))['discount'].mean().to_frame()
```

[153]:

date	discount
2019-12	1396233.645
2020-01	145243.990
2020-02	172504.983
2020-03	172838.174
2020-04	162410.372
2020-05	344307.865
2020-06	194802.956

```
2020-07  164151.163
2020-08  244211.765
2020-09  164869.118
```

[ ]:

0.5 Purchase DataFrame

To Do List

1. Users with Most purchase
2. Seasons with Most Purchase

[154]:
```
purchase_df = pd.read_csv('komodaa/purchases.csv')
purchase_df.head()
```

[154]:

	buyer_uuid	product_uuid	count	date
0	372fe527-470c-48a3-a95e-fe32cc4bcc15	99863785-21e9-4ebc-ae7a-8c558d8421b4	1	2018-02-15 17:38:52
1	970ef8d9-c50e-4821-9687-145379af2b9a	e68a5832-7292-4a54-ad92-02f31a46494c	1	2017-02-16 00:47:16
2	970ef8d9-c50e-4821-9687-145379af2b9a	ecd1b68f-9db4-49f5-bac5-5679d9f9d784	1	2018-03-01 15:10:23
3	0bc6742f-d58a-4539-b85b-62bc3463033b	af59299c-2d08-4993-9be0-cfd7e641ae0d	1	2017-12-17 12:49:20
4	233255de-55b8-4a46-800f-172a58016eec	0cd2b781-bd44-40cf-9422-a50dac21f4b5	1	2018-03-12 20:26:13

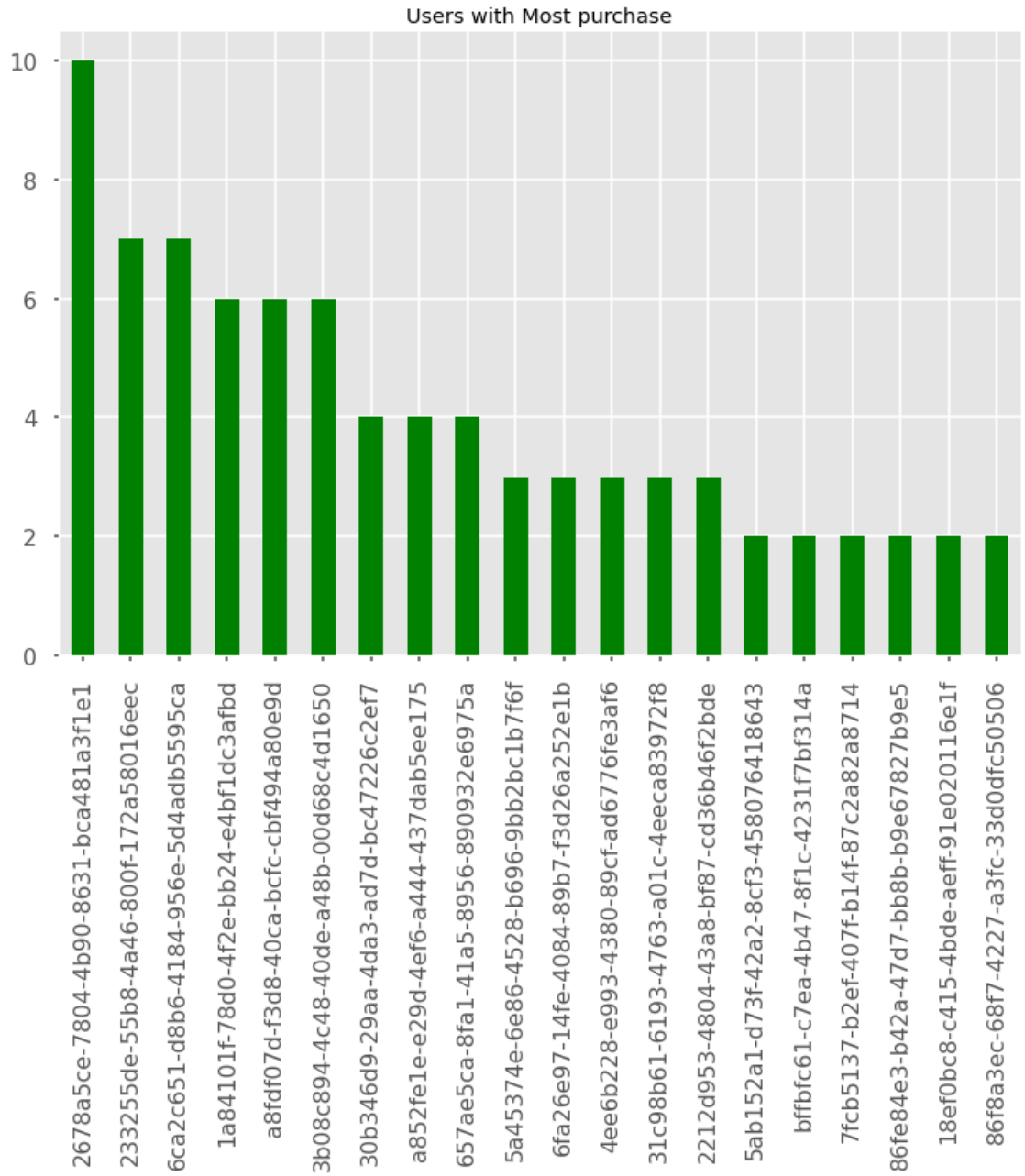
[155]:
```
purchase_df['count'].value_counts()
```

[155]:
```
1    184
Name: count, dtype: int64
```

Users with Most purchase

[156]:
```
purchase_df.buyer_uuid.value_counts().head(20).plot(kind='bar', legend = False,␣
↪color = 'green',figsize=(12,8), title='Users with Most purchase')
```

[156]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x284e1743eb0>
```



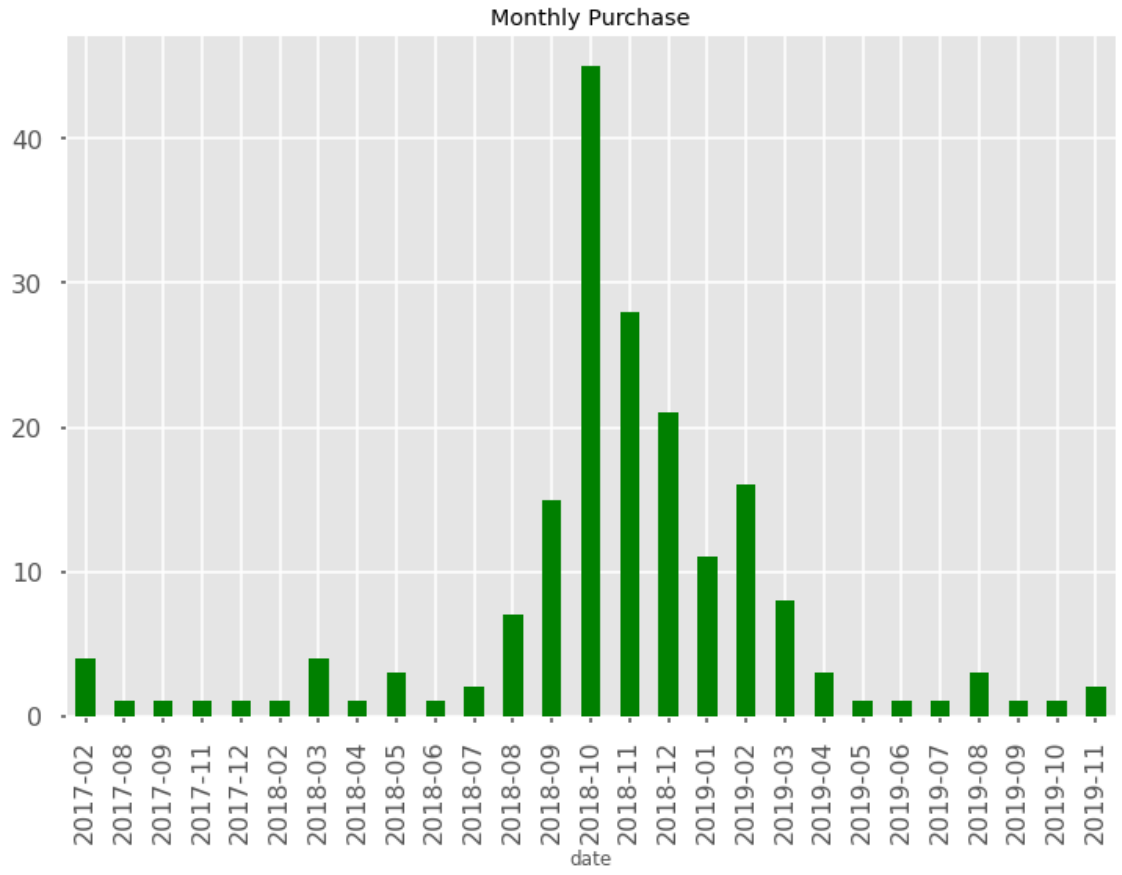
[ ]:

Time and Purchase

[157]:
```
purchase_df["date"] = pd.to_datetime(purchase_df.date, cache=True)

print("Start Of The Period: " , purchase_df.date.min())
print("End Of The Period  : " , purchase_df.date.max() )
print("Time Period        : " , purchase_df.date.max() - purchase_df.date.min())
```

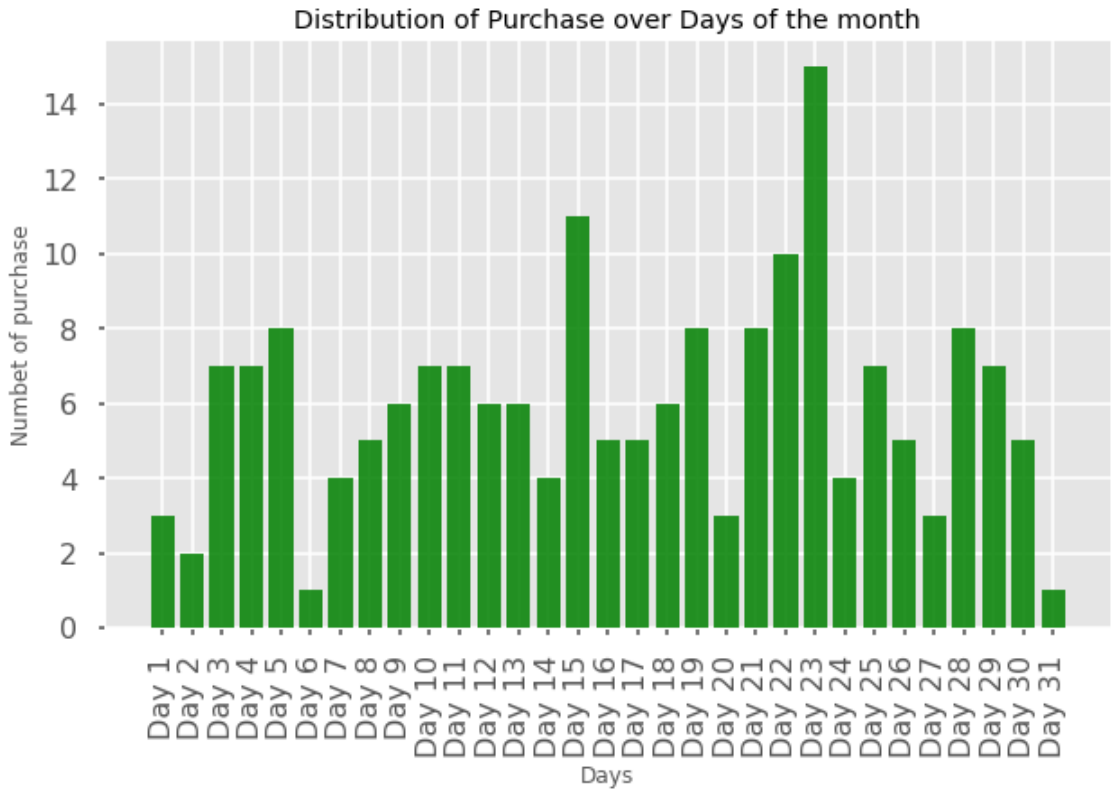
```
Start Of The Period:  2017-02-03 14:29:37
End Of The Period  :  2019-11-22 13:52:43
Time Period        :  1021 days 23:23:06
```

[158]:
```
purchase_month_count = pd.DataFrame(purchase_df.groupby(pd.
↪to_datetime(purchase_df["date"]).dt.to_period('M'))['product_uuid'].count()).
↪plot(kind='bar', legend = False,
↪                                                                    color =␣
↪'green',figsize=(12,8),
↪                                                                    title='Monthly␣
↪Purchase')
```



[159]:
```
# Daily distribution count
daily_count = purchase_df.groupby(purchase_df["date"].dt.day, as_index = False).
↪count()

fig= plt.figure(figsize=(10,6))
```

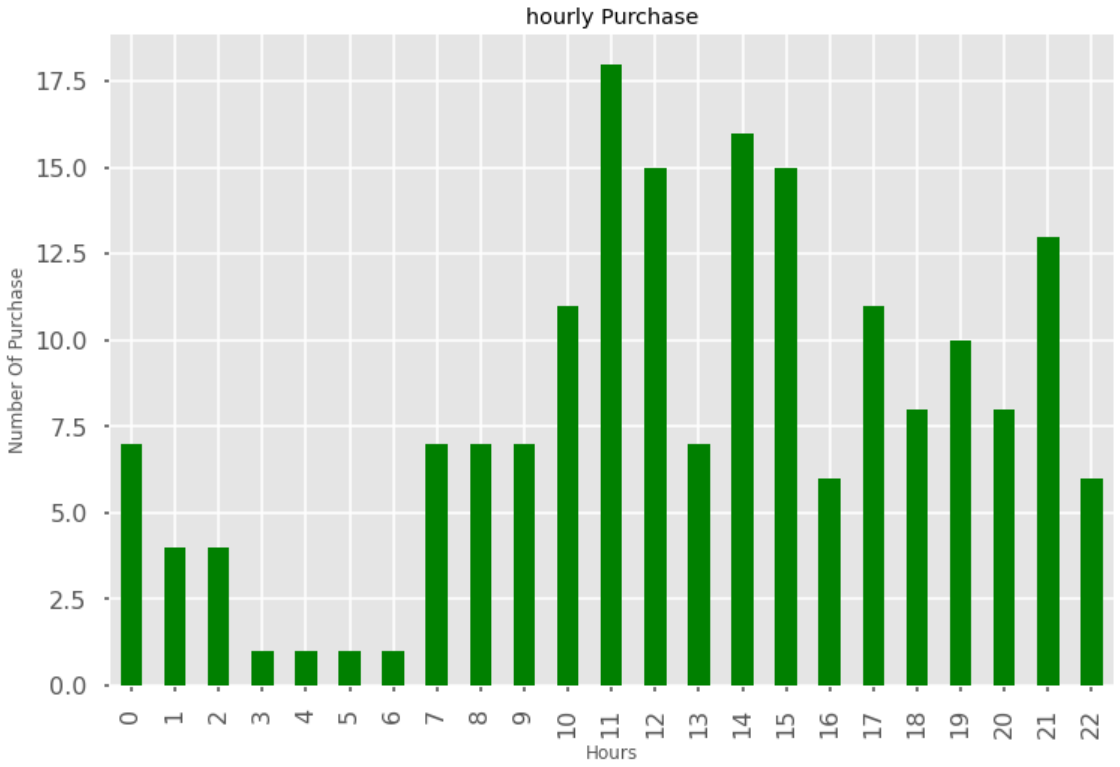
```
# Choose the names of the bars
bars = []
for i in range(1,32):
    bars.append('Day '+str(i))

y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, daily_count.date, color = 'green', alpha = 0.85)

# Create names on the x-axis
plt.xticks(y_pos, bars, rotation=90)
plt.yticks()

# Add title and axis names
plt.title('Distribution of Purchase over Days of the month')
plt.xlabel('Days', fontsize='12')
plt.ylabel('Numbet of purchase')


# Show graphic
plt.savefig('Images/Distribution of purchase over days.png',bbox_inches =␣
 ↪'tight')
plt.show()
```



[160]:
```
fig= plt.figure(figsize=(10,6))
hourly_count = purchase_df.groupby(purchase_df["date"].dt.hour, as_index =␣
↪False)['date'].count().plot(kind='bar', legend = False,
                                                                              ␣
↪                        color = 'green',figsize=(12,8),
                                                                              ␣
↪                        title='hourly Purchase')

plt.xlabel('Hours', fontsize='12')
plt.ylabel('Number Of Purchase')
```

[160]: Text(0, 0.5, 'Number Of Purchase')

<Figure size 720x432 with 0 Axes>



[ ]:

0.6 Join tables

1. join User table & Follow table
2. Join Product table & Purchase table
   - season cost average

user follow Dataframe

[161]:
```
users_df.head(2)
```

[161]:

	uuid	city_uuid	gender	date	month
0	06abe0e5-c408-4a33-9d81-5bfacc891f15	22972566-8739-40ca-bf19-a270d320dc83	1	2017-10-10 00:14:33	2017-10
1	33d7f40c-ab30-4517-881c-24d61acc62b8	22972566-8739-40ca-bf19-a270d320dc83	1	2017-10-10 00:14:33	2017-10

[162]:
```
users_df.rename(columns = {'uuid': 'userid', 'city_uuid' : 'city_userid',␣
↪'gender':'user_gender', 'date' : 'user_date_register' , }, inplace =True)
users_df.head(2)
```

[162]:
```
                                 userid                             city_userid  \
0  06abe0e5-c408-4a33-9d81-5bfacc891f15  22972566-8739-40ca-bf19-a270d320dc83
1  33d7f40c-ab30-4517-881c-24d61acc62b8  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10
```

[163]:
```
follows_df.head(2)
```

[163]:
```
                              user_uuid                            follow_uuid  \
0  7effce15-12ba-469c-b4b4-b529425488db  372fe527-470c-48a3-a95e-fe32cc4bcc15
1  f262a138-9585-4570-9d2d-6815c4d8b6a6  ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd

                  date
0  2018-10-05 18:47:56
1  2020-06-29 14:57:37
```

[164]:
```
follows_df.rename(columns = {'user_uuid': 'userid', 'follow_uuid' :␣
↪'follow_userid', 'date' : 'date_follow' }, inplace =True)
follows_df.head(2)
```

[164]:
```
                                 userid                          follow_userid  \
0  7effce15-12ba-469c-b4b4-b529425488db  372fe527-470c-48a3-a95e-fe32cc4bcc15
1  f262a138-9585-4570-9d2d-6815c4d8b6a6  ca28e52f-ad3f-4ad8-8722-4a0fa355cbcd

           date_follow
0  2018-10-05 18:47:56
1  2020-06-29 14:57:37
```

[165]:
```
user_follow_df = pd.merge(users_df, follows_df, on='userid')
user_follow_df.head()
```

[165]:
```
                                 userid                             city_userid  \
0  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
1  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
2  331739cb-6aa3-4689-b002-ff1ba9014532  22972566-8739-40ca-bf19-a270d320dc83
3  372fe527-470c-48a3-a95e-fe32cc4bcc15  22972566-8739-40ca-bf19-a270d320dc83
4  2ac6cb5f-c758-43c4-9d77-3b61831998a2  907bef2e-4846-44a8-bea1-c5e77c3a328d

   user_gender  user_date_register    month  \
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10
2            1 2017-10-10 00:14:33  2017-10
3            1 2017-10-10 00:14:33  2017-10
4            1 2017-10-10 00:14:33  2017-10
```

```
                          follow_userid          date_follow
0  8608a01b-09e9-487f-a1d4-15309222973a  2017-09-19 16:05:31
1  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-09-18 23:37:51
2  233255de-55b8-4a46-800f-172a58016eec  2018-07-04 20:01:25
3  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-04-02 01:30:04
4  1ed79a26-26f7-4ce9-a711-a4fe0ea5e285  2019-04-23 00:42:15
```

[166]:
```
user_follow_df.shape
```

[166]:
```
(5886, 7)
```

[167]:
```
user_follow_df.dropna()
```

[167]:
```
                                    userid  \
0     634edd5c-318e-40ec-aeb9-1e863836754d
1     634edd5c-318e-40ec-aeb9-1e863836754d
2     331739cb-6aa3-4689-b002-ff1ba9014532
3     372fe527-470c-48a3-a95e-fe32cc4bcc15
4     2ac6cb5f-c758-43c4-9d77-3b61831998a2
…                                      …
5881  a8ebeeaa-456d-4c52-87b6-af933bb0f996
5882  a8ebeeaa-456d-4c52-87b6-af933bb0f996
5883  824abc12-e8d1-478b-94bf-576ba68a6159
5884  f5602717-0ece-475a-ab77-c5c0e822ff53
5885  f5602717-0ece-475a-ab77-c5c0e822ff53

                             city_userid  user_gender  user_date_register  \
0     22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
1     22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
2     22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
3     22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
4     907bef2e-4846-44a8-bea1-c5e77c3a328d            1 2017-10-10 00:14:33
…                                      …          …                   …
5881  da6236de-bc72-4abb-97db-c8065d36c366            1 2018-10-31 18:26:10
5882  da6236de-bc72-4abb-97db-c8065d36c366            1 2018-10-31 18:26:10
5883  49233767-888b-43fc-8441-9698f45417f6            1 2018-10-31 19:18:43
5884  308873a5-249c-4bd3-b721-9e5aa54cd0ce            1 2018-10-31 19:43:47
5885  308873a5-249c-4bd3-b721-9e5aa54cd0ce            1 2018-10-31 19:43:47

        month                          follow_userid          date_follow
0     2017-10  8608a01b-09e9-487f-a1d4-15309222973a  2017-09-19 16:05:31
1     2017-10  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-09-18 23:37:51
2     2017-10  233255de-55b8-4a46-800f-172a58016eec  2018-07-04 20:01:25
3     2017-10  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-04-02 01:30:04
4     2017-10  1ed79a26-26f7-4ce9-a711-a4fe0ea5e285  2019-04-23 00:42:15
…       …                                      …                   …
5881  2018-10  d98c1170-5f2a-446d-aa26-5beef1f054cf  2018-11-07 10:32:06
```

```
5882  2018-10  17fc2fcd-55d4-4342-a553-302e56d6d816  2018-10-31 18:50:33
5883  2018-10  e5b4f882-7dca-4e63-a665-f412cb383e39  2018-11-24 18:22:32
5884  2018-10  114a3247-5226-460f-850f-3d53ad112ed1  2019-08-28 23:50:50
5885  2018-10  28eeb457-8e0c-467e-a471-f72b065fc285  2019-09-01 16:17:18

[5886 rows x 7 columns]
```

[168]:
```
# Chech for Null Value
missing_percentage = user_follow_df.isnull().sum() / user_follow_df.shape[0] *␣
↪100
missing_percentage
```

[168]:
```
userid               0.000
city_userid          0.000
user_gender          0.000
user_date_register   0.000
month                0.000
follow_userid        0.000
date_follow          0.000
dtype: float64
```

[169]:
```
user_follow_df.head(2)
```

[169]:
```
                                  userid                              city_userid  \
0  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
1  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month  \
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10

                         follow_userid          date_follow
0  8608a01b-09e9-487f-a1d4-15309222973a  2017-09-19 16:05:31
1  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-09-18 23:37:51
```

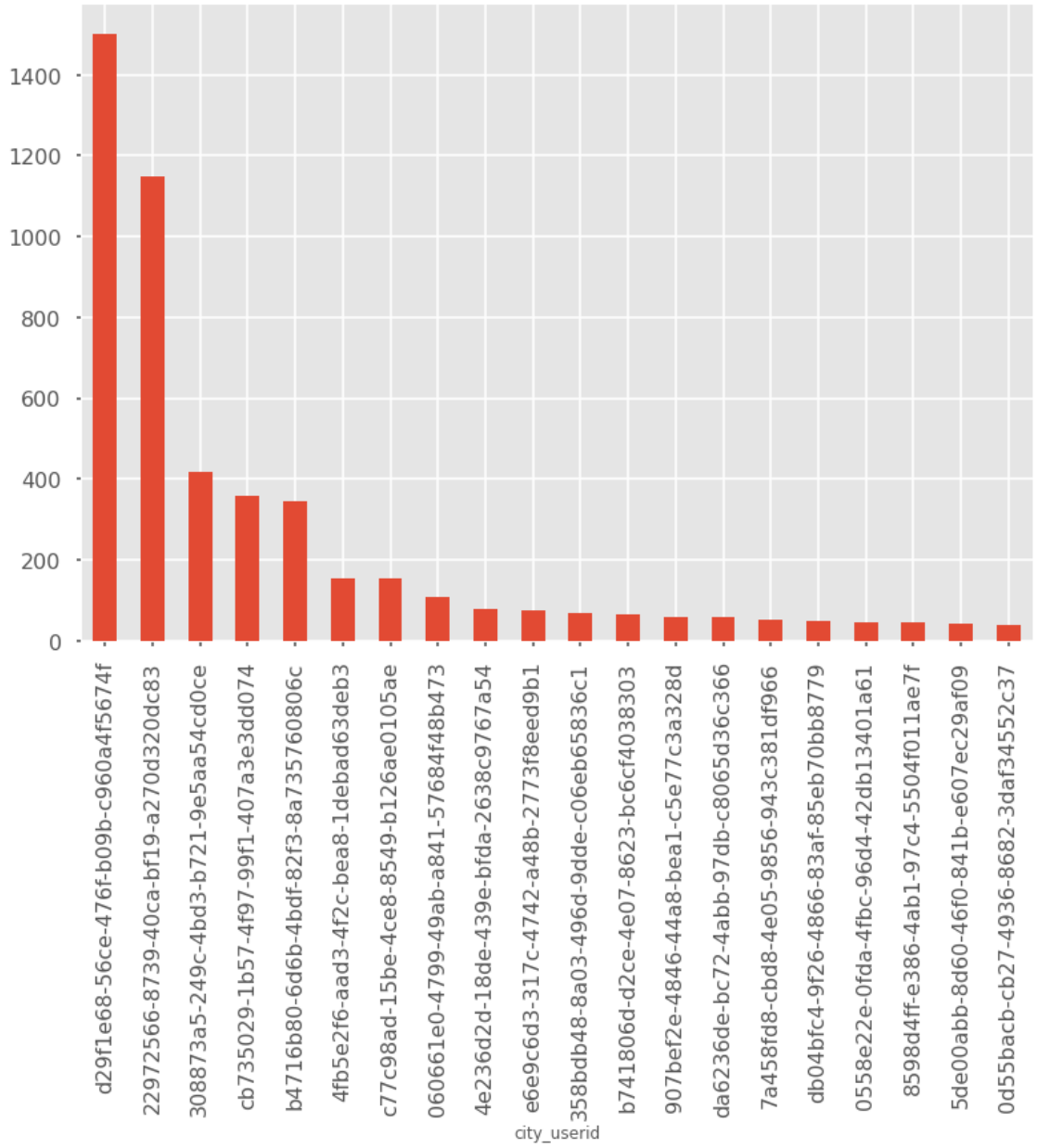
[170]:
```
# All the users in this dataset are male so we drop user_gender column in order␣
↪to make our data more readable
user_follow_df.drop(columns= 'user_gender', inplace =True )
```

more Socialable cities

[171]:
```
# user_follow_df.groupby(pd.to_datetime(products_df["date"]).dt.
↪to_period('M'))['uuid'].count().plot(kind='bar')
```

[172]:
```
user_follow_df.groupby('city_userid')['userid'].count().nlargest(20).
↪plot(kind='bar')
```

[172]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x284e1266100>
```



Time between registeration and first follow

[173]:
```
user_follow_df.head(2)
```

[173]:
```
                                 userid                              city_userid  \
0  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
1  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83

   user_date_register    month                              follow_userid  \
0 2017-10-10 00:14:33  2017-10  8608a01b-09e9-487f-a1d4-15309222973a
```

```
1 2017-10-10 00:14:33  2017-10  b629b255-1f46-4a92-b6d1-5d67b1ad8c54

           date_follow
0  2017-09-19 16:05:31
1  2017-09-18 23:37:51
```

[174]:
```
user_follow_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5886 entries, 0 to 5885
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   userid              5886 non-null   object
 1   city_userid         5886 non-null   object
 2   user_date_register  5886 non-null   datetime64[ns]
 3   month               5886 non-null   period[M]
 4   follow_userid       5886 non-null   object
 5   date_follow         5886 non-null   object
dtypes: datetime64[ns](1), object(4), period[M](1)
memory usage: 321.9+ KB
```

[175]:
```
user_follow_df['time_dif'] = pd.to_datetime(user_follow_df['date_follow']) - pd.
 ↪to_datetime(user_follow_df['user_date_register'])
user_follow_df.head()
```

[175]:
```
                                 userid                           city_userid  \
0  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
1  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83
2  331739cb-6aa3-4689-b002-ff1ba9014532  22972566-8739-40ca-bf19-a270d320dc83
3  372fe527-470c-48a3-a95e-fe32cc4bcc15  22972566-8739-40ca-bf19-a270d320dc83
4  2ac6cb5f-c758-43c4-9d77-3b61831998a2  907bef2e-4846-44a8-bea1-c5e77c3a328d

   user_date_register    month                          follow_userid  \
0 2017-10-10 00:14:33  2017-10  8608a01b-09e9-487f-a1d4-15309222973a
1 2017-10-10 00:14:33  2017-10  b629b255-1f46-4a92-b6d1-5d67b1ad8c54
2 2017-10-10 00:14:33  2017-10  233255de-55b8-4a46-800f-172a58016eec
3 2017-10-10 00:14:33  2017-10  b629b255-1f46-4a92-b6d1-5d67b1ad8c54
4 2017-10-10 00:14:33  2017-10  1ed79a26-26f7-4ce9-a711-a4fe0ea5e285

           date_follow            time_dif
0  2017-09-19 16:05:31  -21 days +15:50:58
1  2017-09-18 23:37:51  -22 days +23:23:18
2  2018-07-04 20:01:25   267 days 19:46:52
3  2017-04-02 01:30:04 -191 days +01:15:31
4  2019-04-23 00:42:15   560 days 00:27:42
```

[176]:
```
# It seems that our data has some problem with regards to time!
# let's double check to be sure!
```

[177]:
```
users_df[users_df['userid'] == '634edd5c-318e-40ec-aeb9-1e863836754d']
```

[177]:
```
                                  userid                             city_userid  \
4  634edd5c-318e-40ec-aeb9-1e863836754d  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month
4            1 2017-10-10 00:14:33  2017-10
```

[178]:
```
follows_df[follows_df['userid'] == '634edd5c-318e-40ec-aeb9-1e863836754d']
```

[178]:
```
                                  userid  \
324  634edd5c-318e-40ec-aeb9-1e863836754d
416  634edd5c-318e-40ec-aeb9-1e863836754d

                            follow_userid          date_follow
324  8608a01b-09e9-487f-a1d4-15309222973a  2017-09-19 16:05:31
416  b629b255-1f46-4a92-b6d1-5d67b1ad8c54  2017-09-18 23:37:51
```

The user with **634edd5c-318e-40ec-aeb9-1e863836754d** regiester on **2017-10-10 00:14:33** and follow two other user on **2017-09-19 16:05:3** and **2017-09-18 23:37:51**, which is impossible!

[ ]:
```

```

product_purchase Dataframe

[179]:
```
products_df.head(2)
```

[179]:
```
                                    uuid  \
10814  56d54b87-3a0f-4164-94d5-0217aa7bb8de
638    f7858eea-4370-410f-a77d-1260376e9f9f

                                  user_uuid  \
10814  a9502826-05f5-4551-84b7-4ab459715245
638    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0

                             category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd

                                  city_uuid  is_new  is_homemade  \
10814  1b1f27b2-40da-4fa7-bec4-697dca294770       1            0
638    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0

       original_price  price  comments_count  bookmarks_count  gender  \
10814      2147483647  10000               0                1       1
638        2147483647  25000               0                1       1
```

```
                       date    discount
10814 2019-12-19 02:42:28  2147473647
638   2019-12-19 02:42:28  2147458647
```

[180]:
```
products_df.rename(columns = {'uuid': 'productid', 'user_uuid' : 'userid',␣
↪'city_uuid': 'city_product_id', 'gender':'product_gender', 'date' :␣
↪'product_date_register' , }, inplace =True)
products_df.head(2)
```

[180]:
```
                                  productid  \
10814  56d54b87-3a0f-4164-94d5-0217aa7bb8de
638    f7858eea-4370-410f-a77d-1260376e9f9f

                                     userid  \
10814  a9502826-05f5-4551-84b7-4ab459715245
638    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0

                                category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd

                              city_product_id  is_new  is_homemade  \
10814  1b1f27b2-40da-4fa7-bec4-697dca294770       1            0
638    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0

       original_price  price  comments_count  bookmarks_count  product_gender  \
10814      2147483647  10000               0                1               1
638        2147483647  25000               0                1               1

      product_date_register    discount
10814   2019-12-19 02:42:28  2147473647
638     2019-12-19 02:42:28  2147458647
```

[181]:
```
purchase_df.head(2)
```

[181]:
```
                              buyer_uuid                              product_uuid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  99863785-21e9-4ebc-ae7a-8c558d8421b4
1  970ef8d9-c50e-4821-9687-145379af2b9a  e68a5832-7292-4a54-ad92-02f31a46494c

   count                 date
0      1  2018-02-15 17:38:52
1      1  2017-02-16 00:47:16
```

[182]:
```
purchase_df.rename(columns = {'product_uuid': 'productid', 'date' :␣
↪'date_purchase' }, inplace =True)
purchase_df.head(2)
```

[182]:
```
                              buyer_uuid                              productid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  99863785-21e9-4ebc-ae7a-8c558d8421b4
1  970ef8d9-c50e-4821-9687-145379af2b9a  e68a5832-7292-4a54-ad92-02f31a46494c

   count       date_purchase
0      1 2018-02-15 17:38:52
1      1 2017-02-16 00:47:16
```

[183]:
```
product_purchase_df = pd.merge(products_df, purchase_df, on='productid')
product_purchase_df.head(2)
```

[183]:
```
                               productid                              userid  \
0  52b9d9de-708b-4d79-9435-bb5175f50ee8  31c98b61-6193-4763-a01c-4eeca83972f8
1  58334645-bd86-481c-bd78-cdfc4b42a201  31c98b61-6193-4763-a01c-4eeca83972f8

                           category_uuid                        city_product_id  \
0  aa9c2861-90e8-4210-994b-332ecab092fd  7a458fd8-cbd8-4e05-9856-943c381df966
1  aa9c2861-90e8-4210-994b-332ecab092fd  7a458fd8-cbd8-4e05-9856-943c381df966

   is_new  is_homemade  original_price  price  comments_count  \
0       1            0         2800000  80000               0
1       0            0         2000000  50000               0

   bookmarks_count  product_gender product_date_register  discount  \
0               14               1   2019-12-19 02:42:28   2720000
1               25               1   2019-12-19 02:42:28   1950000

                              buyer_uuid  count       date_purchase
0  2678a5ce-7804-4b90-8631-bca481a3f1e1      1 2019-02-04 17:42:28
1  2678a5ce-7804-4b90-8631-bca481a3f1e1      1 2019-01-03 21:23:20
```

[184]:
```
product_purchase_df.shape
```

[184]:
```
(184, 16)
```

[185]:
```
product_purchase_df.dropna()
```

[185]:
```
                               productid  \
0    52b9d9de-708b-4d79-9435-bb5175f50ee8
1    58334645-bd86-481c-bd78-cdfc4b42a201
2    57a3a4ae-525e-490a-9109-16ade75d30d5
3    f7e69ba3-9a3f-4e7e-adaf-68b4c88395b5
4    2a4a1882-445d-4b73-b3d3-ba3c94e58b18
..                                    …
179  a5d8e285-8251-4348-9849-e99bd64bb4e9
180  0cd2b781-bd44-40cf-9422-a50dac21f4b5
181  35b5cb78-c42e-4542-aa84-594897150568
182  fa490d30-9fd5-4c52-afb7-fb4b5a1fe93f
```

```
183  acadb735-a654-4185-b22d-d12711c741f3

                                    userid  \
0    31c98b61-6193-4763-a01c-4eeca83972f8
1    31c98b61-6193-4763-a01c-4eeca83972f8
2    a29806a8-fa15-4c64-a75d-05de6798d53f
3    7ff60783-190e-4bcb-a78a-0ba20f469e3e
4    1aa167b8-0ebf-4d43-8853-33c3faf539d6
..                                    …
179  233255de-55b8-4a46-800f-172a58016eec
180  498439f1-52ed-4e3d-b07a-fe77061837b5
181  b629b255-1f46-4a92-b6d1-5d67b1ad8c54
182  b629b255-1f46-4a92-b6d1-5d67b1ad8c54
183  b629b255-1f46-4a92-b6d1-5d67b1ad8c54

                              category_uuid  \
0    aa9c2861-90e8-4210-994b-332ecab092fd
1    aa9c2861-90e8-4210-994b-332ecab092fd
2    aa9c2861-90e8-4210-994b-332ecab092fd
3    aa9c2861-90e8-4210-994b-332ecab092fd
4    aa9c2861-90e8-4210-994b-332ecab092fd
..                                    …
179  aa9c2861-90e8-4210-994b-332ecab092fd
180  aa9c2861-90e8-4210-994b-332ecab092fd
181  3be89fba-3211-44de-917d-2a968dabcf38
182  aa9c2861-90e8-4210-994b-332ecab092fd
183  aa9c2861-90e8-4210-994b-332ecab092fd

                           city_product_id  is_new  is_homemade  \
0    7a458fd8-cbd8-4e05-9856-943c381df966       1            0
1    7a458fd8-cbd8-4e05-9856-943c381df966       0            0
2    22972566-8739-40ca-bf19-a270d320dc83       0            0
3    d29f1e68-56ce-476f-b09b-c960a4f5674f       1            0
4    308873a5-249c-4bd3-b721-9e5aa54cd0ce       0            0
..                                    …     …           …
179  22972566-8739-40ca-bf19-a270d320dc83       0            0
180  22972566-8739-40ca-bf19-a270d320dc83       0            0
181  22972566-8739-40ca-bf19-a270d320dc83       1            0
182  22972566-8739-40ca-bf19-a270d320dc83       1            0
183  22972566-8739-40ca-bf19-a270d320dc83       1            0

     original_price   price  comments_count  bookmarks_count  product_gender  \
0           2800000   80000               0               14               1
1           2000000   50000               0               25               1
2            680000   50000               0                3               1
3            880000  350000               0               18               1
4            370000    5000               0                3               1
```

```
..                …      …                …                …                …
179               0   33000                0                9                1
180               0   35000                0                0                1
181               0   40000                0               16                1
182               0   55000                0                1                1
183               0   60000                0                1                1

    product_date_register  discount                            buyer_uuid  \
0     2019-12-19 02:42:28   2720000  2678a5ce-7804-4b90-8631-bca481a3f1e1
1     2019-12-19 02:42:28   1950000  2678a5ce-7804-4b90-8631-bca481a3f1e1
2     2019-12-19 02:42:28    630000  31c98b61-6193-4763-a01c-4eeca83972f8
3     2019-12-19 02:42:28    530000  8c5a7228-e794-4b14-b9cb-91e2156c9c2c
4     2019-12-19 02:42:28    365000  20afdbd8-d0c2-40ea-bf0c-9302a0c9e236
..                    …       …                                     …
179   2019-12-19 02:42:28    -33000  3b08c894-4c48-40de-a48b-00d68c4d1650
180   2019-12-19 02:42:28    -35000  233255de-55b8-4a46-800f-172a58016eec
181   2019-12-19 02:42:28    -40000  4ee6b228-e993-4380-89cf-ad6776fe3af6
182   2019-12-19 02:42:28    -55000  770728cb-3f6c-4230-af5c-19cb9684be16
183   2019-12-19 02:42:28    -60000  770728cb-3f6c-4230-af5c-19cb9684be16

     count       date_purchase
0        1 2019-02-04 17:42:28
1        1 2019-01-03 21:23:20
2        1 2018-10-10 11:17:55
3        1 2018-10-26 23:13:27
4        1 2018-10-24 21:57:27
..     …                   …
179      1 2018-09-10 19:59:01
180      1 2018-03-12 20:26:13
181      1 2018-12-19 10:52:36
182      1 2017-02-25 12:19:12
183      1 2017-02-25 12:15:31

[184 rows x 16 columns]
```

[186]:
```
# Chech for Null Value
missing_percentage = product_purchase_df.isnull().sum() / product_purchase_df.
 ↪shape[0] * 100
missing_percentage
```

[186]:
```
productid              0.000
userid                 0.000
category_uuid          0.000
city_product_id        0.000
is_new                 0.000
is_homemade            0.000
original_price         0.000
```

```
price                   0.000
comments_count          0.000
bookmarks_count         0.000
product_gender          0.000
product_date_register   0.000
discount                0.000
buyer_uuid              0.000
count                   0.000
date_purchase           0.000
dtype: float64
```

[187]:
```
product_purchase_df.head(2)
```

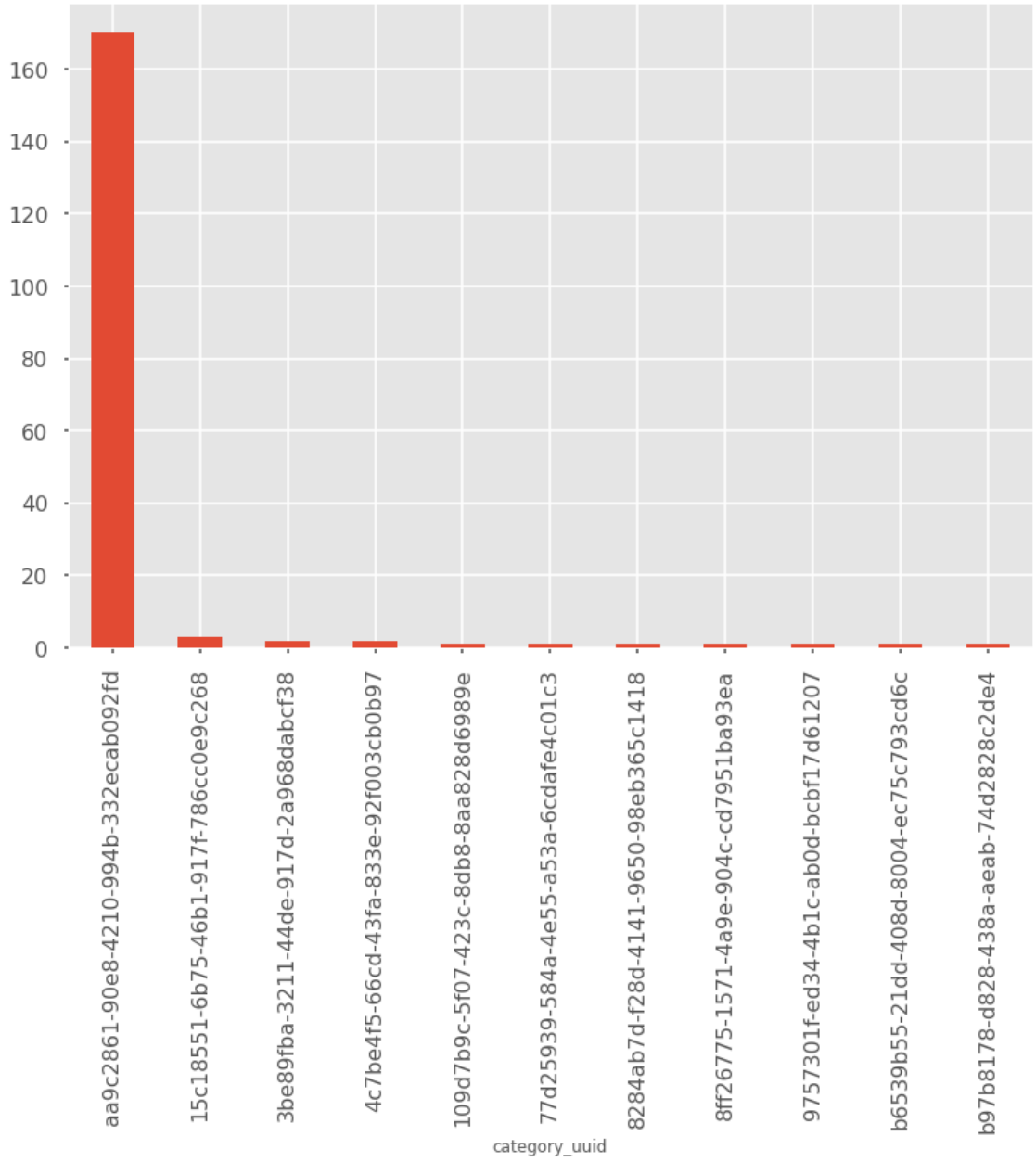
[187]:

	productid	userid	category_uuid	city_product_id	is_new	is_homemade	original_price	price	comments_count	bookmarks_count	product_gender	product_date_register	discount	buyer_uuid	count	date_purchase
0	52b9d9de-708b-4d79-9435-bb5175f50ee8	31c98b61-6193-4763-a01c-4eeca83972f8	aa9c2861-90e8-4210-994b-332ecab092fd	7a458fd8-cbd8-4e05-9856-943c381df966	1	0	2800000	80000	0	14	1	2019-12-19 02:42:28	2720000	2678a5ce-7804-4b90-8631-bca481a3f1e1	1	2019-02-04 17:42:28
1	58334645-bd86-481c-bd78-cdfc4b42a201	31c98b61-6193-4763-a01c-4eeca83972f8	aa9c2861-90e8-4210-994b-332ecab092fd	7a458fd8-cbd8-4e05-9856-943c381df966	0	0	2000000	50000	0	25	1	2019-12-19 02:42:28	1950000	2678a5ce-7804-4b90-8631-bca481a3f1e1	1	2019-01-03 21:23:20

category with most purchase

[188]:
```
product_purchase_df.groupby('category_uuid')['category_uuid'].count().
↪nlargest(20).plot(kind='bar')
```

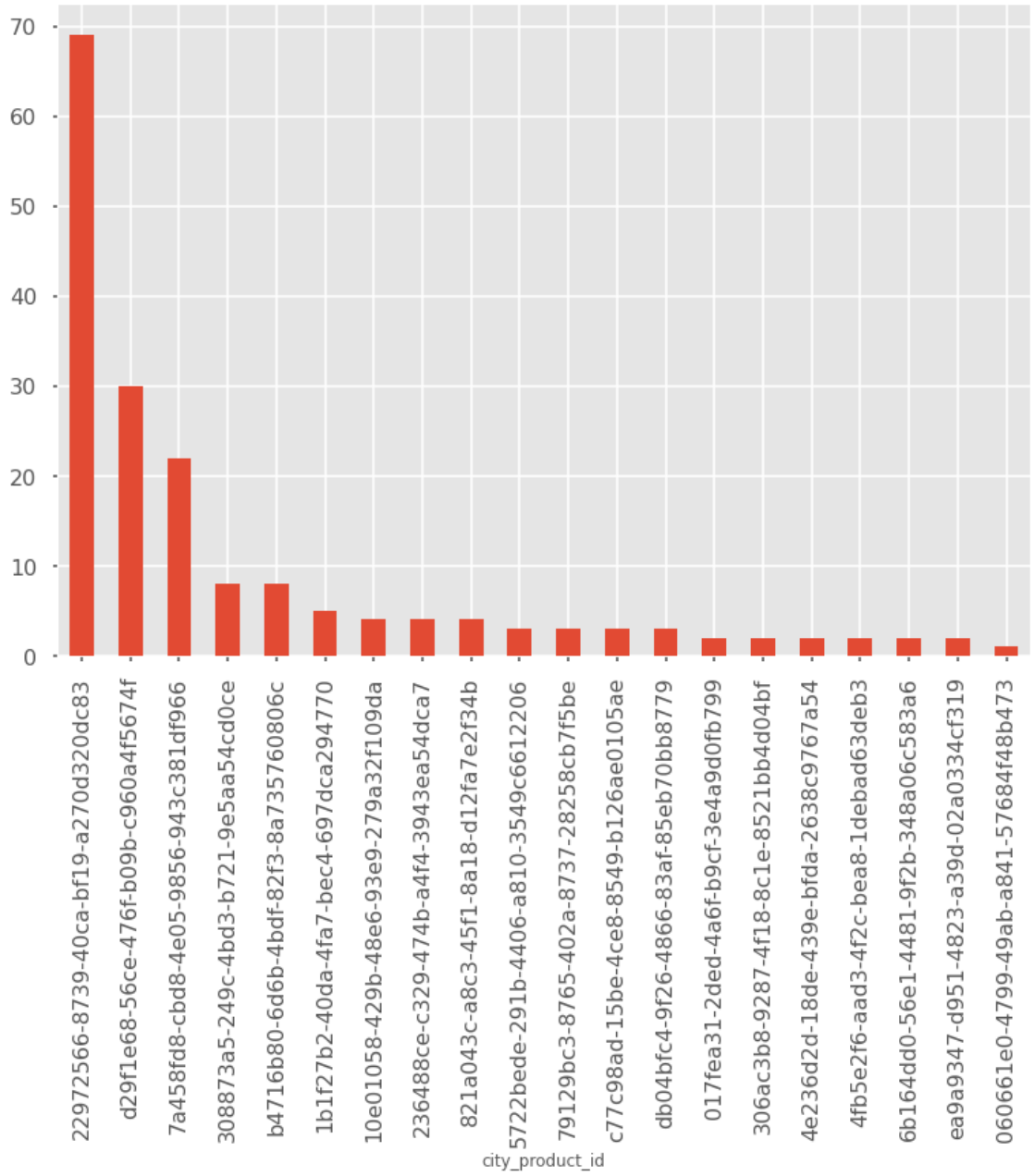
[188]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x284e0ce36d0>
```



cities with most successful sales!

[189]:
```
product_purchase_df.groupby('city_product_id')['city_product_id'].count().
 ↪nlargest(20).plot(kind='bar')
```

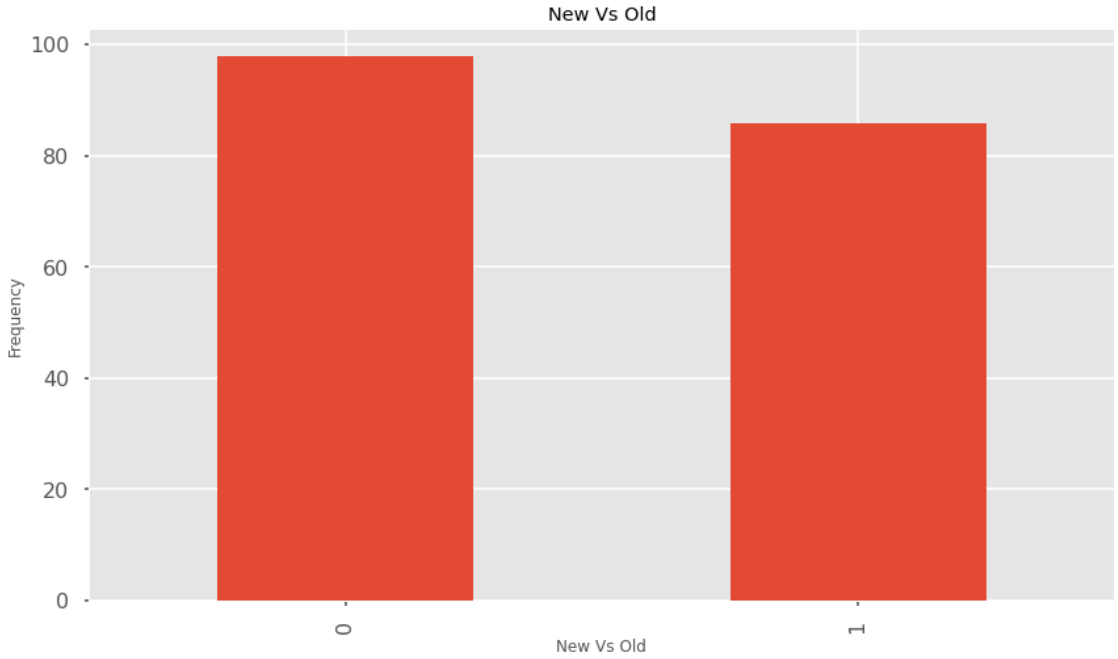
[189]: <matplotlib.axes._subplots.AxesSubplot at 0x284df6e2070>



Which one sales better? New or Old?

[190]:
```
ax = product_purchase_df['is_new'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    title="New Vs Old")
ax.set_xlabel("New Vs Old")
ax.set_ylabel("Frequency")
```

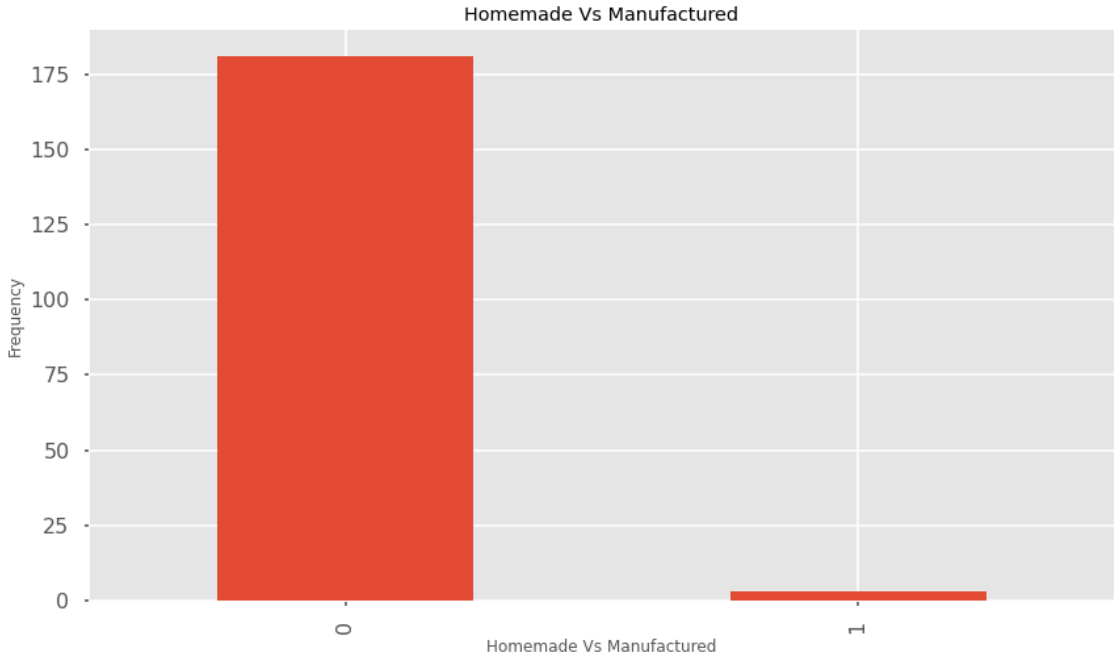
[190]: Text(0, 0.5, 'Frequency')



Which one sales better? homemade or Manufactured?

[191]:
```
ax = product_purchase_df['is_homemade'].value_counts().plot(kind='bar',
                                    figsize=(14,8),
                                    title="Homemade Vs Manufactured")
ax.set_xlabel("Homemade Vs Manufactured")
ax.set_ylabel("Frequency")
```

[191]:
```
Text(0, 0.5, 'Frequency')
```



How long does it take to sale a product on average?

[192]:
```
product_purchase_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 0 to 183
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   productid             184 non-null    object        
 1   userid                184 non-null    object        
 2   category_uuid         184 non-null    object        
 3   city_product_id       184 non-null    object        
 4   is_new                184 non-null    int64         
 5   is_homemade           184 non-null    int64         
 6   original_price        184 non-null    int64         
 7   price                 184 non-null    int64         
 8   comments_count        184 non-null    int64         
 9   bookmarks_count       184 non-null    int64         
 10  product_gender        184 non-null    int64         
 11  product_date_register 184 non-null    datetime64[ns]
 12  discount              184 non-null    int64         
 13  buyer_uuid            184 non-null    object        
 14  count                 184 non-null    int64         
 15  date_purchase         184 non-null    datetime64[ns]
dtypes: datetime64[ns](2), int64(9), object(5)
memory usage: 24.4+ KB
```

[193]:
```
product_purchase_df['time_dif'] = pd.
 ↪to_datetime(product_purchase_df['date_purchase']) - pd.
 ↪to_datetime(product_purchase_df['product_date_register'])
product_purchase_df.head()
```

[193]:

	productid	userid	\
0	52b9d9de-708b-4d79-9435-bb5175f50ee8	31c98b61-6193-4763-a01c-4eeca83972f8	
1	58334645-bd86-481c-bd78-cdfc4b42a201	31c98b61-6193-4763-a01c-4eeca83972f8	
2	57a3a4ae-525e-490a-9109-16ade75d30d5	a29806a8-fa15-4c64-a75d-05de6798d53f	
3	f7e69ba3-9a3f-4e7e-adaf-68b4c88395b5	7ff60783-190e-4bcb-a78a-0ba20f469e3e	
4	2a4a1882-445d-4b73-b3d3-ba3c94e58b18	1aa167b8-0ebf-4d43-8853-33c3faf539d6	

	category_uuid	city_product_id	\
0	aa9c2861-90e8-4210-994b-332ecab092fd	7a458fd8-cbd8-4e05-9856-943c381df966	
1	aa9c2861-90e8-4210-994b-332ecab092fd	7a458fd8-cbd8-4e05-9856-943c381df966	
2	aa9c2861-90e8-4210-994b-332ecab092fd	22972566-8739-40ca-bf19-a270d320dc83	
3	aa9c2861-90e8-4210-994b-332ecab092fd	d29f1e68-56ce-476f-b09b-c960a4f5674f	
4	aa9c2861-90e8-4210-994b-332ecab092fd	308873a5-249c-4bd3-b721-9e5aa54cd0ce	

	is_new	is_homemade	original_price	price	comments_count	\
0	1	0	2800000	80000	0	
1	0	0	2000000	50000	0	
2	0	0	680000	50000	0	
3	1	0	880000	350000	0	
4	0	0	370000	5000	0	

	bookmarks_count	product_gender	product_date_register	discount	\
0	14	1	2019-12-19 02:42:28	2720000	
1	25	1	2019-12-19 02:42:28	1950000	
2	3	1	2019-12-19 02:42:28	630000	
3	18	1	2019-12-19 02:42:28	530000	
4	3	1	2019-12-19 02:42:28	365000	

	buyer_uuid	count	date_purchase	\
0	2678a5ce-7804-4b90-8631-bca481a3f1e1	1	2019-02-04 17:42:28	
1	2678a5ce-7804-4b90-8631-bca481a3f1e1	1	2019-01-03 21:23:20	
2	31c98b61-6193-4763-a01c-4eeca83972f8	1	2018-10-10 11:17:55	
3	8c5a7228-e794-4b14-b9cb-91e2156c9c2c	1	2018-10-26 23:13:27	
4	20afdbd8-d0c2-40ea-bf0c-9302a0c9e236	1	2018-10-24 21:57:27	

	time_dif
0	-318 days +15:00:00
1	-350 days +18:40:52
2	-435 days +08:35:27
3	-419 days +20:30:59
4	-421 days +19:14:59

It seems that we have the same problem here with this table too! let's double check to make sure!

[194]:
```
products_df.head(2)
```

[194]:
```
                                  productid  \
10814  56d54b87-3a0f-4164-94d5-0217aa7bb8de
638    f7858eea-4370-410f-a77d-1260376e9f9f

                                     userid  \
10814  a9502826-05f5-4551-84b7-4ab459715245
638    85ba8dcc-eab6-47fb-a112-0f4fe72ef5a0

                               category_uuid  \
10814  aa9c2861-90e8-4210-994b-332ecab092fd
638    aa9c2861-90e8-4210-994b-332ecab092fd

                            city_product_id  is_new  is_homemade  \
10814  1b1f27b2-40da-4fa7-bec4-697dca294770       1            0
638    d29f1e68-56ce-476f-b09b-c960a4f5674f       0            0

       original_price  price  comments_count  bookmarks_count  product_gender  \
10814      2147483647  10000               0                1               1
638        2147483647  25000               0                1               1

      product_date_register    discount
10814   2019-12-19 02:42:28  2147473647
638     2019-12-19 02:42:28  2147458647
```

[195]:
```
products_df[products_df['productid'] == '0cd2b781-bd44-40cf-9422-a50dac21f4b5']
```

[195]:
```
                               productid  \
44  0cd2b781-bd44-40cf-9422-a50dac21f4b5

                                  userid  \
44  498439f1-52ed-4e3d-b07a-fe77061837b5

                            category_uuid  \
44  aa9c2861-90e8-4210-994b-332ecab092fd

                         city_product_id  is_new  is_homemade  original_price  \
44  22972566-8739-40ca-bf19-a270d320dc83       0            0               0

    price  comments_count  bookmarks_count  product_gender  \
44  35000               0                0               1

   product_date_register  discount
44   2019-12-19 02:42:28    -35000
```

[196]:
```
purchase_df[purchase_df['productid'] == '0cd2b781-bd44-40cf-9422-a50dac21f4b5']
```

[196]:
```
                                buyer_uuid                                 productid  \
4  233255de-55b8-4a46-800f-172a58016eec  0cd2b781-bd44-40cf-9422-a50dac21f4b5

   count       date_purchase
4      1 2018-03-12 20:26:13
```

The product with **0cd2b781-bd44-40cf-9422-a50dac21f4b5** regiester on **2019-12-19 02:42:28** and sold on **2018-03-12 20:26:13**, which is impossible!

[ ]:
```
```

user_purchase Dataframe

[197]:
```
users_df.head(2)
```

[197]:
```
                                 userid                                city_userid  \
0  06abe0e5-c408-4a33-9d81-5bfacc891f15  22972566-8739-40ca-bf19-a270d320dc83
1  33d7f40c-ab30-4517-881c-24d61acc62b8  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10
```

[198]:
```
purchase_df.head(2)
```

[198]:
```
                                buyer_uuid                                 productid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  99863785-21e9-4ebc-ae7a-8c558d8421b4
1  970ef8d9-c50e-4821-9687-145379af2b9a  e68a5832-7292-4a54-ad92-02f31a46494c

   count       date_purchase
0      1 2018-02-15 17:38:52
1      1 2017-02-16 00:47:16
```

[199]:
```
purchase_df.rename(columns = {'buyer_uuid': 'userid'}, inplace = True)
purchase_df.head(2)
```

[199]:
```
                                 userid                                 productid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  99863785-21e9-4ebc-ae7a-8c558d8421b4
1  970ef8d9-c50e-4821-9687-145379af2b9a  e68a5832-7292-4a54-ad92-02f31a46494c

   count       date_purchase
0      1 2018-02-15 17:38:52
1      1 2017-02-16 00:47:16
```

[200]:
```
user_purchase_df = pd.merge(users_df, purchase_df, on='userid')
user_purchase_df.head()
```

[200]:
```
                                 userid                                city_userid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  22972566-8739-40ca-bf19-a270d320dc83
```

```
1  970ef8d9-c50e-4821-9687-145379af2b9a  22972566-8739-40ca-bf19-a270d320dc83
2  970ef8d9-c50e-4821-9687-145379af2b9a  22972566-8739-40ca-bf19-a270d320dc83
3  0bc6742f-d58a-4539-b85b-62bc3463033b  22972566-8739-40ca-bf19-a270d320dc83
4  233255de-55b8-4a46-800f-172a58016eec  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month  \
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10
2            1 2017-10-10 00:14:33  2017-10
3            1 2017-10-10 00:14:33  2017-10
4            1 2017-10-10 00:14:33  2017-10

                              productid  count       date_purchase
0  99863785-21e9-4ebc-ae7a-8c558d8421b4      1 2018-02-15 17:38:52
1  e68a5832-7292-4a54-ad92-02f31a46494c      1 2017-02-16 00:47:16
2  ecd1b68f-9db4-49f5-bac5-5679d9f9d784      1 2018-03-01 15:10:23
3  af59299c-2d08-4993-9be0-cfd7e641ae0d      1 2017-12-17 12:49:20
4  0cd2b781-bd44-40cf-9422-a50dac21f4b5      1 2018-03-12 20:26:13
```

[201]:
```
user_purchase_df.shape
```

[201]:
```
(184, 8)
```

[202]:
```
user_purchase_df.dropna()
```

[202]:
```
                                  userid  \
0    372fe527-470c-48a3-a95e-fe32cc4bcc15
1    970ef8d9-c50e-4821-9687-145379af2b9a
2    970ef8d9-c50e-4821-9687-145379af2b9a
3    0bc6742f-d58a-4539-b85b-62bc3463033b
4    233255de-55b8-4a46-800f-172a58016eec
..                                    …
179  af9048ee-3109-483c-b710-ed0fd8dbe41c
180  8abf1063-e8d4-4630-83e0-b139cfebb37f
181  e5789a70-db24-483e-8d3f-85002011d954
182  61c3b4b1-e33a-47d6-b62a-3dd2c78e4497
183  a85d3645-000b-4a62-a8e3-4e5b7574c934

                              city_userid  user_gender  user_date_register  \
0    22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
1    22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
2    22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
3    22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
4    22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33
..                                    …          …                   …
179  cb735029-1b57-4f97-99f1-407a3e3dd074            1 2018-10-25 12:18:46
180  060661e0-4799-49ab-a841-57684f48b473            1 2018-10-25 18:41:59
```

```
181  a200397b-66c5-4161-8280-cc2bcafedfed            1 2018-10-28 09:48:01
182  7a458fd8-cbd8-4e05-9856-943c381df966            1 2018-10-28 18:13:39
183  cb735029-1b57-4f97-99f1-407a3e3dd074            1 2018-10-30 15:14:20

       month                                productid  count       date_purchase
0    2017-10  99863785-21e9-4ebc-ae7a-8c558d8421b4      1 2018-02-15 17:38:52
1    2017-10  e68a5832-7292-4a54-ad92-02f31a46494c      1 2017-02-16 00:47:16
2    2017-10  ecd1b68f-9db4-49f5-bac5-5679d9f9d784      1 2018-03-01 15:10:23
3    2017-10  af59299c-2d08-4993-9be0-cfd7e641ae0d      1 2017-12-17 12:49:20
4    2017-10  0cd2b781-bd44-40cf-9422-a50dac21f4b5      1 2018-03-12 20:26:13
..       …                                    …    …                   …
179  2018-10  d6d66059-e6a5-4309-a864-e0076d0d934e      1 2018-10-29 12:17:37
180  2018-10  7b9fb23f-8598-4752-8c0c-352e568769a5      1 2018-11-23 14:01:24
181  2018-10  290cf9b3-a7ad-4820-85c9-7493e32256e2      1 2018-11-14 22:49:40
182  2018-10  8cd5bef7-9c1f-4d47-ad90-b350356f0883      1 2018-12-07 20:06:18
183  2018-10  f24cd7cc-3154-41e5-92ec-76eebea2afdb      1 2019-09-11 22:20:45

[184 rows x 8 columns]
```

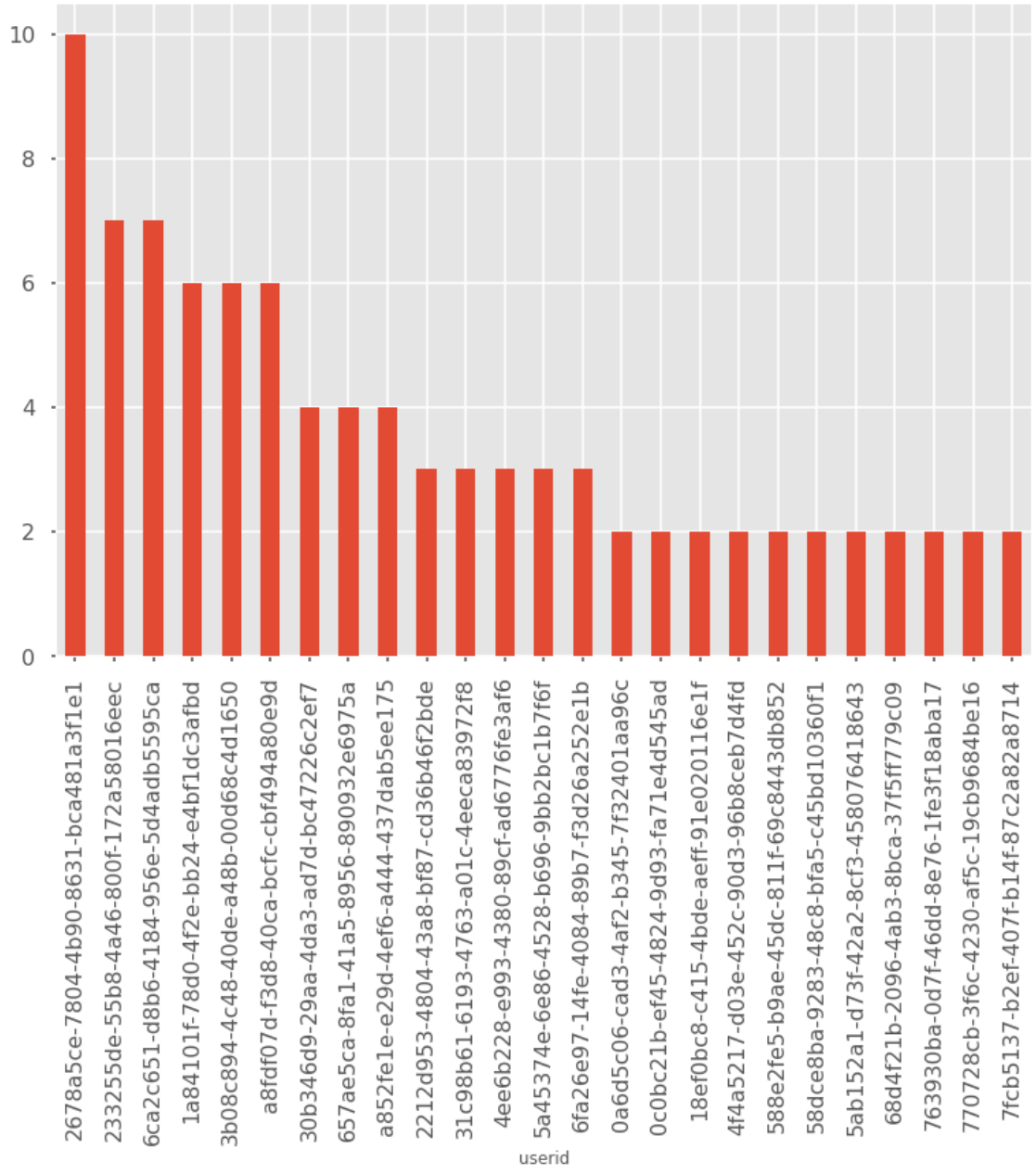
[203]:
```
# Chech for Null Value
missing_percentage = user_purchase_df.isnull().sum() / user_purchase_df.
 ↪shape[0] * 100
missing_percentage
```

[203]:
```
userid               0.000
city_userid          0.000
user_gender          0.000
user_date_register   0.000
month                0.000
productid            0.000
count                0.000
date_purchase        0.000
dtype: float64
```

Users with most purchases

[204]:
```
user_purchase_df.groupby('userid')['userid'].count().nlargest(25).
 ↪plot(kind='bar')
```

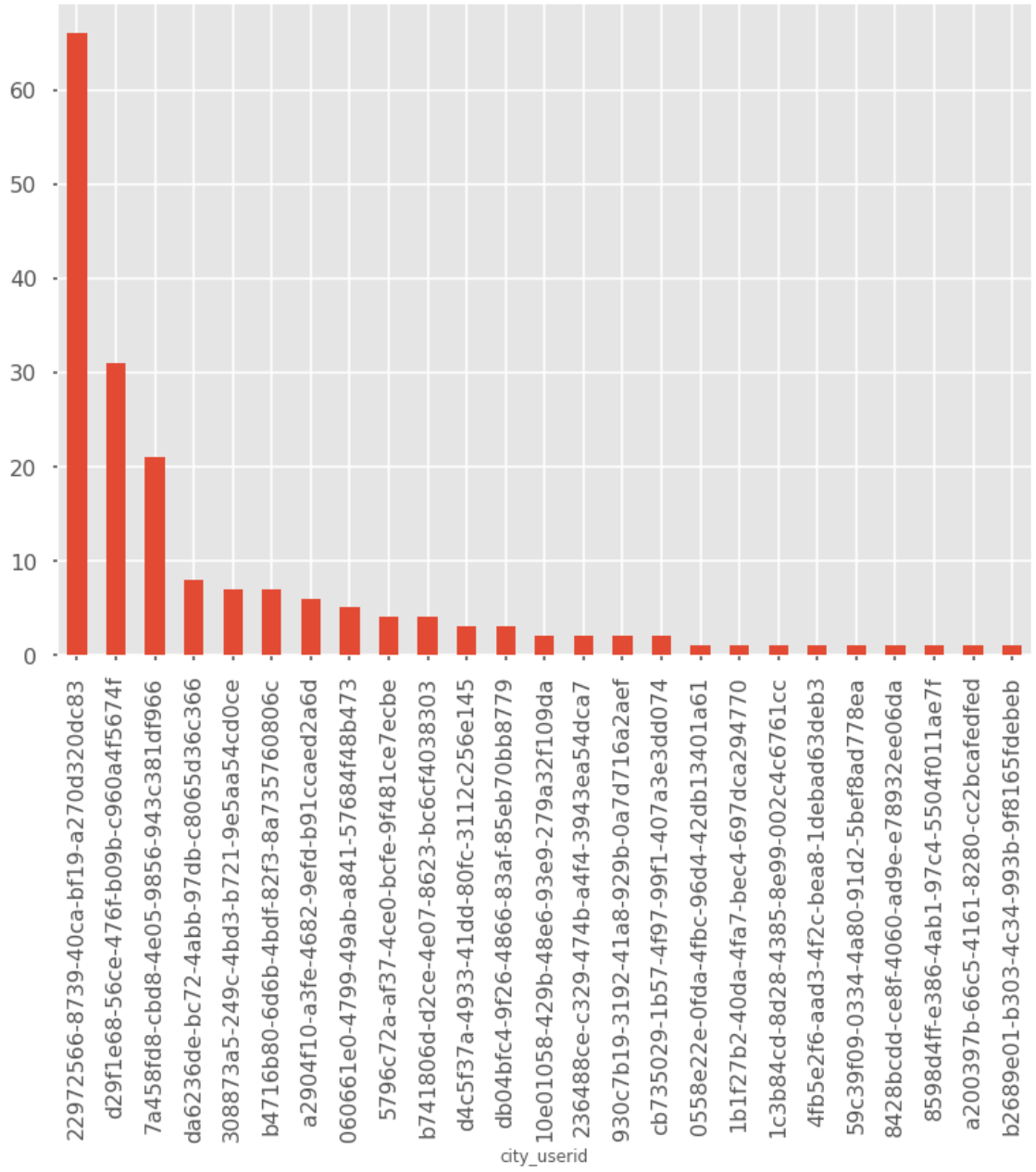
[204]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x284d4753580>
```



cities with most purchases

[205]:
```
user_purchase_df.groupby('city_userid')['city_userid'].count().nlargest(25).
↪plot(kind='bar')
```

[205]: <matplotlib.axes._subplots.AxesSubplot at 0x284d4708a30>



Time between registeration and first purchase

[206]:
```
user_purchase_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184 entries, 0 to 183
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   userid              184 non-null    object
```

```
 1   city_userid          184 non-null    object
 2   user_gender          184 non-null    int64
 3   user_date_register   184 non-null    datetime64[ns]
 4   month                184 non-null    period[M]
 5   productid            184 non-null    object
 6   count                184 non-null    int64
 7   date_purchase        184 non-null    datetime64[ns]
dtypes: datetime64[ns](2), int64(2), object(3), period[M](1)
memory usage: 12.9+ KB
```

[207]:
```
user_purchase_df['time_dif'] = pd.
 ↪to_datetime(user_purchase_df['date_purchase']) - pd.
 ↪to_datetime(user_purchase_df['user_date_register'])
user_purchase_df.head()
```

[207]:
```
                                 userid                            city_userid  \
0  372fe527-470c-48a3-a95e-fe32cc4bcc15  22972566-8739-40ca-bf19-a270d320dc83
1  970ef8d9-c50e-4821-9687-145379af2b9a  22972566-8739-40ca-bf19-a270d320dc83
2  970ef8d9-c50e-4821-9687-145379af2b9a  22972566-8739-40ca-bf19-a270d320dc83
3  0bc6742f-d58a-4539-b85b-62bc3463033b  22972566-8739-40ca-bf19-a270d320dc83
4  233255de-55b8-4a46-800f-172a58016eec  22972566-8739-40ca-bf19-a270d320dc83

   user_gender  user_date_register    month  \
0            1 2017-10-10 00:14:33  2017-10
1            1 2017-10-10 00:14:33  2017-10
2            1 2017-10-10 00:14:33  2017-10
3            1 2017-10-10 00:14:33  2017-10
4            1 2017-10-10 00:14:33  2017-10

                             productid  count       date_purchase  \
0  99863785-21e9-4ebc-ae7a-8c558d8421b4      1 2018-02-15 17:38:52
1  e68a5832-7292-4a54-ad92-02f31a46494c      1 2017-02-16 00:47:16
2  ecd1b68f-9db4-49f5-bac5-5679d9f9d784      1 2018-03-01 15:10:23
3  af59299c-2d08-4993-9be0-cfd7e641ae0d      1 2017-12-17 12:49:20
4  0cd2b781-bd44-40cf-9422-a50dac21f4b5      1 2018-03-12 20:26:13

             time_dif
0   128 days 17:24:19
1 -236 days +00:32:43
2   142 days 14:55:50
3    68 days 12:34:47
4   153 days 20:11:40
```

And again the same problem!

[208]:
```
users_df[users_df['userid'] == '970ef8d9-c50e-4821-9687-145379af2b9a']
```

[208]:
```
                                    userid  \
41  970ef8d9-c50e-4821-9687-145379af2b9a

                                city_userid  user_gender  user_date_register  \
41  22972566-8739-40ca-bf19-a270d320dc83            1 2017-10-10 00:14:33

      month
41  2017-10
```

[209]:
```
purchase_df[purchase_df['userid'] == '970ef8d9-c50e-4821-9687-145379af2b9a']
```

[209]:
```
                                  userid                                 productid  \
1  970ef8d9-c50e-4821-9687-145379af2b9a  e68a5832-7292-4a54-ad92-02f31a46494c
2  970ef8d9-c50e-4821-9687-145379af2b9a  ecd1b68f-9db4-49f5-bac5-5679d9f9d784

   count       date_purchase
1      1 2017-02-16 00:47:16
2      1 2018-03-01 15:10:23
```

User **970ef8d9-c50e-4821-9687-145379af2b9a** make a purchase twice and one of which is **-236 days** before he/she register to the website!

[ ]:

[ ]: