```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
user_data=pd.read_excel('/content/drive/MyDrive/casestudy/user_gameplay_data.xlsx')
deposit_data=pd.read_excel('/content/drive/MyDrive/casestudy/Deposite.xlsx')
withdrawal_data=pd.read_excel('/content/drive/MyDrive/casestudy/withdrawal.xlsx')
```

```
user_data.head()
```

	User ID	Games Played	Datetime
0	851	1	2022-10-01
1	717	1	2022-10-01
2	456	1	2022-10-01
3	424	1	2022-10-01
4	845	1	2022-10-01

```
deposit_data.head()
```

	User Id	Datetime	Amount
0	357	2022-10-01 00:03:00	2000
1	776	2022-10-01 00:03:00	2500
2	492	2022-10-01 00:06:00	5000
3	803	2022-10-01 00:07:00	5000
4	875	2022-10-01 00:09:00	1500

```
withdrawal_data.head()
```

	User Id	Datetime	Amount
0	190	2022-10-01 00:03:00	5872
1	159	2022-10-01 00:16:00	9540
2	164	2022-10-01 00:24:00	815
3	946	2022-10-01 00:29:00	23000
4	763	2022-10-01 00:40:00	9473

```
user_data.shape
```

```
(355266, 3)
```

```
user_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 355266 entries, 0 to 355265
Data columns (total 3 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   User ID       355266 non-null  int64         
 1   Games Played  355266 non-null  int64         
 2   Datetime      355266 non-null  datetime64[ns]
dtypes: datetime64[ns](1), int64(2)
memory usage: 8.1 MB
```

```
user_data.isna().sum()
```

```
User ID         0
Games Played    0
Datetime        0
dtype: int64
```

```
deposit_data.shape
```

```
(17438, 3)
```

```
deposit_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17438 entries, 0 to 17437
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   User Id   17438 non-null  int64         
 1   Datetime  17438 non-null  datetime64[ns]
 2   Amount    17438 non-null  int64         
dtypes: datetime64[ns](1), int64(2)
memory usage: 408.8 KB
```

```
deposit_data.isna().sum()
```

```
User Id     0
Datetime    0
Amount      0
dtype: int64
```

```
withdrawal_data.shape
```

```
(3566, 3)
```

```
withdrawal_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3566 entries, 0 to 3565
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   User Id   3566 non-null   int64
 1   Datetime  3566 non-null   datetime64[ns]
 2   Amount    3566 non-null   int64
dtypes: datetime64[ns](1), int64(2)
memory usage: 83.7 KB
```

```
withdrawal_data.isna().sum()
```

```
User Id     0
Datetime    0
Amount      0
dtype: int64
```

```
user_data.rename({'User ID': 'user_id','Datetime':'game_play_datetime','Games Played': 'ga
deposit_data.rename({'User Id':'user_id','Datetime':'deposite_datetime','Amount':'deposite
withdrawal_data.rename({'User Id':'user_id','Datetime':'withdrawal_datetime','Amount':'wit
```

```
user_data=user_data.sort_values(by='user_id',ascending=True)
deposit_data=deposit_data.sort_values(by='user_id',ascending=True)
withdrawal_data=withdrawal_data.sort_values(by='user_id',ascending=True)
```

```
total_users=pd.DataFrame(user_data.groupby('user_id')['games_played'].sum()).reset_index()
total_deposite=pd.DataFrame(deposit_data.groupby('user_id')['deposite_amount'].sum()).rese
total_withdrawal=pd.DataFrame(withdrawal_data.groupby('user_id')['withdrawal_amount'].sum(
```

```
a=pd.merge(total_users,total_deposite,on='user_id')
```

```
df=pd.merge(a,total_withdrawal,on='user_id')
```

```
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount
0	2	97	567000	1270215
1	5	391	74100	32700
2	9	3416	193684	171456
3	11	769	46300	101500
4	12	189	99403	20286

```
#this expression for calculating total loyalty point but we want to calculated in slots th
df['Loyalty_point']=(0.01*df['deposite_amount'])+(0.005*df['withdrawal_amount'])+(0.001*ab
```

```
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	97	567000	1270215	12743.690
1	5	391	74100	32700	1024.100
2	9	3416	193684	171456	3499.548
3	11	769	46300	101500	1179.500
4	12	189	99403	20286	1212.377

```
user_data
```

	user_id	games_played	game_play_datetime
57218	0	1	2022-10-05 23:03:00
315150	0	1	2022-10-28 11:44:00
227335	0	1	2022-10-20 19:58:00
121899	0	1	2022-10-11 14:50:00
257821	0	1	2022-10-23 11:41:00
...	...	...	...
312554	999	1	2022-10-28 06:22:00
320566	999	1	2022-10-28 23:15:00
230915	999	1	2022-10-21 03:31:00
304362	999	1	2022-10-27 12:59:00
214538	999	1	2022-10-19 17:03:00

355266 rows × 3 columns

```
from datetime import date,time
user_data['new_date'] = [d.date() for d in user_data['game_play_datetime']]
user_data['new_time'] = [d.time() for d in user_data['game_play_datetime']]
```

```
user_data
```

	user_id	games_played	game_play_datetime	new_date	new_time
57218	0	1	2022-10-05 23:03:00	2022-10-05	23:03:00
315150	0	1	2022-10-28 11:44:00	2022-10-28	11:44:00
227335	0	1	2022-10-20 19:58:00	2022-10-20	19:58:00
121899	0	1	2022-10-11 14:50:00	2022-10-11	14:50:00
257821	0	1	2022-10-23 11:41:00	2022-10-23	11:41:00
...	...	...	...	...	...

```
user_data['Year']=user_data['game_play_datetime'].dt.year
user_data['month']=user_data['game_play_datetime'].dt.month
user_data['day']=user_data['game_play_datetime'].dt.day
user_data['hours']=user_data['game_play_datetime'].dt.hour
user_data['minute']=user_data['game_play_datetime'].dt.minute
user_data['second']=user_data['game_play_datetime'].dt.second
```

calculationg loyalty point for S1 slot

```
slot1_user_data=user_data[user_data['hours']<=12]
```

```
slot2_user_data=user_data[user_data['hours']>12]
```

```
slot1_user_data1=pd.DataFrame(slot1_user_data.groupby('user_id')['games_played'].sum()).re
```

```
c=pd.merge(slot1_user_data1,total_deposite,on='user_id')
```

```
d=pd.merge(c,total_withdrawal,on='user_id')
```

```
d
```

	user_id	games_played	deposite_amount	withdrawal_amount
0	2	51	567000	1270215
1	5	226	74100	32700
2	9	1846	193684	171456
3	11	398	46300	101500

```
d['Loyalty_point']=(0.01*d['deposite_amount'])+(0.005*d['withdrawal_amount'])+(0.001*abs(d
```

```
d
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	51	567000	1270215	12734.490
1	5	226	74100	32700	991.100
2	9	1846	193684	171456	3185.548
3	11	398	46300	101500	1105.300
4	12	108	99403	20286	1196.177
...	...	...	...	...	...
472	980	33	33986	87900	839.874
473	985	483	186500	155000	2768.100
474	987	1198	69900	135650	1682.600
475	989	1887	240000	1339000	10571.400
476	992	1341	804978	616278	11588.070

477 rows × 5 columns

calculating loyalty point for slot2

```
slot2_user_data1=pd.DataFrame(slot2_user_data.groupby('user_id')['games_played'].sum()).re
```

```
e=pd.merge(slot2_user_data,total_deposite,on='user_id')
```

```
f=pd.merge(e,total_withdrawal,on='user_id')
```

```
f['Loyalty_point']=(0.01*f['deposite_amount'])+(0.005*f['withdrawal_amount'])+(0.001*abs(f
```

```
f.head()
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month	day	l
0	2	1	2022-10-31 16:32:00	2022-10-31	16:32:00	2022	10	31	
1	2	1	2022-10-21 16:12:00	2022-10-21	16:12:00	2022	10	21	
2	2	1	2022-10-17 16:27:00	2022-10-17	16:27:00	2022	10	17	
3	2	1	2022-10-14 18:16:00	2022-10-14	18:16:00	2022	10	14	
4	2	1	2022-10-15 15:10:00	2022-10-15	15:10:00	2022	10	15	

"Based on the above information and the data provided answer the following questions:

1. Find Playerwise Loyalty points earned by Players in the following slots:-

a. 2nd October Slot S1

```
slot1_user_data.head()
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month	d
315150	0	1	2022-10-28 11:44:00	2022-10-28	11:44:00	2022	10	
257821	0	1	2022-10-23 11:41:00	2022-10-23	11:41:00	2022	10	
322568	0	1	2022-10-29 03:27:00	2022-10-29	03:27:00	2022	10	

```
October_Slot_S1=slot1_user_data.loc[slot1_user_data['month']==10]
```

```
October_2nd_Slot_S1 = October_Slot_S1[October_Slot_S1['day']==2]
```

```
October_2nd_Slot_S1.head(1)
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month	d
14299	2	1	2022-10-02 06:04:00	2022-10-	06:04:00	2022	10	

```
October_2nd_Slot_S1=pd.DataFrame(October_2nd_Slot_S1.groupby('user_id')['games_played'].su
```

```
October_2nd_Slot_S1.head(2)
```

	user_id	games_played
0	2	2
1	5	12

```
g=pd.merge(October_2nd_Slot_S1,total_deposite,on='user_id')

df2=pd.merge(g,total_withdrawal,on='user_id')

df2['Loyalty_point']=(0.01*df2['deposite_amount'])+(0.005*df2['withdrawal_amount'])+(0.001

df2.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	2	567000	1270215	12724.690
1	5	12	74100	32700	948.300
2	9	56	193684	171456	2827.548
3	11	15	46300	101500	1028.700
4	12	2	99403	20286	1174.977

18th October Slot S1

```
October_18th_Slot_S1 = October_Slot_S1[October_Slot_S1['day']==18]

October_18th_Slot_S1=pd.DataFrame(October_18th_Slot_S1.groupby('user_id')['games_played'].

October_18th_Slot_S1.head(2)
```

	user_id	games_played
0	2	2
1	3	2

```
h=pd.merge(October_18th_Slot_S1,total_deposite,on='user_id')

df3=pd.merge(h,total_withdrawal,on='user_id')

df3['Loyalty_point']=(0.01*df3['deposite_amount'])+(0.005*df3['withdrawal_amount'])+(0.001
```

```
df3.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	2	567000	1270215	12724.690
1	5	10	74100	32700	947.900
2	9	52	193684	171456	2826.748
3	11	13	46300	101500	1028.300
4	12	5	99403	20286	1175.577

16th October Slot S2

```
slot2_user_data.head()
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month
57218	0	1	2022-10-05 23:03:00	2022-10-05	23:03:00	2022	10
227335	0	1	2022-10-20 19:58:00	2022-10-20	19:58:00	2022	10
121899	0	1	2022-10-11 14:50:00	2022-10-11	14:50:00	2022	10

```
October_Slot_S2=slot2_user_data.loc[slot2_user_data['month']==10]
```

```
October_16th_Slot_S2 = October_Slot_S2[October_Slot_S2['day']==16]
```

```
October_16th_Slot_S2.head(2)
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month
180121	2	1	2022-10-16 17:18:00	2022-10-16	17:18:00	2022	10

```
October_16th_Slot_S2=pd.DataFrame(October_16th_Slot_S2.groupby('user_id')['games_played'].
```

```
I=pd.merge(October_16th_Slot_S2,total_deposite,on='user_id')
```

```
df4=pd.merge(I,total_withdrawal,on='user_id')
```

```
df4['Loyalty_point']=(0.01*df4['deposite_amount'])+(0.005*df4['withdrawal_amount'])+(0.001
```

```
df4.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	2	567000	1270215	12724.690
1	5	9	74100	32700	947.700
2	9	43	193684	171456	2824.948
3	11	13	46300	101500	1028.300
4	12	3	99403	20286	1175.177

26th October Slot S2

```
October_26th_Slot_S2 = October_Slot_S2[October_Slot_S2['day']==26]
```

```
October_26th_Slot_S2.head(2)
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month
295353	5	1	2022-10-26 18:12:00	2022-10-26	18:12:00	2022	10

```
October_26th_Slot_S2=pd.DataFrame(October_26th_Slot_S2.groupby('user_id')['games_played'].
```

```
J=pd.merge(October_26th_Slot_S2,total_deposite,on='user_id')
```

```
df5=pd.merge(J,total_withdrawal,on='user_id')
```

```
df5['Loyalty_point']=(0.01*df5['deposite_amount'])+(0.005*df5['withdrawal_amount'])+(0.001
```

```
df5.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	5	4	74100	32700	946.700
1	9	42	193684	171456	2824.748
2	11	11	46300	101500	1027.900
3	12	4	99403	20286	1175.377
4	16	34	360201	418387	5758.931

2. Calculate overall loyalty points earned and rank players on the basis of loyalty points in the month of October. In case of tie, number of games played should be taken as the next

criteria for ranking.

```
user_data.head()
```

	user_id	games_played	game_play_datetime	new_date	new_time	Year	month
57218	0	1	2022-10-05 23:03:00	2022-10-05	23:03:00	2022	10
315150	0	1	2022-10-28 11:44:00	2022-10-28	11:44:00	2022	10
227335	0	1	2022-10-20 19:58:00	2022-10-20	19:58:00	2022	10

```
user_data['month'].unique()
```

```
array([10])
```

```
#overall loyalty point we calculated above
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point
0	2	97	567000	1270215	12743.690
1	5	391	74100	32700	1024.100
2	9	3416	193684	171456	3499.548
3	11	769	46300	101500	1179.500
4	12	189	99403	20286	1212.377

```
df['Rank_loyaltypoint'] = df['Loyalty_point'].rank(ascending=False)
```

```
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point	Rank_lo
0	2	97	567000	1270215	12743.690	
1	5	391	74100	32700	1024.100	
2	9	3416	193684	171456	3499.548	
3	11	769	46300	101500	1179.500	
4	12	189	99403	20286	1212.377	

```
df['Rank_loyaltypoint'].duplicated().sum()
```

```
0
```

```
# no duplicated value in rank means there is no tie
```

3. What is the average deposit amount?
4. What is the average deposit amount per user in a month?
5. What is the average number of games played per user?"

```
#What is the average deposit amount?
deposit_data['deposite_amount'].mean()
```

```
5492.185399701801
```

```
#What is the average deposit amount per user in a month?
avg_deposite=pd.DataFrame(deposit_data.groupby('user_id')['deposite_amount'].mean()).reset
avg_deposite.head()
```

	user_id	deposite_amount
0	1	5000.000000
1	2	28350.000000
2	3	10000.000000
3	4	1750.000000
4	5	1105.970149

```
#What is the average number of games played per user?"
avg_game_users=pd.DataFrame(user_data.groupby('user_id')['games_played'].mean()).reset_ind
avg_game_users.head()
```

	user_id	games_played
0	0	1.0
1	1	1.0
2	2	1.0
3	3	1.0
4	4	1.0

"Part B - How much bonus should be allocated to leaderboard players?

After calculating the loyalty points for the whole month find out which 50 players are at the top of the leaderboard. The company has allocated a pool of Rs 50000 to be given away as bonus money to the loyal players.

Now the company needs to determine how much bonus money should be given to the players.

Should they base it on the amount of loyalty points? Should it be based on number of games? Or something else?

That's for you to figure out.

Suggest a suitable way to divide the allocated money keeping in mind the following points:

1. Only top 50 ranked players are awarded bonus

```
df.head(2)
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point	Rank_lo
0	2	97	567000	1270215	12743.69	
1	5	391	74100	32700	1024.10	

```
df=df.sort_values(by='Rank_loyaltypoint').head(50).copy()
```

```
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point	Rank_
315	634	24	515000	15737705	99066.030	
44	99	10	1164800	2403141	24904.046	
335	672	10	2158700	233750	24682.700	
101	212	1	1924981	589850	23534.391	
283	566	183	1819175	185071	20787.809	

I prefer the bonus amount should be given on the basis of Loyalty_point because when we observed in data some player played more no of games but there loyalty point are low

```
mul_factor=50000/sum(df['Loyalty_point'])
```

```
df['bonus_amount']=df['Loyalty_point']*mul_factor
```

```
df.head()
```

	user_id	games_played	deposite_amount	withdrawal_amount	Loyalty_point	Rank_
315	634	24	515000	15737705	99066.030	

Would you say the loyalty point formula is fair or unfair?

Can you suggest any way to make the loyalty point formula more robust?"

101	212	1	1924961	589650	23534.391	

----yes its a fair formula,but we closely observe the data that some of the player which are playing more number of games but there loyalty point are low may be because lower amount of deposite and withdrawal so we can increase the multiplication factor in no of games played (more than 0.2) that will also increase the visibility of that players in terms of loyalty point