Steam Online Game

Imaginary Client

We have been approached by a company hoping to understand online game market via Steam. We will provide the findings to inform decisions about #1 game/genre preference per country #2 game addictive variable based on time factor #3 game popularity

Business Value: Why Online Game

The Online Game model provides social interaction, popularity and addictivity with time factor. These are the key drivers for value generation.

Segmenting customers and identifying patterns / trends are valuable to assist business decision.

Project Goal

The motivation of this project is to retrieve, process and analyse data via Steam Get API. This is to gain insights of what makes certain games popular in terms of #1 game/genre preference per country #2 game addictive variable based on time factor #3 game popularity

00

Project Flow

1 Data Collection & Data Cleaning

- [28 Jan] Day 1…
  - -Define Project Title
  - -Source Code - Steam API
  - -Write Code to generate Steam API
- [29 Jan] Day 2…
  - -Source Code - Steam API
  - -Realise there's too many API returning empty lists
  - -Realise there's too many Steam ID returning empty data
- [30 Jan] Day 2+
  - -Continue to generate Steam API to match AppID, returning data
  - -Key got blocked by Steam website. Source more keys from friends
  - -After getting 4 new keys, we generate codes with 4 separate computers, running full day
- [31 Jan] Day 2+
  - -Continue to generate Steam API to match AppID, returning data
- [01 Feb] Day 3…
  - -Continue to generate Steam API to match AppID, returning data

2 Visualisation

- [01 Feb] Day 3…
  - Generate Dataframe

3 Presentation

- [02 Feb] Day 4…

01

Data Collection & Data Cleaning

STEAM®

Get API Approach

“In 2003, digital storefront Steam is launched. Steam is a digital store for purchasing, downloading and playing video game, similar to google play store and apple app store”

32,000

Total Game on Steam Today

26 mil users

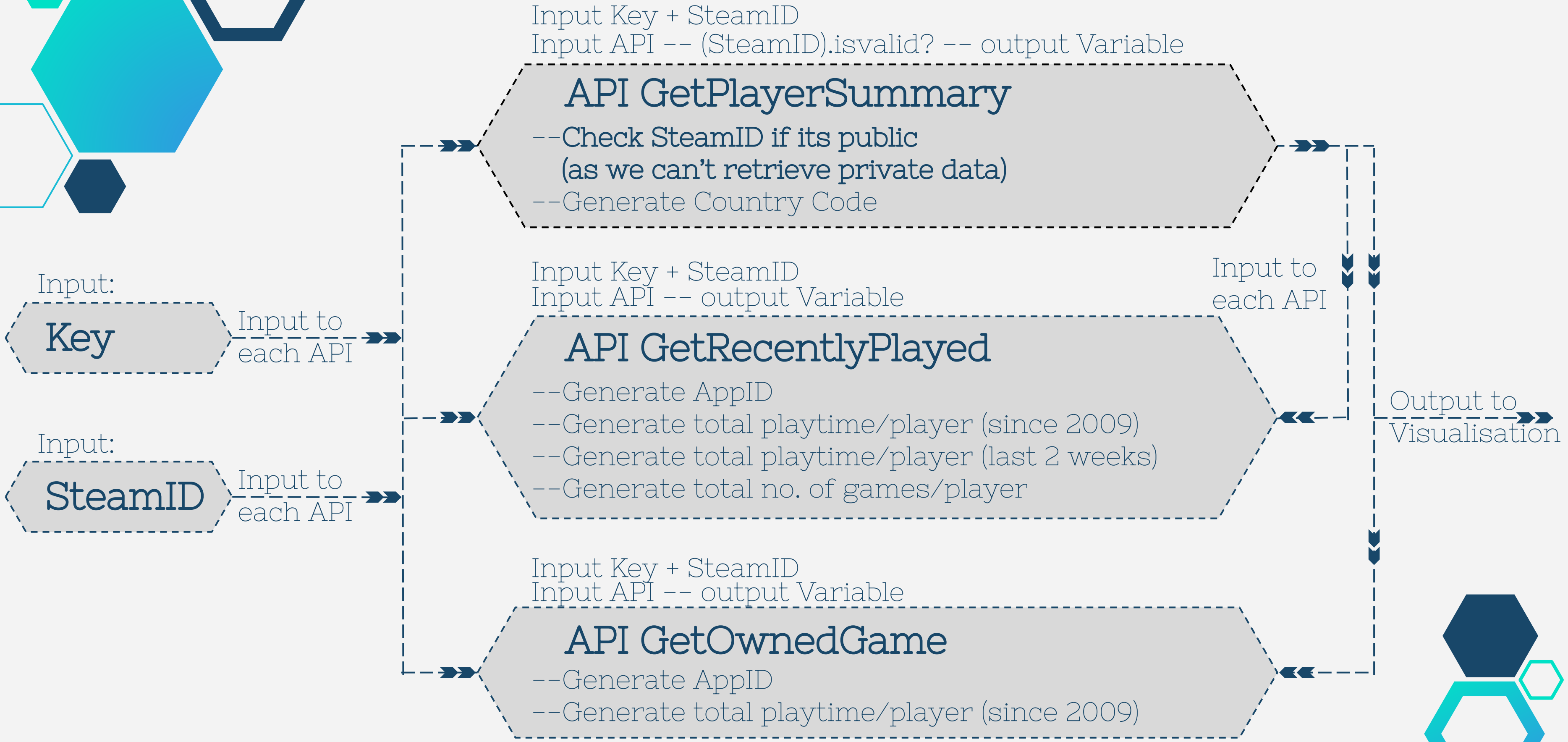
Total Game Users

Under-tapped

Plenty of Games, Plenty of Players with Under-tapped **Raw** Data

Data Collection Strategy

“With the basic data collection framework above, we elaborate and refine code efficiency to
1) Generate lists of SteamID to filter for valid ones
2) Match SteamID to AppID to generate data
3) Generate Variables”

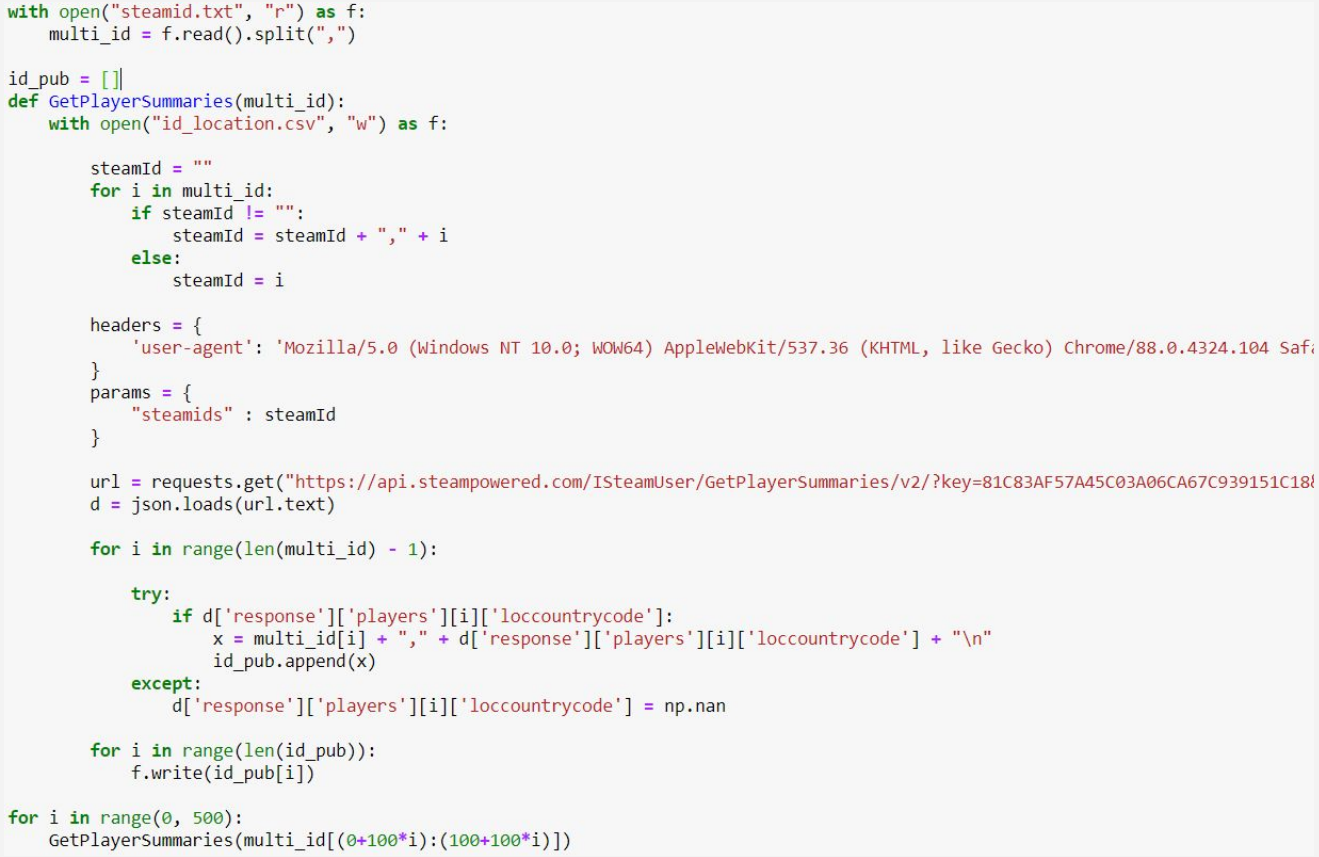

Input:
Key
Input to each API
Input:
SteamID
Input to each API
Input Key + SteamID
Input API -- (SteamID).isvalid? -- output Variable
API GetPlayerSummary
--Check SteamID if its public
(as we can't retrieve private data)
--Generate Country Code
Input Key + SteamID
Input API -- output Variable
API GetRecentlyPlayed
--Generate AppID
--Generate total playtime/player (since 2009)
--Generate total playtime/player (last 2 weeks)
--Generate total no. of games/player
Input Key + SteamID
Input API -- output Variable
API GetOwnedGame
--Generate AppID
--Generate total playtime/player (since 2009)
Input to each API
Output to Visualisation

Project Framework -- Data Collection

Key

Input to each API

- Key is a passcode to access API and Steam Games
- Challenge: Only allow 100,000 Requests per day
- Solution: Input 100 SteamID per request instead of 1 SteamID each time (GetPlayer Summaries)
  : Request keys from friends to assist
- e.g. 72797CA67785C46C4DDB70C6F4C295D3

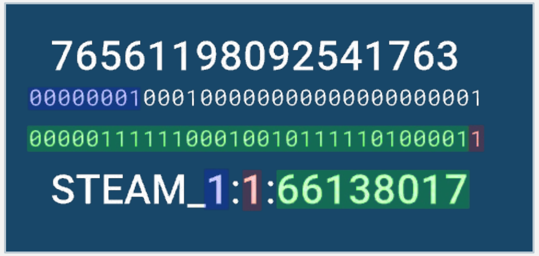


```python
with open("steamid.txt", "r") as f:
    multi_id = f.read().split(",")

id_pub = []
def GetPlayerSummaries(multi_id):
    with open("id_location.csv", "w") as f:

        steamId = ""
        for i in multi_id:
            if steamId != "":
                steamId = steamId + "," + i
            else:
                steamId = i

        headers = {
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Saf
        }
        params = {
            "steamids" : steamId
        }

        url = requests.get("https://api.steampowered.com/ISteamUser/GetPlayerSummaries/v2/?key=81C83AF57A45C03A06CA67C939151C18
        d = json.loads(url.text)

        for i in range(len(multi_id) - 1):

            try:
                if d['response']['players'][i]['loccountrycode']:
                    x = multi_id[i] + "," + d['response']['players'][i]['loccountrycode'] + "\n"
                    id_pub.append(x)
            except:
                d['response']['players'][i]['loccountrycode'] = np.nan

        for i in range(len(id_pub)):
            f.write(id_pub[i])

for i in range(0, 500):
    GetPlayerSummaries(multi_id[(0+100*i):(100+100*i)])
```

SteamID

Input to each API

- It is a user ID
- Challenge: Only 1% of SteamID is valid
  Range is too large (17 digit range) to run for loop
- Solution: Convert to 64bit(17 dig)
- e.g. 76561197960265728



76561198092541763

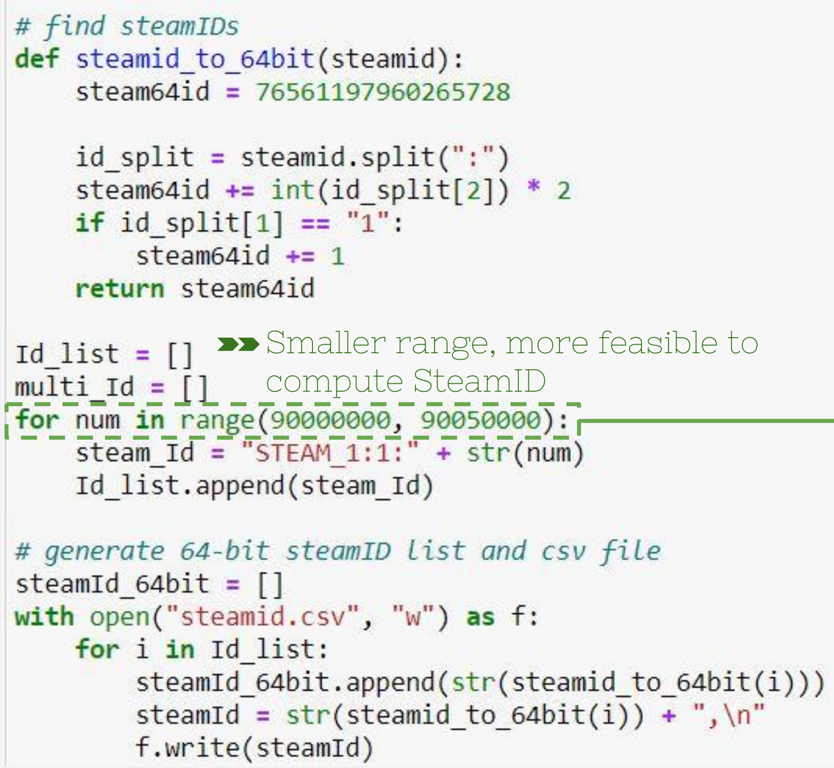
00000001000100000000000000000001

00000111111000100101111101000011

STEAM_1:1:66138017

SteamID

Solution: Use this code to convert the original user id into SteamID, using a pattern to loop computationally generate 8 digit and convert to 64bit(17 dig)



```python
# find steamIDs
def steamid_to_64bit(steamid):
    steam64id = 76561197960265728

    id_split = steamid.split(":")
    steam64id += int(id_split[2]) * 2
    if id_split[1] == "1":
        steam64id += 1
    return steam64id

Id_list = []
multi_Id = []
for num in range(90000000, 90050000):
    steam_Id = "STEAM_1:1:" + str(num)
    Id_list.append(steam_Id)

# generate 64-bit steamID list and csv file
steamId_64bit = []
with open("steamid.csv", "w") as f:
    for i in Id_list:
        steamId_64bit.append(str(steamid_to_64bit(i)))
        steamId = str(steamid_to_64bit(i)) + ",\n"
        f.write(steamId)
```

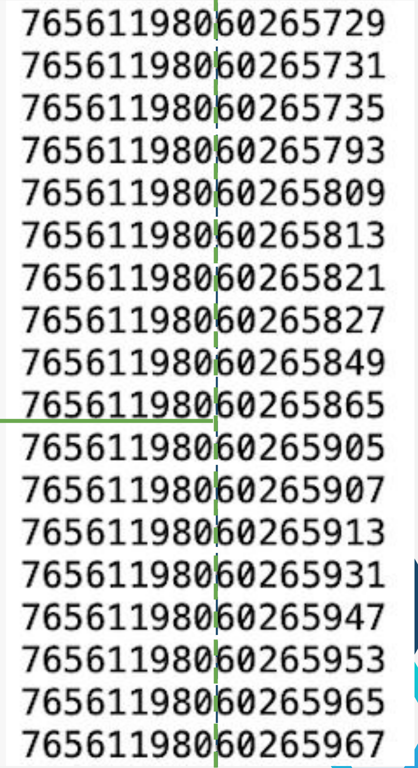
Smaller range, more feasible to compute SteamID

SteamID

Input to each API

64bit (17 digit)

8 digit



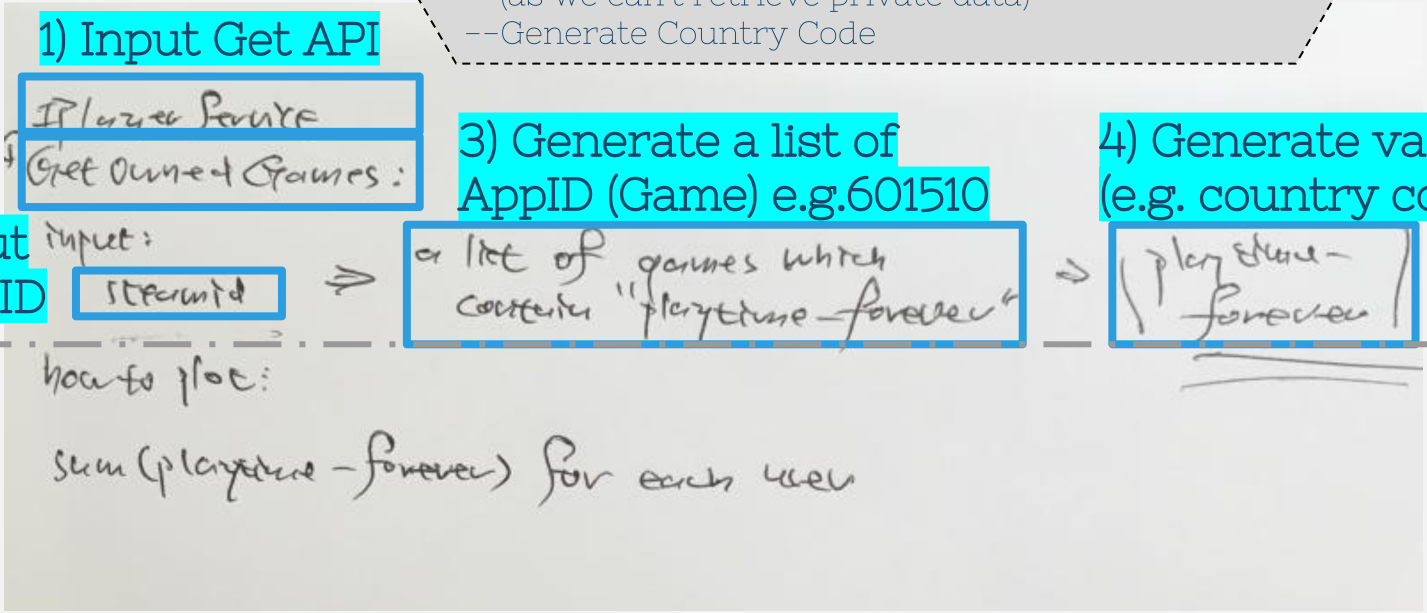
```
76561198060265729
76561198060265731
76561198060265735
76561198060265793
76561198060265809
76561198060265813
76561198060265821
76561198060265827
76561198060265849
76561198060265865
76561198060265905
76561198060265907
76561198060265913
76561198060265931
76561198060265947
76561198060265953
76561198060265965
76561198060265967
```

SteamID

Get API - Code

API GetPlayerSummary

--Check SteamID if its public
(as we can't retrieve private data)
--Generate Country Code



1) Input Get API

IPlayer Service
Get Owned Games :

2) Input SteamID

input :
steamid ⇒

3) Generate a list of AppID (Game) e.g.601510

a list of games which contain "playtime_forever"

⇒

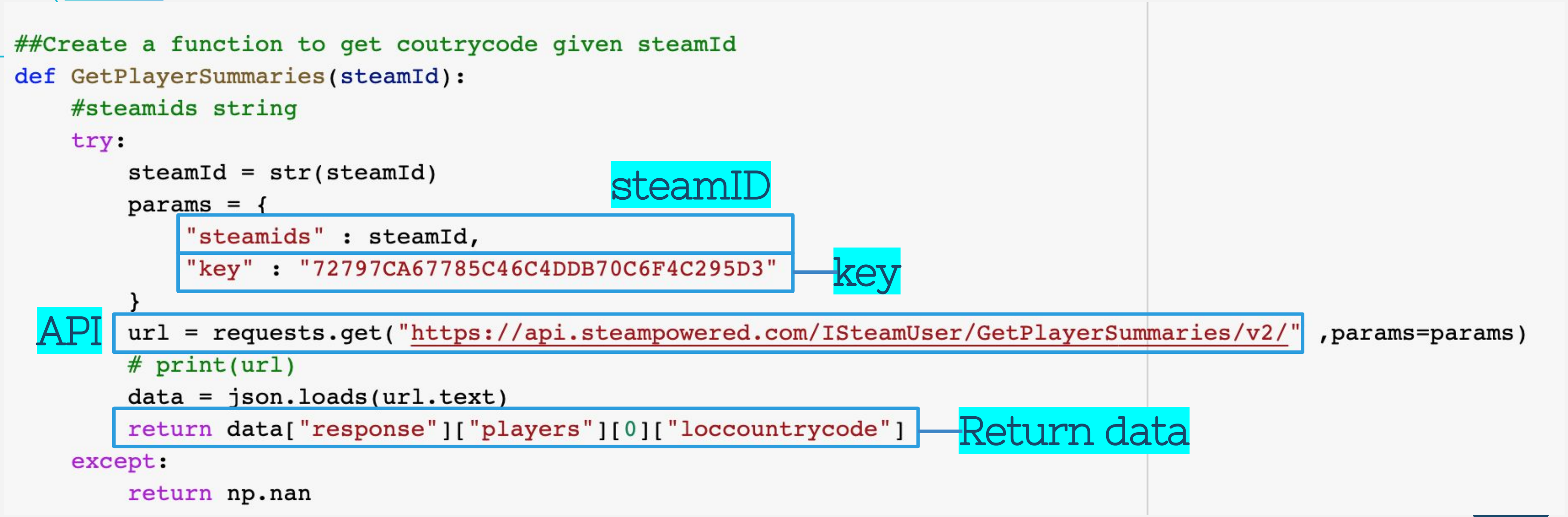
4) Generate variable (e.g. country code)

(playtime_forever)

how to plot:

sum(playtime_forever) for each user

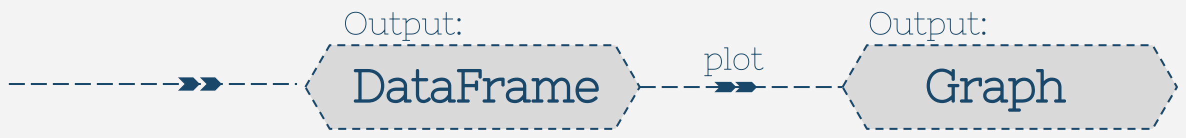
Code :



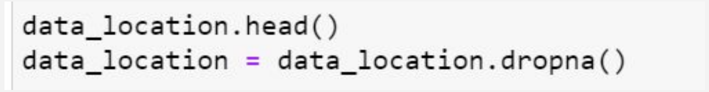
```
##Create a function to get coutrycode given steamId
def GetPlayerSummaries(steamId):
    #steamids string
    try:
        steamId = str(steamId)
        params = {
            "steamids" : steamId,
            "key" : "72797CA67785C46C4DDB70C6F4C295D3"
        }
        url = requests.get("https://api.steampowered.com/ISteamUser/GetPlayerSummaries/v2/" ,params=params)
        # print(url)
        data = json.loads(url.text)
        return data["response"]["players"][0]["loccountrycode"]
    except:
        return np.nan
```

02

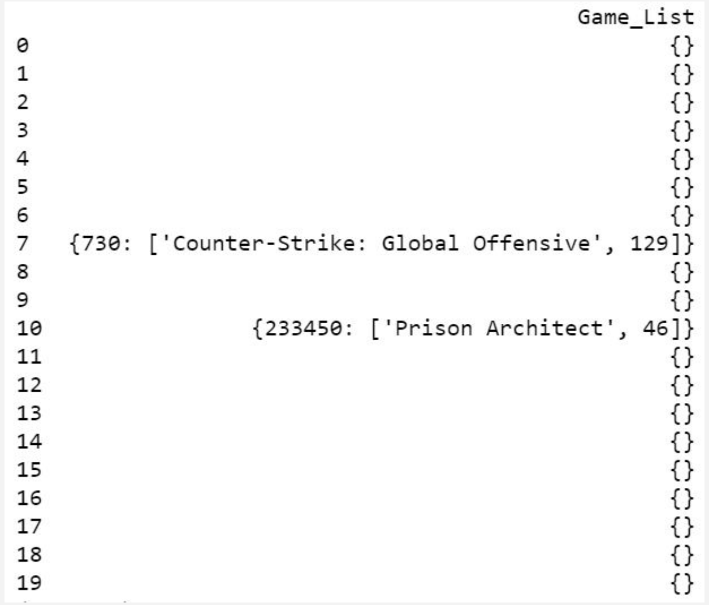
Visualisation & Insights



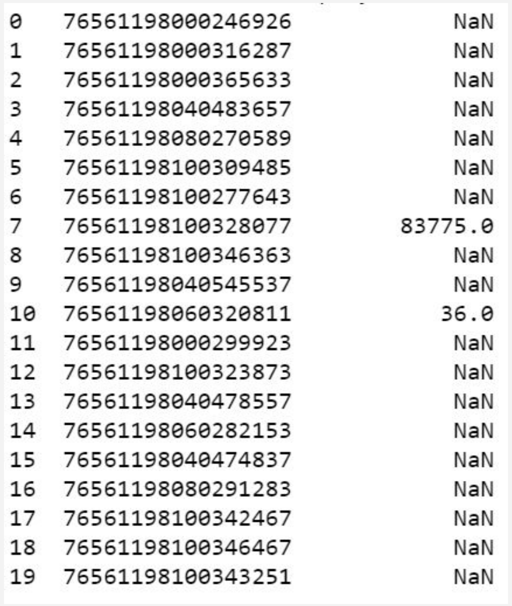
Project Framework -- Data Visualization



```
data_location.head()
data_location = data_location.dropna()
```



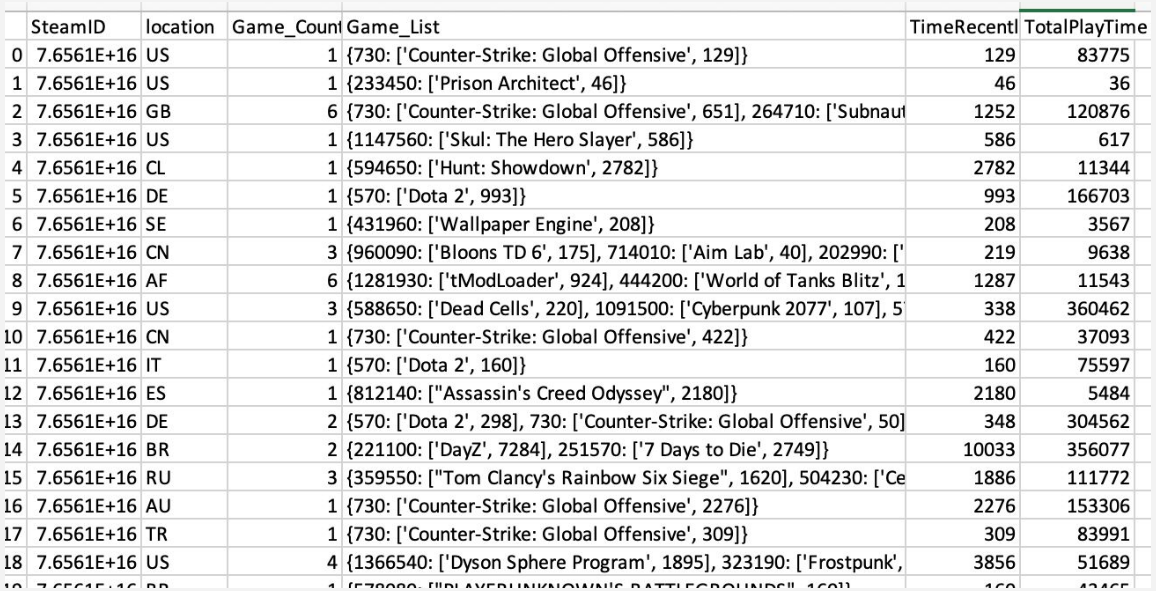
```
                                               Game_List
0                                                     {}
1                                                     {}
2                                                     {}
3                                                     {}
4                                                     {}
5                                                     {}
6                                                     {}
7   {730: ['Counter-Strike: Global Offensive', 129]}
8                                                     {}
9                                                     {}
10                  {233450: ['Prison Architect', 46]}
11                                                    {}
12                                                    {}
13                                                    {}
14                                                    {}
15                                                    {}
16                                                    {}
17                                                    {}
18                                                    {}
19                                                    {}
```



```
0   76561198000246926           NaN
1   76561198000316287           NaN
2   76561198000365633           NaN
3   76561198040483657           NaN
4   76561198080270589           NaN
5   76561198100309485           NaN
6   76561198100277643           NaN
7   76561198100328077       83775.0
8   76561198100346363           NaN
9   76561198040545537           NaN
10  76561198060320811          36.0
11  76561198000299923           NaN
12  76561198100323873           NaN
13  76561198040478557           NaN
14  76561198060282153           NaN
15  76561198040474837           NaN
16  76561198080291283           NaN
17  76561198100342467           NaN
18  76561198100346467           NaN
19  76561198100343251           NaN
```

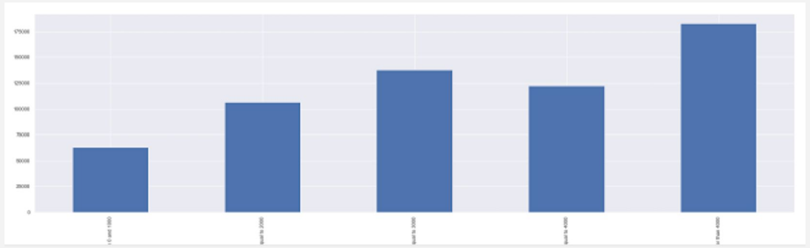
DataFrame – Data Cleaning

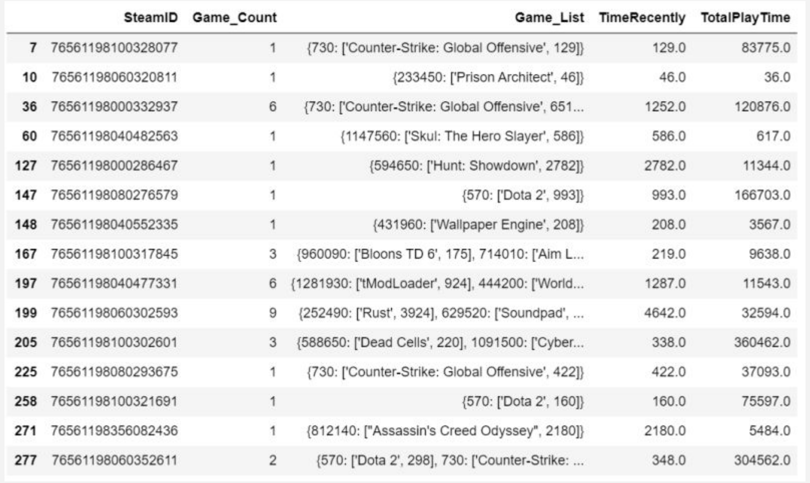
Plot to csv file:



	SteamID	location	Game_Count	Game_List	TimeRecentl	TotalPlayTime
0	7.6561E+16	US	1	{730: ['Counter-Strike: Global Offensive', 129]}	129	83775
1	7.6561E+16	US	1	{233450: ['Prison Architect', 46]}	46	36
2	7.6561E+16	GB	6	{730: ['Counter-Strike: Global Offensive', 651], 264710: ['Subnaut	1252	120876
3	7.6561E+16	US	1	{1147560: ['Skul: The Hero Slayer', 586]}	586	617
4	7.6561E+16	CL	1	{594650: ['Hunt: Showdown', 2782]}	2782	11344
5	7.6561E+16	DE	1	{570: ['Dota 2', 993]}	993	166703
6	7.6561E+16	SE	1	{431960: ['Wallpaper Engine', 208]}	208	3567
7	7.6561E+16	CN	3	{960090: ['Bloons TD 6', 175], 714010: ['Aim Lab', 40], 202990: ['	219	9638
8	7.6561E+16	AF	6	{1281930: ['tModLoader', 924], 444200: ['World of Tanks Blitz', 1	1287	11543
9	7.6561E+16	US	3	{588650: ['Dead Cells', 220], 1091500: ['Cyberpunk 2077', 107], 5	338	360462
10	7.6561E+16	CN	1	{730: ['Counter-Strike: Global Offensive', 422]}	422	37093
11	7.6561E+16	IT	1	{570: ['Dota 2', 160]}	160	75597
12	7.6561E+16	ES	1	{812140: ["Assassin's Creed Odyssey", 2180]}	2180	5484
13	7.6561E+16	DE	2	{570: ['Dota 2', 298], 730: ['Counter-Strike: Global Offensive', 50]	348	304562
14	7.6561E+16	BR	2	{221100: ['DayZ', 7284], 251570: ['7 Days to Die', 2749]}	10033	356077
15	7.6561E+16	RU	3	{359550: ["Tom Clancy's Rainbow Six Siege", 1620], 504230: ['Ce	1886	111772
16	7.6561E+16	AU	1	{730: ['Counter-Strike: Global Offensive', 2276]}	2276	153306
17	7.6561E+16	TR	1	{730: ['Counter-Strike: Global Offensive', 309]}	309	83991
18	7.6561E+16	US	4	{1366540: ['Dyson Sphere Program', 1895], 323190: ['Frostpunk',	3856	51689
19	7.6561E+16	BR	1	{578080: ["PLAYERUNKNOWN'S BATTLEGROUNDS", 160]}	160	43465

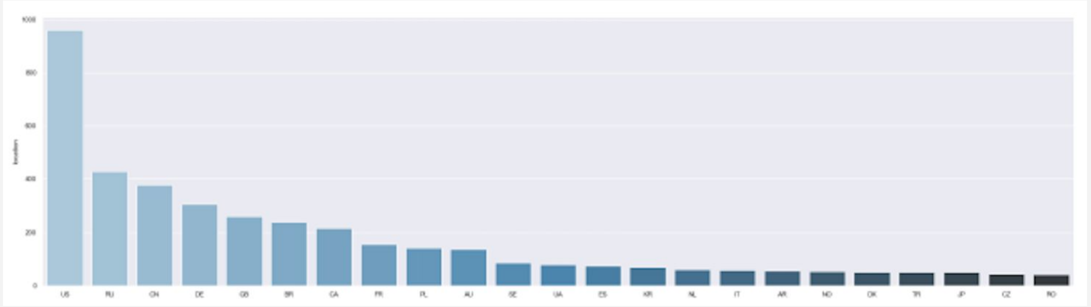
DataFrame – Data Cleaning

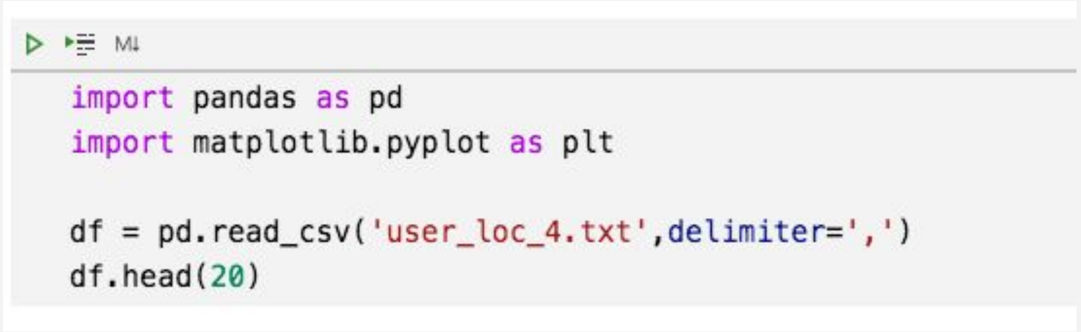




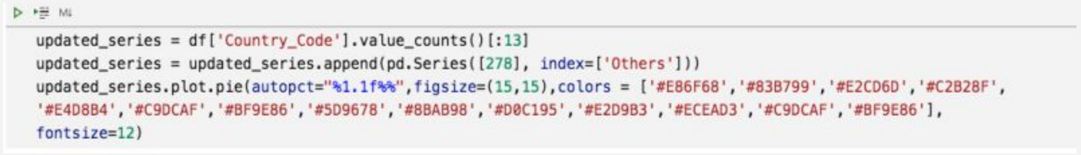
	SteamID	Game_Count	Game_List	TimeRecently	TotalPlayTime
7	76561198100328077	1	{730: ['Counter-Strike: Global Offensive', 129]}	129.0	83775.0
10	76561198060320811	1	{233450: ['Prison Architect', 46]}	46.0	36.0
36	76561198000332937	6	{730: ['Counter-Strike: Global Offensive', 651...	1252.0	120876.0
60	76561198040482563	1	{1147560: ['Skul: The Hero Slayer', 586]}	586.0	617.0
127	76561198000286467	1	{594650: ['Hunt: Showdown', 2782]}	2782.0	11344.0
147	76561198080276579	1	{570: ['Dota 2', 993]}	993.0	166703.0
148	76561198040552335	1	{431960: ['Wallpaper Engine', 208]}	208.0	3567.0
167	76561198100317845	3	{960090: ['Bloons TD 6', 175], 714010: ['Aim L...	219.0	9638.0
197	76561198040477331	6	{1281930: ['tModLoader', 924], 444200: ['World...	1287.0	11543.0
199	76561198060302593	9	{252490: ['Rust', 3924], 629520: ['Soundpad', ...	4642.0	32594.0
205	76561198100302601	3	{588650: ['Dead Cells', 220], 1091500: ['Cyber...	338.0	360462.0
225	76561198080293675	1	{730: ['Counter-Strike: Global Offensive', 422]}	422.0	37093.0
258	76561198100321691	1	{570: ['Dota 2', 160]}	160.0	75597.0
271	76561198356082436	1	{812140: ["Assassin's Creed Odyssey", 2180]}	2180.0	5484.0
277	76561198060352611	2	{570: ['Dota 2', 298], 730: ['Counter-Strike: ...	348.0	304562.0

DataFrame via Panda & Seaborn

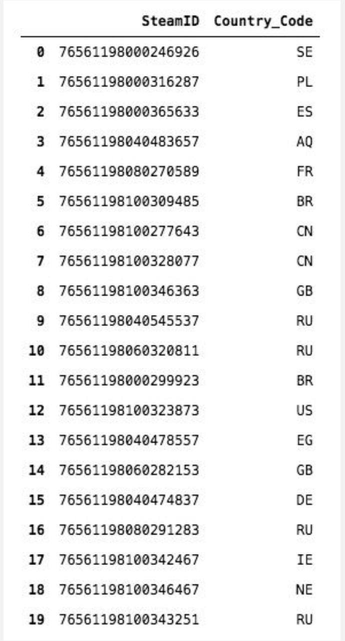




```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('user_loc_4.txt',delimiter=',')
df.head(20)
```



```
updated_series = df['Country_Code'].value_counts()[:13]
updated_series = updated_series.append(pd.Series([278], index=['Others']))
updated_series.plot.pie(autopct="%1.1f%%",figsize=(15,15),colors = ['#E86F68','#83B799','#E2CD6D','#C2B28F',
'#E4D8B4','#C9DCAF','#BF9E86','#5D9678','#8BAB98','#D0C195','#E2D9B3','#ECEAD3','#C9DCAF','#BF9E86'],
fontsize=12)
```



	SteamID	Country_Code
0	76561198000246926	SE
1	76561198000316287	PL
2	76561198000365633	ES
3	76561198040483657	AQ
4	76561198080270589	FR
5	76561198100309485	BR
6	76561198100277643	CN
7	76561198100328077	CN
8	76561198100346363	GB
9	76561198040545537	RU
10	76561198060320811	RU
11	76561198000299923	BR
12	76561198100323873	US
13	76561198040478557	EG
14	76561198060282153	GB
15	76561198040474837	DE
16	76561198080291283	RU
17	76561198100342467	IE
18	76561198100346467	NE
19	76561198100343251	RU

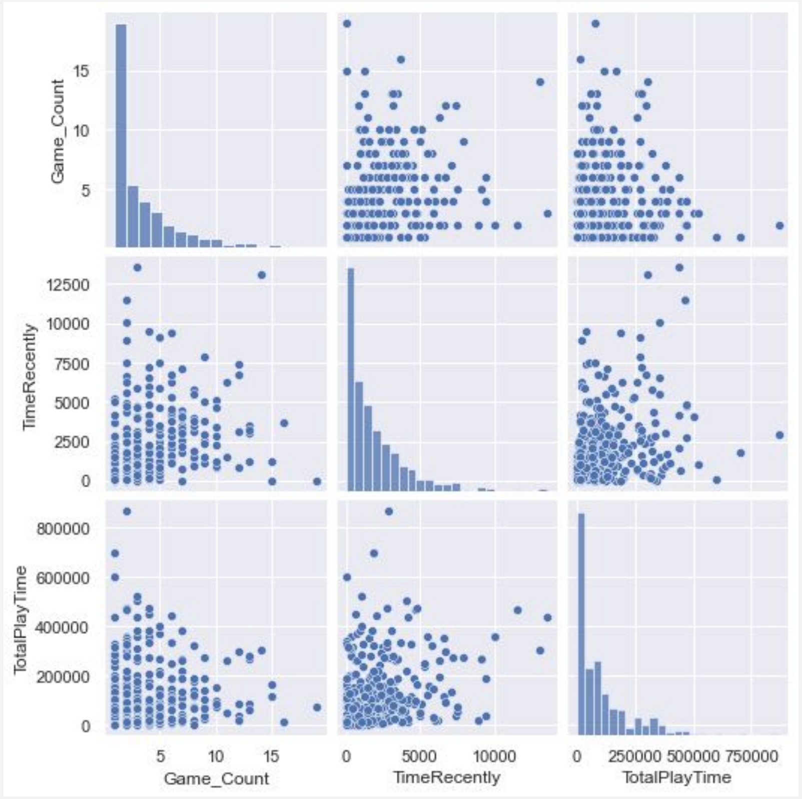
DataFrame via Panda & Seaborn

Visualisation Outcome

“With existing 5000 valid SteamID, we plot 4 graphs as followings, to share our insights”

Observation:

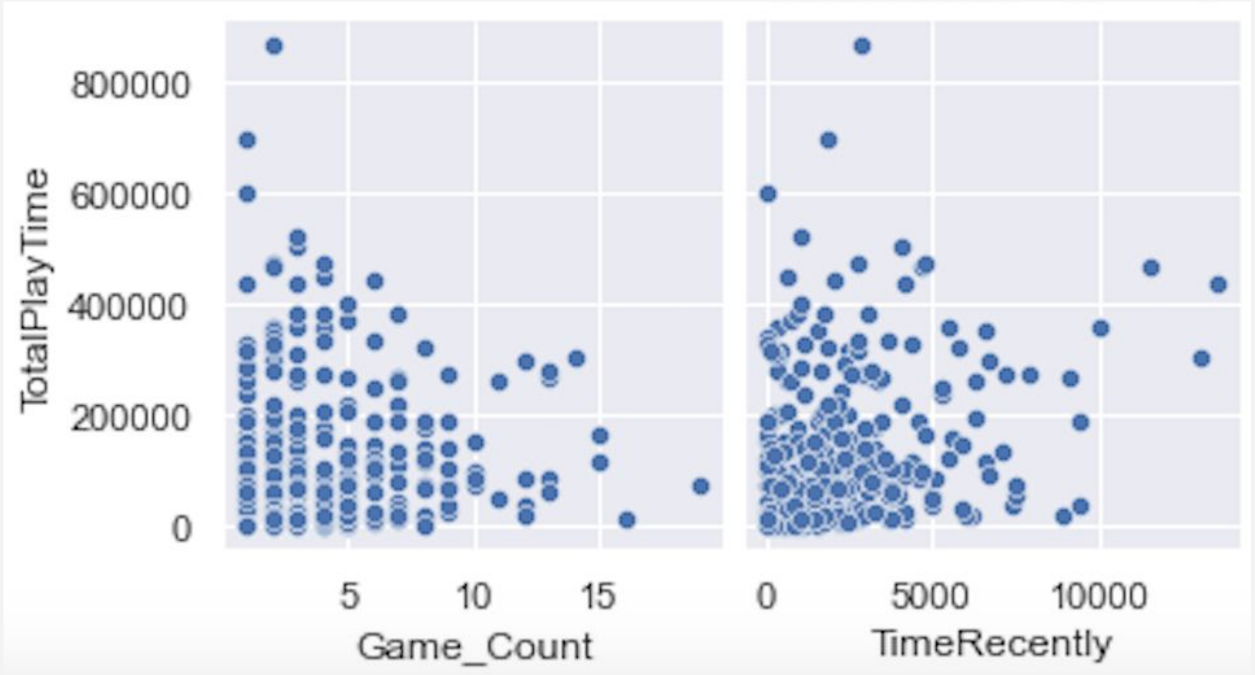
The graph reflects the gamer habit



Visualisation 3

Observation:

The graph reflects the gamer habit

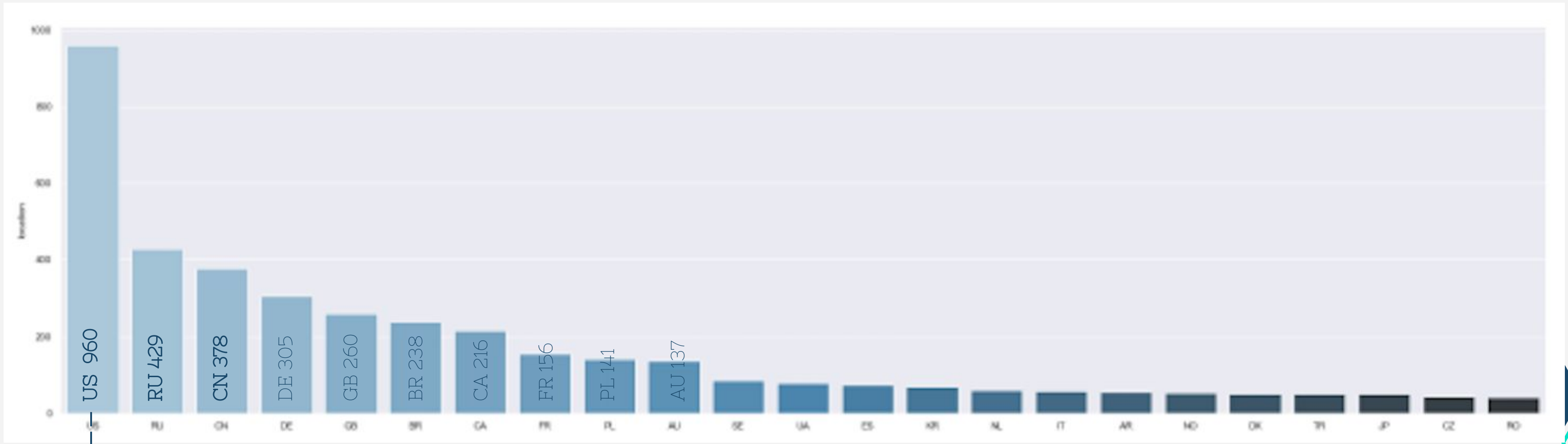


Visualisation 3

Our suggestion:

We suggest to include English, Russian, and Mandarin to game development as US, Russia, and China has the most populated concentrated players

*Similar data to the pie chart behind , current bar chart , in a more visually eye catching format, shows US has a larger player population, twice more than Russia.



In the US, there are 960 players out of 5000 total players, which is 18.8% of total players

Visualisation 2

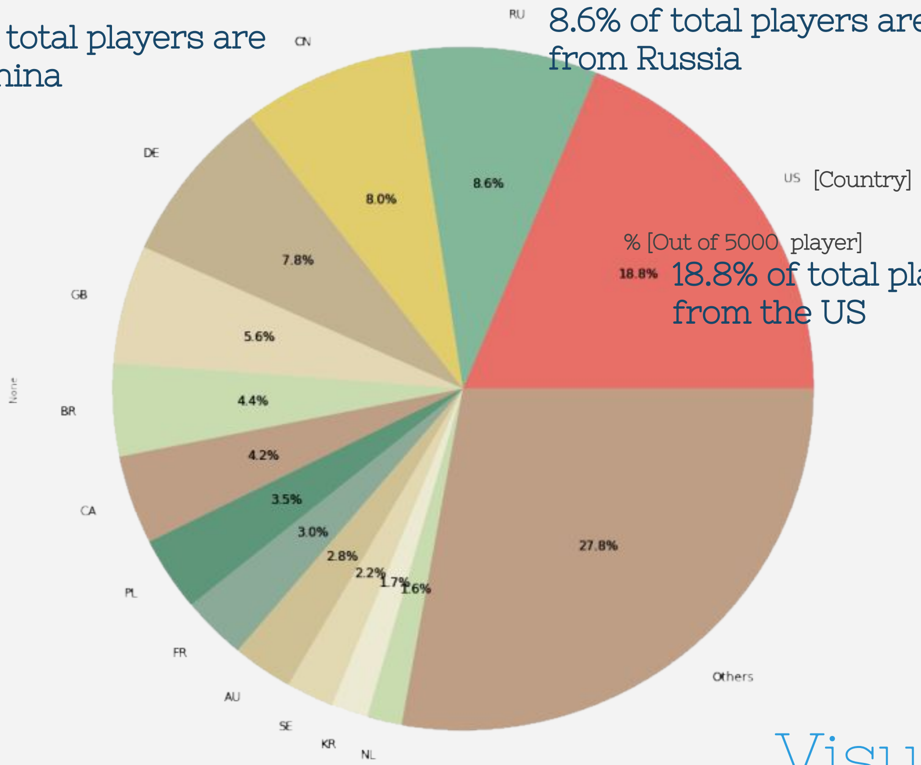
Our suggestion:

Please consider to continue to develop Steam online game in the US region - a potential lightly-tapped market

8.0% of total players are from China

8.6% of total players are from Russia

18.8% of total players are from the US



Visualisation 1 .

Our suggestion:

We suggested the US region due to its consistent revenue growth with +7.3% on the year 2020, high spending power, and a leading game industry (as well as, as a profession).

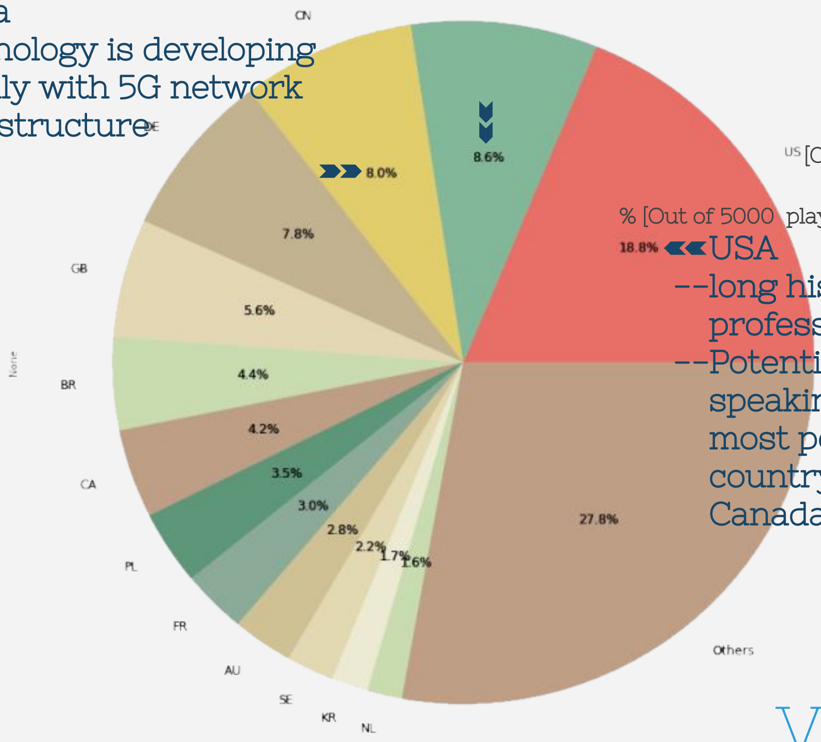
Russia

China
--Technology is developing rapidly with 5G network infrastructure

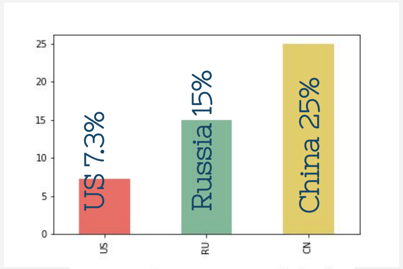
[Country]

% [Out of 5000 player]

USA
--long history of gaming as profession
--Potentially tapping English speaking market within the top 10 most populated gamers per country such as England(5.8%), Canada(4.2%), and Australia(2.8%)



Revenue Growth



```
countries  growth(%)
       US        7.3
       RU       15.0
       CN       25.0
```

Visualisation 1

THINK
BIGGER

Moving Forward:

Scraping game player keyword such as “game soundtrack” , “story rich”, “female protagonist”; on strategy game genre, for example.

Such content will be easier to find by speaking the customer’s language.

Thank You !

hello@teamsteam.co

github.com/teamsteam