Mobile Game Business Analysis

This project is about business analysis for mobile apps \ games. We'll look at 2 important KPI's, analyze abnormalities and discuss the concerns and benefits of implementing a new feature in an app.

1. Sudden decrease in revenue on a given day, we'll discuss the probable causes and get insights from the data.
2. Finding suspicious activity in the app \ game, we'll analyze suspicious behaviour and find users who might be cheating.
3. Implementing a new feature in the game, the course of A\B testing and key points. this part wont include code but it includes my analysis and methodology regarding A\B testing and the implementatino of new features.

Importing relevant libraries and getting rid of scientific notations

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import math
import scipy as sp
import matplotlib as mpl
import matplotlib.pyplot as plt
import mpl_toolkits
from mpl_toolkits.mplot3d import axes3d
import os
import datetime as dt

import time as t

#No sci-pen
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

Importing DB and light light exploration

In [3]:
```
input_file = 'NP_Business_Analysis.csv'

def get_df(file):
    ds = pd.read_csv(file)
    df = pd.DataFrame(ds)
    df = df.fillna(0)
    return df

# Call the function
df = get_df(input_file)
```

In [4]:
```
#Turning date columns to date format

df['dim_t'] = pd.to_datetime(df['agg_date']).apply(lambda x: x.date())


df['install_t'] = pd.to_datetime(df['install_date']).apply(lambda x: x.date())
```

In [5]:
```
rows = df.shape[0]
users = df.nunique()[0]

print(f'Number of users on DS is\n{users}\nand the number of rows is\n{rows}\n\n(Obviously I would not disclose real IDs

#Date for part 1:
d24 = dt.date(2020,12,24)
```

```
Number of users on DS is
699661
and the number of rows is
699661

(Obviously I would not disclose real IDs and its all unique!)
```

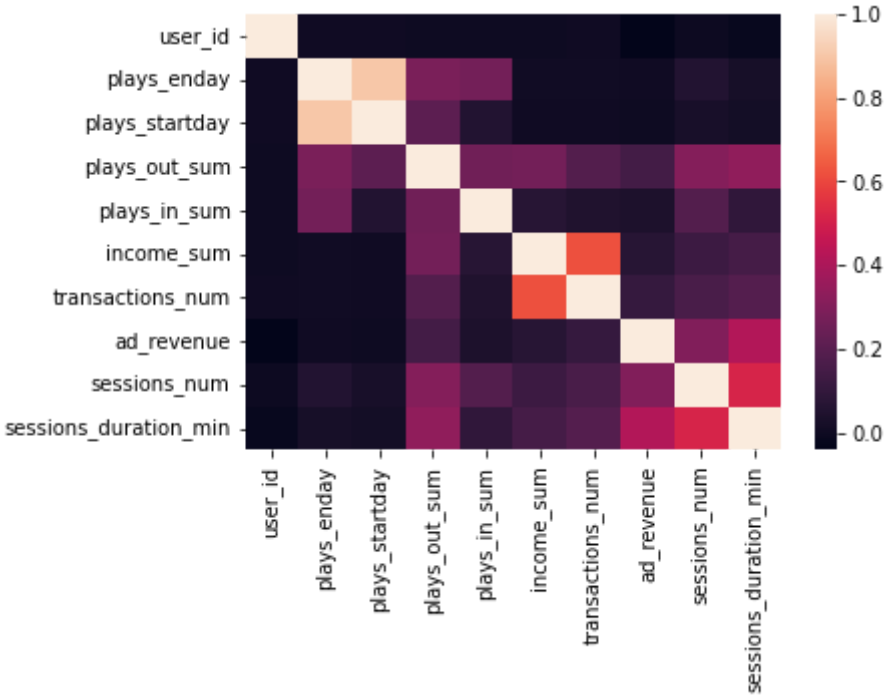
In [6]:
```
df.head(3)
```

Out[6]:

	user_id	agg_date	platform	install_date	plays_enday	plays_startday	plays_out_sum	plays_in_sum	income_sum	transactions_num	ad_revenu
0	1	13/12/2020	android	09/11/2020	429	0	10425	11423	3.980	2	0.57
1	2	13/12/2020	ios	28/11/2020	2	236	15849	16012	7.970	3	0.14
2	3	13/12/2020	android	19/11/2020	3	236	21974	22039	13.960	4	0.02

Correlations

In [7]:
```
def corr_table(columns):
    return sns.heatmap(columns.corr())

corr_table(df[df.columns]);
```



Correlations summary:

1. There are no negative correlations in the data.
2. the number of plays is highly correlated to the number of plays at the beggining of a day. Probably due to the fact that people who end their day with alot of plays tend to also begin the next with a high number of plays.
3. the sessions length is positively but weakly correlated to the number of plays the user played, which indicates that this might be the main activity in the game.
4. the duration and count of sessions is strongly correlated which suggests that people who play more are also playing longer, and indicates a high tier players with a high level of engagement.

In [8]:
```
df.dtypes
```

Out[8]:
```
user_id                 int64
agg_date               object
platform               object
install_date           object
plays_enday             int64
plays_startday          int64
plays_out_sum           int64
plays_in_sum            int64
income_sum            float64
transactions_num        int64
ad_revenue            float64
sessions_num            int64
sessions_duration_min float64
dim_t                  object
install_t              object
dtype: object
```

In [9]:
```
df.describe()
```

Out[9]:

	user_id	plays_enday	plays_startday	plays_out_sum	plays_in_sum	income_sum	transactions_num	ad_revenue	sessions_num	session
count	699661.000	699661.000	699661.000	699661.000	699661.000	699661.000	699661.000	699661.000	699661.000	
mean	349831.000	983.373	748.545	1391.959	1489.648	0.554	0.100	0.107	7.109	
std	201974.878	83055.578	75426.635	4656.941	18977.236	5.876	0.723	0.372	7.972	
min	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
25%	174916.000	2.000	0.000	246.000	302.000	0.000	0.000	0.000	3.000	
50%	349831.000	18.000	11.000	661.000	738.000	0.000	0.000	0.000	5.000	
75%	524746.000	116.000	96.000	1359.000	1467.000	0.000	0.000	0.047	9.000	
max	699661.000	11724032.000	11724033.000	662663.000	10984455.000	849.850	65.000	18.583	1674.000	

In [10]:
```
def explore_data (dataframe):
    print('columns\n',dataframe.columns)
    print('shape\n', dataframe.shape)

# Call the function

explore_data(df)
```

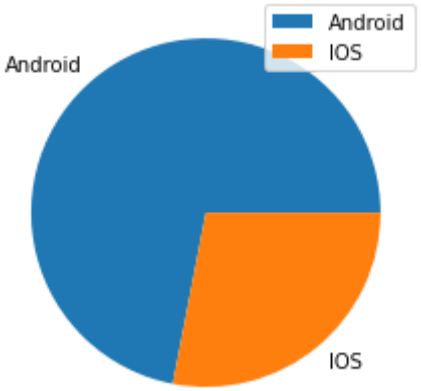
```
columns
 Index(['user_id', 'agg_date', 'platform', 'install_date', 'plays_enday',
       'plays_startday', 'plays_out_sum', 'plays_in_sum', 'income_sum',
       'transactions_num', 'ad_revenue', 'sessions_num',
       'sessions_duration_min', 'dim_t', 'install_t'],
      dtype='object')
shape
 (699661, 15)
```

Part 1 - Sudden Decrease in income on December 24

OS Distribution

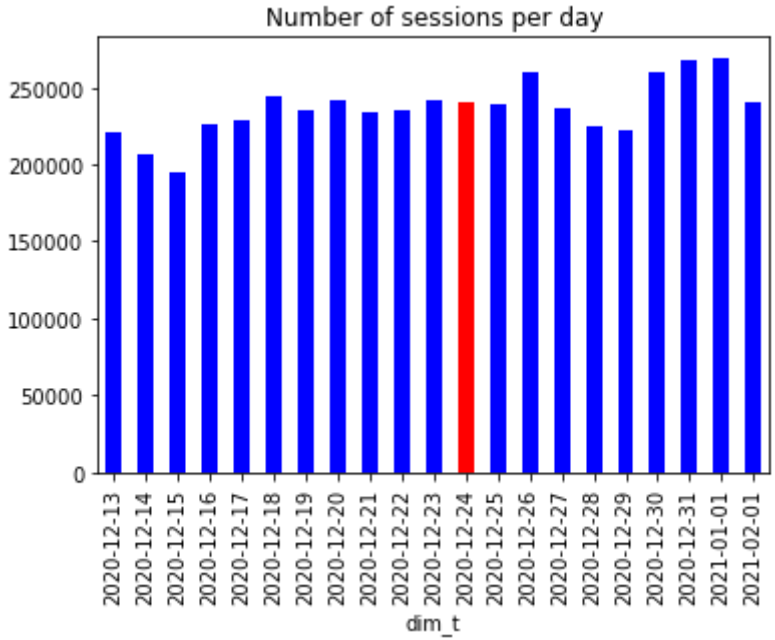
In [11]:
```
# Getting the number of unique occurances of 'platform'.
# Getting the percentage of each appearance.
# plotting the appearance.

uplatforms = df.groupby('platform')['user_id'].nunique()

android_p = round((uplatforms[0] / users) * 100,2)
ios_p = round((uplatforms[1]  / users) * 100,2)

pie_arr = np.array([android_p,ios_p])
labels_pie = ['Android','IOS']

print(f'Android: {android_p}%\nIOS: {ios_p}%')

plt.pie(pie_arr, labels = labels_pie)
plt.legend()
plt.show()
```

```
Android: 71.97%
IOS: 28.03%
```



Number of Sessions

In [12]:
```
#Getting the number of sessions played on the 24th and the number of unique users

#Get total of sessions on the 24th

sessions_per_day = df.groupby('dim_t')['sessions_num'].sum()

colors = ['b']

sspdp = sessions_per_day.plot(kind='bar',title='Number of sessions per day',color=colors)

#color a column function!

def coloring_colum(df, plots, target):
    for bar in plots.patches:
        bar.set_facecolor('b')

    highlight = target
    pos = df.index.get_loc(highlight)

    plots.patches[pos].set_facecolor('r')



coloring_colum(sessions_per_day,sspdp,d24)
```


Number of sessions per day
250000
200000
150000
100000
50000
0
2020-12-13
2020-12-14
2020-12-15
2020-12-16
2020-12-17
2020-12-18
2020-12-19
2020-12-20
2020-12-21
2020-12-22
2020-12-23
2020-12-24
2020-12-25
2020-12-26
2020-12-27
2020-12-28
2020-12-29
2020-12-30
2020-12-31
2021-01-01
2021-02-01
dim_t

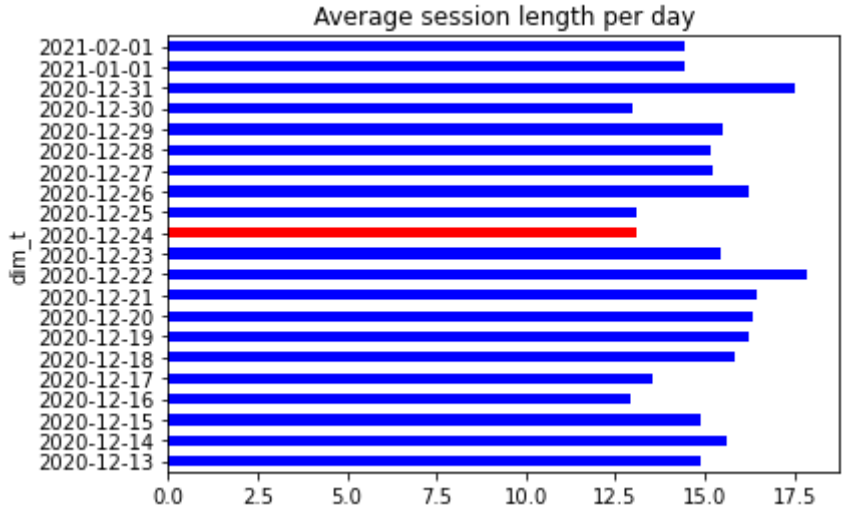
In [13]:
```
sessions_total = df.groupby('dim_t')['sessions_num'].sum()
sessions_median = round(sessions_total.median(),2)
sessions_avg = round(sessions_total.mean(),2)

rule1 = (df['dim_t'] == d24)

date24 = df.loc[rule1]

sum_sessions24 = date24['sessions_num'].sum()

print(f'The Average amount of sessions per day is: {sessions_avg}\nThe Median is: {sessions_median}\nWhile the total ses

#Get the number of unique users who played on the 24th
rule2 = (df['dim_t'] == d24) & (df['sessions_num'] > 0)
date24_2 = df.loc[rule2]

uniq24 = date24_2.nunique()[0]

print(f'\nOn the 24th there were {uniq24} different users playing')
```

```
The Average amount of sessions per day is: 236842.29
The Median is: 236299.0
While the total sessions on the 24th: 240223

On the 24th there were 33973 different users playing
```

Length of Sessions

In [14]:
```
length_per_d = round((df.groupby('dim_t')['sessions_duration_min'].sum() / df.groupby('dim_t')['sessions_num'].sum()),2)
lpdp = length_per_d.plot(kind='barh',title='Average session length per day',color=colors)

coloring_colum(length_per_d,lpdp,d24)

length_per_d24 = round((date24.groupby('dim_t')['sessions_duration_min'].sum() / date24.groupby('dim_t')['sessions_num']
length_per_d_mean = round(length_per_d.mean(),2)
length_per_d_median = round(length_per_d.median(),2)

print(f'Average daily session length is {length_per_d_mean}\nThe median length is {length_per_d_median}\nAnd the length
```

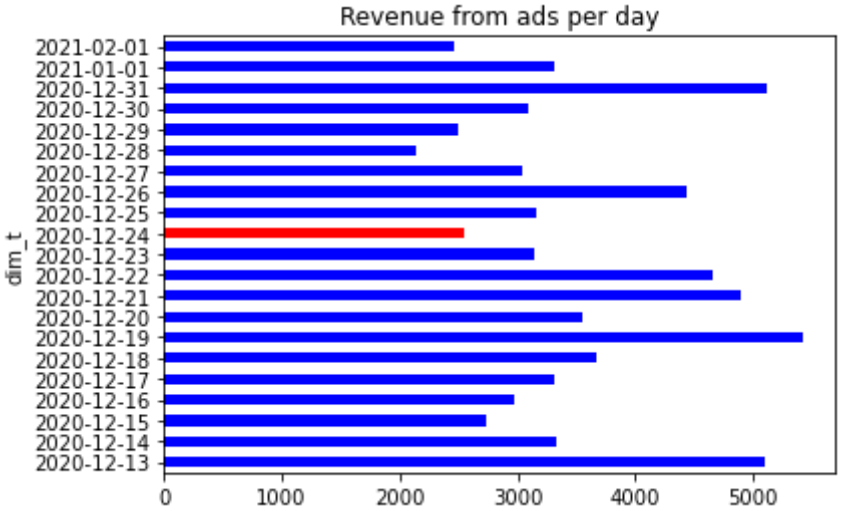
```
Average daily session length is 15.13
The median length is 15.22
And the length on the 24th was 13.11
```



Revenue From Ads

In [15]:
```
#Checking revenue from ads on the 24th

ads_income = df.groupby('dim_t')['ad_revenue'].sum()
aip = ads_income.plot(kind='barh',title='Revenue from ads per day')
ads_avg = round(df.groupby('dim_t')['ad_revenue'].sum().mean(),2)
ads_median = round(df.groupby('dim_t')['ad_revenue'].sum().median(),2)
adr24 = round(date24['ad_revenue'].sum(),2)

coloring_colum(ads_income,aip,d24)

print(f'revenue from ads on the 24th:',adr24)
print(f'Average revenue from ads per day is: {ads_avg}')
print(f'Median revenue from ads (per day) is: {ads_median}')
```

```
revenue from ads on the 24th: 2544.35
Average revenue from ads per day is: 3553.88
Median revenue from ads (per day) is: 3308.15
```

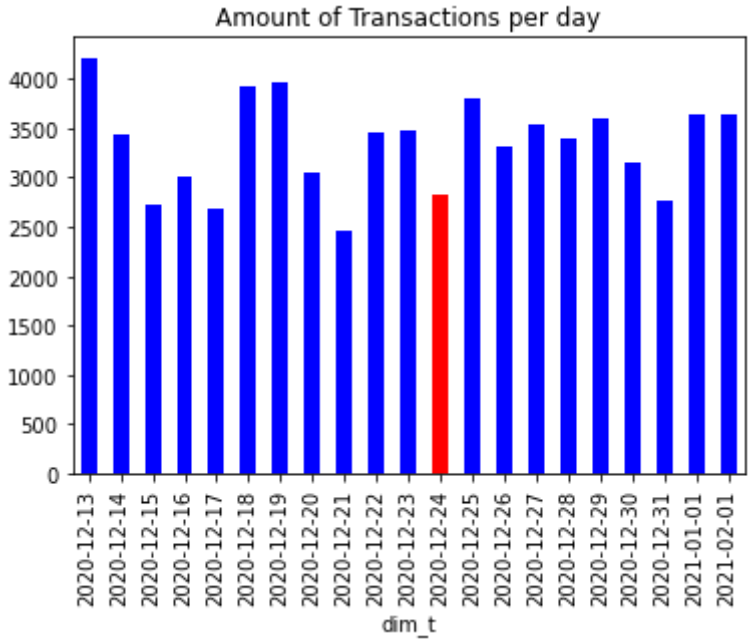


Transactions info - Count & Amount

Trx Count :

In [16]:
```
trxcplot = df.groupby('dim_t')['transactions_num'].sum()
trxp = trxcplot.plot(kind='bar',title='Amount of Transactions per day')
trxc24 = round(date24['transactions_num'].sum(),2)
trxcmean = round(df.groupby('dim_t')['transactions_num'].sum().mean(),2)
trxcmedian = round(df.groupby('dim_t')['transactions_num'].sum().median(),2)

coloring_colum(trxcplot, trxp, d24)

print(f'The Average amount of transactions per day is {trxcmean}\nThe median value of the aggregated amount is {trxcmedi
```

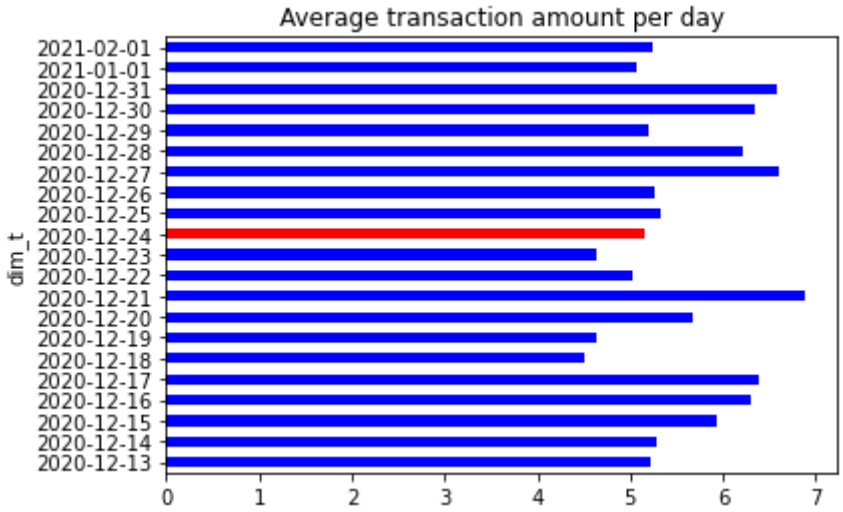
```
The Average amount of transactions per day is 3338.29
The median value of the aggregated amount is 3432.0
While the amount of transactions on the 24th is 2820
```



Trx Amount :

In [17]:
```
r_per_d = df.groupby('dim_t')['income_sum'].sum() / df.groupby('dim_t')['transactions_num'].sum()
rpdp = r_per_d.plot(kind='barh',title='Average transaction amount per day')

coloring_colum(r_per_d, rpdp, d24)

r_per_d24 = round((date24.groupby('dim_t')['income_sum'].sum() / date24.groupby('dim_t')['transactions_num'].sum())[0],2
r_per_d_mean = round(r_per_d.mean(),2)
r_per_d_median = round(r_per_d.median(),2)

print(f'Average amount spent per transaction daily is {r_per_d_mean}\nThe median amount is {r_per_d_median}\nAnd the amou
```

```
Average amount spent per transaction daily is 5.59
The median amount is 5.29
And the amount on the 24th was 5.16
```



Conclusions Part 1 -

Techinical & Externalities -

1. December 24th is Thanksgiving! It's possible that people paid less - they had more expenses.
2. I have found no technical issue, the number of sessions on the 24th was roughly greater than average, indicating no technical issue.
3. The sessions avergae length on the 24th is around 13 minutes, lower than average of 15 minutes.

Revenue & Transactions -

1. Revenue from ads - revenue was ads was considerably lower than the average and the median.
2. The number of transactions - on the 24th, fewer transactions were made which is lower than average and the median.
3. amount per transaction - Amount per transaction was lower on the 24th which is lower than average and the median but not significantly.

Conclusion -

I have found that while nearly all parameters and KPIs were lower than average none were significantly lower as to indicate abnormal activity.

According to the data, the decrease in revenue on the 24th is unlikely because of a technical issue, but it is most probably caused by 'externalities', since december 24th is thanksgiving.

an important holiday such as this one often includes high in-game rewards and surprises as well as many great deals and bundles.

this could explain the lower number of transactions, and the lower revenue from ads (rewarded videos) and also why the amound per transaction is a bit lower.

Part 2

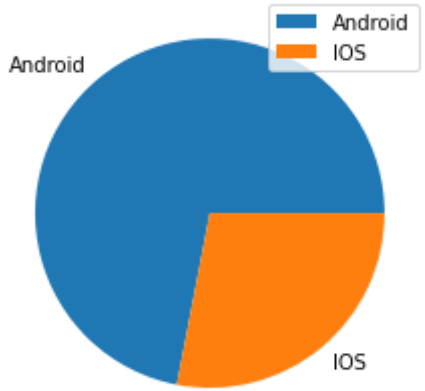
Odd Roll Balance - Cumulative sums of 'plays'.

In [18]:
```
rule_4 = (df.groupby(['user_id','dim_t'])['plays_in_sum'].cumsum() < df.groupby(['user_id','dim_t'])['plays_out_sum'].cu

suspicious_1 = df.loc[rule_4]
#For future use - Array of suspicious IDs and their Platforms
suspicious_ids = suspicious_1[['user_id','platform']]
s_num = suspicious_ids.nunique()[0]

print(f'About {s_num} users are suspicious\nWhihch are about {round((s_num / users),2)}% of our total users')
```

```
About 99833 users are suspicious
Whihch are about 0.14% of our total users
```

In [19]:
```
s_details = suspicious_ids.groupby('platform')['user_id'].nunique()


s_android_p = round((s_details[0] / s_num) * 100,2)
s_ios_p = round((s_details[1]  / s_num) * 100,2)

s_pie_arr = np.array([android_p,ios_p])
s_labels_pie = ['Android','IOS']

print(f'Android: {s_android_p}%\nIOS: {s_ios_p}%')
plt.pie(pie_arr, labels = labels_pie)
plt.legend()
plt.show()
```

```
Android: 71.98%
IOS: 28.02%
```



Conclusions Part 2 -

To identify cheaters, players who have more 'plays' than they should, I calculated the comulative sum of 'plays_in' which keeps tracks of every play users get legally, including winnings and buyings, and according to that logic, while people can save their plays from day to day, they should never

be able to spend more than they earned, comulative speaking.

By identifying which users used more rolls than their total rolls recieved at a given point in time, It's possible that around 14% of users are cheating.

It's also possible to see that the platform distribution among cheaters is similar to the whole.

Part 3 - Implementing 'rewarded videos' new feature into the game. Explaining A\B testing main points and discussing specific action points.

According to many sources, rewarded videos are a big part of mobile game monetization. This feature is known for increasing LTV, increasing engagement, increasing session lengths and even retention. And of course revenue. Since this analysis is about adding a new feature to the game, we don’t know how our users will take to it. I suggest running an A\B Test to determine if the impact of this feature is both positive and wanted.

1. Define success metrics.

- 10% percent increase in revenues? In which user segment? Do we also expect an increase in retention? Longer lt?

2. What are we testing (the impact of each version of this feature).

- Where do we show the ad?
- When do we show the ad?
- What are the rewards for watching the ad?
- Do rewards differ between groups? Levels?

3. Split segments (or parts of segments) to halves, and show one half the original game without the feature, control group, and the other the alternate version, treatment group.

- This will allow us to measure the impact of this change on different segments since we will have data with & without the change to compare.
- This will also allow us to understand the impact of different versions of this feature on different groups.

4. Wait for the data to stack, we want a big enough group which is also representative of the “population”, and we also want to reach high statistical significance. (95%-99%). As analysts we have to take all things into account, and find out if this change could have undesired effects. Pros & Cons – IMO the pros and cons are two sides of the same coin, because they’re similar, but which side will be picked is up to the level of execution of this feature. The pros of rewarded ads are plenty.
5. Churn rates might do down – users are less likely to feel frustrated about the game even without paying.

6. Higher retention rates – Same as above, but also because its easier to finish and progress and users get positive feedback.
7. Not only the publisher is being rewarded – Users hate feeling like they're being used to make money. Giving them incentives to watch ads removes negative feelings from this activity.
8. Increase revenue – many users don't pay money for mobile apps, and without ads their LTV would be 0. Increasing avg purchase amount for buying users as well, since they can get for "free" small rewards, when the need really arises these users will opt to buy the more expensive bundles.

Cons and Concerns:

1. Poor execution of this feature might give opposite effects. Users might become frustrated with the ads content, location, viewability or even ad-length or rewards, and stop playing. I feel like repeating the opposite affects of the pros might be overstating but I'd like to touch on Decreased revenue. If offered great rewards for watching ads users might opt for watching ads only instead of paying money, which might lead to decreased revenue.

The best way to handle and avoid such risks include 2 steps. First is research and comparisons. Many games are using this monetization feature and its important to compare and take notes of successful implementations. The second includes continuous check-ups using our data and continuous optimization of this feature's implementation.