Loyalty Points Analysis for Online Game Players

In [1]:
```
import pandas as pd
```

Load and Clean the Data

In [3]:
```
game_df = pd.read_csv('User_Gameplay_Data.csv')
game_df.head()
```

Out[3]:

	User ID	Games Played	Datetime
0	851	1	01/10/2022 00:00
1	717	1	01/10/2022 00:00
2	456	1	01/10/2022 00:00
3	424	1	01/10/2022 00:00
4	845	1	01/10/2022 00:00

In [4]:
```
game_df['User Id'] = game_df['User ID'].astype(int)
game_df['Games Played'] = game_df['Games Played'].astype(int)
game_df['Datetime'] = pd.to_datetime(game_df['Datetime'], dayfirst=True, errors=
```

In [5]:
```
deposit_df = pd.read_csv('Deposit_Data.csv')
deposit_df.head()
```

Out[5]:

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

In [6]:
```
deposit_df['User Id'] = deposit_df['User Id'].astype(int)
deposit_df['Amount'] = deposit_df['Amount'].astype(float)
deposit_df['Datetime'] = pd.to_datetime(deposit_df['Datetime'], dayfirst=True, e
```

In [7]:
```
withdraw_df = pd.read_csv('Withdrawal_Data.csv')
withdraw_df.head()
```

Out[7]:

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

In [8]:
```
withdraw_df['User Id'] = withdraw_df['User Id'].astype(int)
withdraw_df['Amount'] = withdraw_df['Amount'].astype(float)
withdraw_df['Datetime'] = pd.to_datetime(withdraw_df['Datetime'], dayfirst=True,
```

Part A - Calculating loyalty points

On each day, there are 2 slots for each of which the loyalty points are to be calculated:

- S1 from 12am to 12pm
- S2 from 12pm to 12am

In [10]:
```
def get_slot(hour):
    return 'S1' if hour < 12 else 'S2'

for df in [deposit_df, withdraw_df, game_df]:
    df['Date'] = df['Datetime'].dt.date
    df['Slot'] = df['Datetime'].dt.hour.apply(get_slot)
```

Loyality Point Calculation

Basically Loyalty Point depends on 4 things:

- How much money they deposited
- How much they withdrew
- How many times they deposited and withdrew
- How many games they played

This function summarizes each user's behavior (money in/out and games played), applies the loyalty point formula, and gives you a final table with loyalty scores for each user.

In [12]:
```
def compute_loyalty_points(dep, wd, gp):

    # Unique Users from Deposit Data
    merged = pd.DataFrame({'User Id': dep['User Id'].unique()})

    # ------------------------------------------------------------------------
    # Summarize per User
    # 1. sum = total amount deposited
    # 2. count = how many times they deposited
    dep_group = dep.groupby('User Id')['Amount'].agg(['sum', 'count']).reset_ind

    # Summarize Withdrawal Data
    # 1. sum = total withdrawal amount
```

```
    # 2. count = number of withdrawals
    wd_group = wd.groupby('User Id')['Amount'].agg(['sum', 'count']).reset_index

    # Summarize Game Data [Total number of games played per user]
    gp_group = gp.groupby('User Id')['Games Played'].sum().reset_index()
    # ------------------------------------------------------------------------

    # Merge All Info Together
    merged = merged.merge(dep_group, on='User Id', how='left').rename(columns={'
    merged = merged.merge(wd_group, on='User Id', how='left').rename(columns={'s
    merged = merged.merge(gp_group, on='User Id', how='left')

    # Handle Missing Data
    merged.fillna(0, inplace=True)

    # Loyalty Formula
    merged['Loyalty Points'] = (
        0.01 * merged['Deposit'] +
        0.005 * merged['Withdrawal'] +
        0.001 * (merged['Deposit Count'] - merged['Withdrawal Count']).clip(lowe
        0.2 * merged['Games Played']
    )
    return merged
```

Answer Each Sub-question

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

- a. 2nd October Slot S1
- b. 16th October Slot S2
- c. 18th October Slot S1
- d. 26th October Slot S2

In [14]:
```
date_check = pd.to_datetime("2022-10-02").date()
slot_check = "S1"

filtered_dep = deposit_df[(deposit_df['Date'] == date_check) & (deposit_df['Slot
filtered_wd = withdraw_df[(withdraw_df['Date'] == date_check) & (withdraw_df['Sl
filtered_gp = game_df[(game_df['Date'] == date_check) & (game_df['Slot'] == slot

points = compute_loyalty_points(filtered_dep, filtered_wd, filtered_gp)
print('2nd October Slot S1')
print(points[['User Id', 'Loyalty Points']])
```

```
2nd October Slot S1
     User Id  Loyalty Points
0        695          22.254
1        214          77.902
2        945          53.001
3        779          74.202
4        390          24.003
..       ...             ...
212       72           5.601
213      681         100.001
214      699          55.801
215       45           5.601
216      252           0.901

[217 rows x 2 columns]
```

In [15]:
```
date_check = pd.to_datetime("2022-10-16").date()
slot_check = "S2"

filtered_dep = deposit_df[(deposit_df['Date'] == date_check) & (deposit_df['Slot
filtered_wd = withdraw_df[(withdraw_df['Date'] == date_check) & (withdraw_df['Sl
filtered_gp = game_df[(game_df['Date'] == date_check) & (game_df['Slot'] == slot

points = compute_loyalty_points(filtered_dep, filtered_wd, filtered_gp)
print('16th October Slot S2')
print(points[['User Id', 'Loyalty Points']])
```

```
16th October Slot S2
     User Id  Loyalty Points
0        342          65.804
1        114          50.001
2        565          27.380
3        733          11.201
4        695          30.404
..       ...             ...
184      806          50.601
185       47          14.110
186      598          30.601
187      212         999.991
188      439          98.801

[189 rows x 2 columns]
```

In [16]:
```
date_check = pd.to_datetime("2022-10-18").date()
slot_check = "S1"

filtered_dep = deposit_df[(deposit_df['Date'] == date_check) & (deposit_df['Slot
filtered_wd = withdraw_df[(withdraw_df['Date'] == date_check) & (withdraw_df['Sl
filtered_gp = game_df[(game_df['Date'] == date_check) & (game_df['Slot'] == slot

points = compute_loyalty_points(filtered_dep, filtered_wd, filtered_gp)
print('18th October Slot S1')
print(points[['User Id', 'Loyalty Points']])
```

```
18th October Slot S1
     User Id  Loyalty Points
0        180          20.201
1        204          22.001
2        749          40.201
3        613          60.005
4        536           3.002
..       ...             ...
189      699          66.601
190      219          20.001
191      252           1.001
192      266           4.201
193      597          11.601

[194 rows x 2 columns]
```

In [17]:
```
date_check = pd.to_datetime("2022-10-26").date()
slot_check = "S2"

filtered_dep = deposit_df[(deposit_df['Date'] == date_check) & (deposit_df['Slot
filtered_wd = withdraw_df[(withdraw_df['Date'] == date_check) & (withdraw_df['Sl
filtered_gp = game_df[(game_df['Date'] == date_check) & (game_df['Slot'] == slot
```

```
points = compute_loyalty_points(filtered_dep, filtered_wd, filtered_gp)
print('26th October Slot S2')
print(points[['User Id', 'Loyalty Points']])
```

```
26th October Slot S2
     User Id  Loyalty Points
0        422         150.001
1        516          73.202
2        875           8.401
3        553          10.502
4        335          21.201
..       ...             ...
183      588          50.601
184      369        1501.915
185      773           2.501
186      515         351.201
187      162         174.601

[188 rows x 2 columns]
```

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.

In [20]:
```
# Filter October data from all datasets
oct_deposit = deposit_df[deposit_df['Datetime'].dt.month == 10]
oct_withdrawal = withdraw_df[withdraw_df['Datetime'].dt.month == 10]
oct_gameplay = game_df[game_df['Datetime'].dt.month == 10]

# Compute loyalty points using your defined function
oct_loyalty = compute_loyalty_points(oct_deposit, oct_withdrawal, oct_gameplay)

# Rank users by Loyalty Points (descending), and by Games Played in case of tie
oct_loyalty = oct_loyalty.sort_values(by=['Loyalty Points', 'Games Played'], asc
oct_loyalty['Rank'] = range(1, len(oct_loyalty) + 1)

# View top 10 for confirmation
oct_loyalty.head(10)
```

Out[20]:

	User Id	Deposit	Deposit Count	Withdrawal	Withdrawal Count	Games Played	Loyalty Points	Rank
608	634	515000.0	8	15737705.0	67.0	24	83843.325	1
315	99	1164800.0	47	2403141.0	15.0	10	23665.737	2
371	672	2158700.0	35	233750.0	5.0	10	22757.780	3
373	212	1924981.0	26	589850.0	4.0	1	22199.282	4
153	740	1738490.0	91	365288.0	7.0	2	19211.824	5
203	566	1819175.0	53	185071.0	3.0	183	19153.755	6
93	714	1676300.0	34	0.0	0.0	6	16764.234	7
34	421	878600.0	99	1269809.0	84.0	1557	15446.460	8
542	369	650000.0	13	1586208.0	9.0	37	14438.444	9
8	30	1329000.0	51	152145.0	1.0	13	14053.375	10

3. What is the average deposit amount?

In [22]:
```
avg_deposit_amt = deposit_df['Amount'].mean()
print("Average deposit amount across all users and time:", round(avg_deposit_amt
```

```
Average deposit amount across all users and time: 5492.19
```

4. What is the average deposit amount per user in a month?

In [25]:
```
# Total deposit per user in October
oct_deposit_per_user = oct_deposit.groupby('User Id')['Amount'].sum()

# Average across users
avg_deposit_per_user_oct = oct_deposit_per_user.mean()
print("Average deposit amount per user in October:", round(avg_deposit_per_user_
```

```
Average deposit amount per user in October: 104669.65
```

5. What is the average number of games played per user?"

In [27]:
```
# Total games played per user (across all time)
total_games_per_user = game_df.groupby('User Id')['Games Played'].sum()

# Average
avg_games_per_user = total_games_per_user.mean()
print("Average number of games played per user:", round(avg_games_per_user, 2))
```

```
Average number of games played per user: 355.27
```

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

1. Only top 50 ranked players are awarded bonus

In [31]:
```
# Get top 50 players based on loyalty rank (already sorted in Q2)
top_50 = oct_loyalty.head(50).copy()
```

```
# Total loyalty points of top 50
total_loyalty_top_50 = top_50['Loyalty Points'].sum()

# Allocate bonus proportionally
top_50['Bonus ₹'] = (top_50['Loyalty Points'] / total_loyalty_top_50) * 50000

# View final bonus distribution
top_50[['User Id', 'Loyalty Points', 'Rank', 'Bonus ₹']].head(10)
```

Out[31]:

	User Id	Loyalty Points	Rank	Bonus ₹
608	634	83843.325	1	6638.861745
315	99	23665.737	2	1873.894625
371	672	22757.780	3	1802.000995
373	212	22199.282	4	1757.778142
153	740	19211.824	5	1521.225970
203	566	19153.755	6	1516.627965
93	714	16764.234	7	1327.421495
34	421	15446.460	8	1223.077835
542	369	14438.444	9	1143.261357
8	30	14053.375	10	1112.770917

Part C

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

- The existing formula for calculating loyalty points is somewhat dissatisfactory and could be improved. It has the effect of rewarding individuals to play games and deposit money, which promotes activities and expenditures.
- Nevertheless, the system of granting points with withdrawals appears to be anti-productive: it encourages users to withdraw money from the platforms. Moreover, the formula makes reference more to the amount of the deposit than to the player playing the game, which may prove unfair to those users who are very active but do not deposit a lot.
- To ensure that the system is more balanced, it is advised that points for the withdrawals should be lowered, or none at all, weight applied to the games played should be much more, and the points to apply to other faithful actions like logging on daily or referring others can be considered.