ABC COMPANY

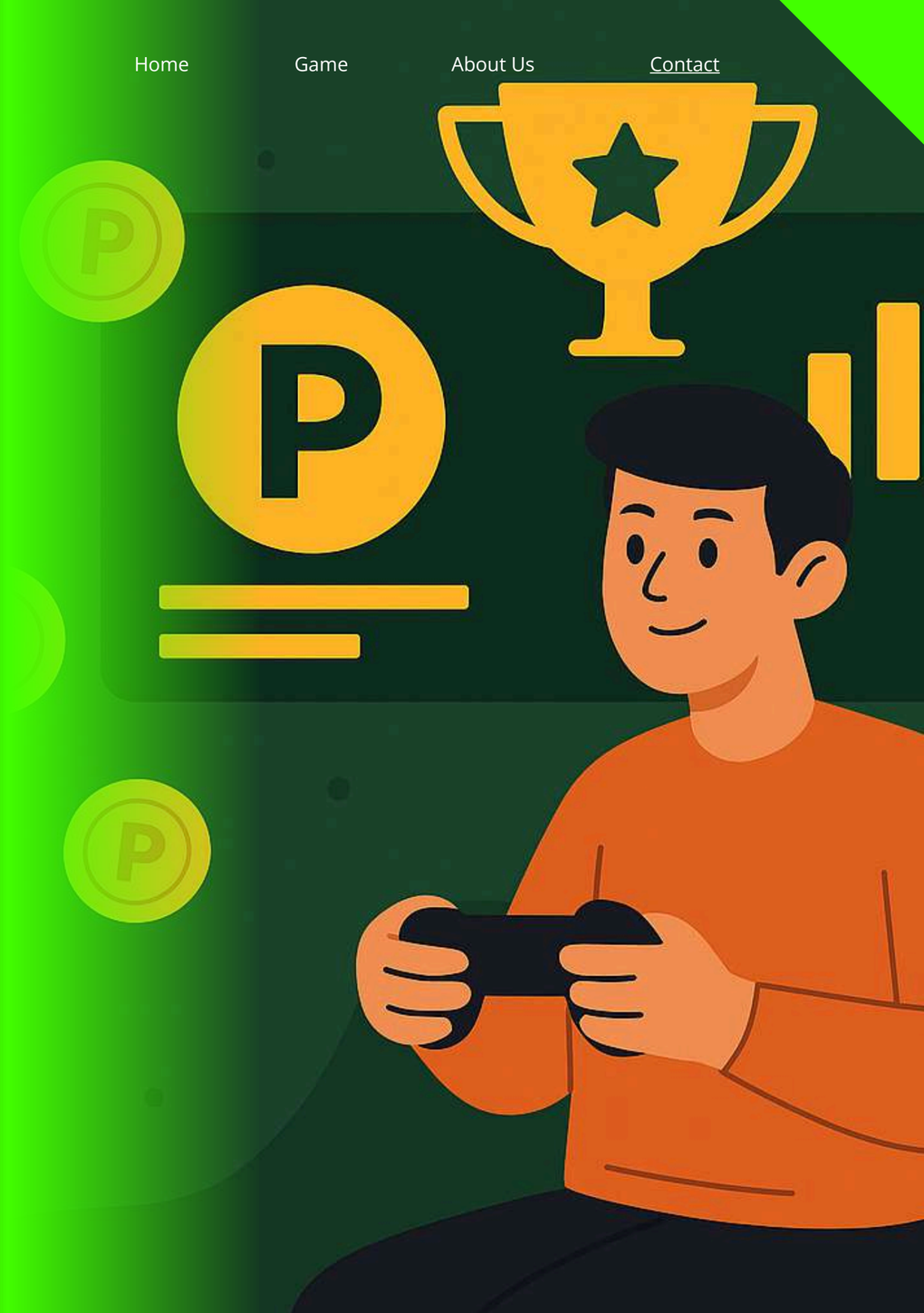
Home Game About Us Contact

🎮 LOYALTY POINTS ANALYSIS FOR A REAL - MONEY GAMING PLATFORM

A DATA-DRIVEN CASE STUDY

PRESENTED BY : PRIYANKA JINKA

DATE : JUNE 2025



PROJECT OVERVIEW

ABC is a real-money online gaming platform offering multiplayer games like Ludo.

🎯 Project Goal:

Analyze user behaviour and transactions to:

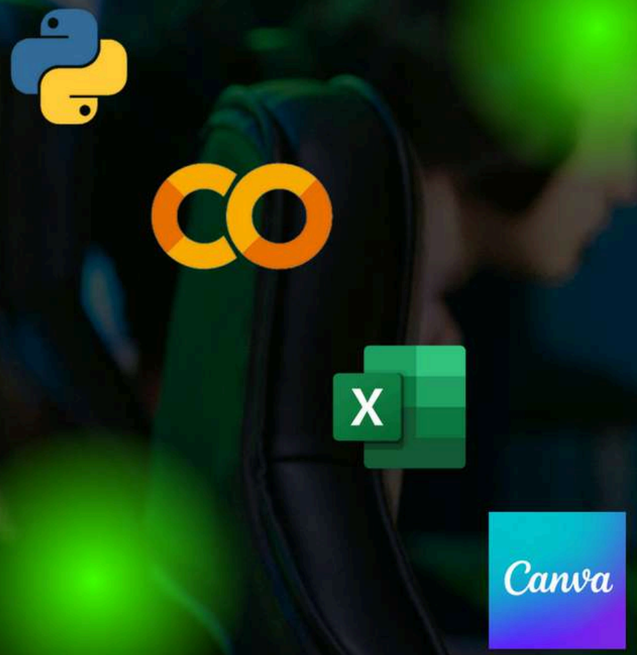
- Calculate player loyalty points.
- Rank top users based on activity.
- Allocate bonus money to the top 50 loyal players.
- Recommend a fair bonus distribution strategy.

LEARN MORE

TOOLS & DATA USED

TOOLS USED

- Python (Pandas, Matplotlib)
- Google Colab
- Excel
- Canva



DATASETS

- User Gameplay
- Deposit Transactions
- Withdrawal Transactions

UNDERSTANDING THE DATA STRUCTURE

🎮 User Gameplay Data

- This data set contains records of games played by users.

User ID	Games Played	Datetime
851	1	01-10-2022 0:00
717	1	01-10-2022 0:00
456	1	01-10-2022 0:00
424	1	01-10-2022 0:00
845	1	01-10-2022 0:00

💰 Deposit Data

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

- This dataset includes user transactions categorized as deposits.

🏧 Withdrawal Data

- This dataset includes transactions where users have withdrawn money from the platform.

User Id	Datetime	Amount
190	01-10-2022 0:03	5872
159	01-10-2022 0:16	9540
164	01-10-2022 0:24	815
946	01-10-2022 0:29	23000

The datasets can be accessed at : GAMING DATASET

LOYALTY POINTS SYSTEM – RULES & FORMULA

ABC is a real-money online gaming platform where users deposit money to play multiplayer games like Ludo and withdraw winnings.

To retain players, ABC awards Loyalty Points based on in-app activity: deposits, withdrawals, and games played.

Each activity has a weight, and top 50 players at month-end receive bonuses based on their loyalty points.

Loyalty Point Calculation Formula:

Type of Action	Weightage per activity	Formulae	eg.
Deposit of money on the platform	0.01	0.01 * Deposit Amount	0.01 * (1000 RS Deposit) = 10 Points
Withdrawal of money from the platform	0.005	0.005 * Withdrawal Amount	0.005 * (500 Rs Withdrawal) = 2.5 Points
How many more times did a player do deposit than withdrawal	0.001	0.001 * maximum of (#deposit - #windrawal) or 0	0.001 * max((5-3, 0)) = 0.001 * 2 = 0.002 points where number of deposit = 5 and number of withdrawal = 3
Number of games played	0.2	0.2 *Number of Games Played	0.2 * (50 Total Games Played) = 10 Points

Loyalty Points =
0.01 × Deposits + 0.005 × Withdrawals + 0.001 × Extra Deposits + 0.2 × Games Played

? CASE STUDY QUESTIONS



- Calculate loyalty points for selected time slots
- Rank users by loyalty score
- Calculate average deposit per user
- Calculate average games played per user
- Design a fair bonus strategy
- Suggest improvements to the loyalty formula

METHODOLOGY

Load & Rename Columns

- Imported the 3 Excel sheets (Gameplay, Deposit, and Withdrawal)
- Skipped unnecessary header rows

```
import pandas as pd

data_file = "Gaming Dataset Case study.xlsx"

gameplay_df = pd.read_excel(data_file,sheet_name= "User Gameplay data",skiprows=3)
deposit_df = pd.read_excel(data_file,sheet_name="Deposit Data",skiprows=3)
withdrawal_df = pd.read_excel(data_file,sheet_name="Withdrawal Data",skiprows=3)
```

- Renamed columns for consistency and clarity.

	User_Id	Games_Played	Datetime
0	851	1	2022-01-10 00:00:00
1	717	1	2022-01-10 00:00:00
2	456	1	2022-01-10 00:00:00
3	424	1	2022-01-10 00:00:00
4	845	1	2022-01-10 00:00:00

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

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

```
gameplay_df.columns=['User_Id','Games_Played','Datetime']
deposit_df.columns = ['User_Id','Datetime', 'Deposit_Amount']
withdrawal_df.columns = ['User_Id', 'Datetime', 'Withdraw_Amount']
```

METHODOLOGY

Convert to datetime

- Datetime column is in object,convert it into Datetime datatype for further date filters.

```
gameplay_df['Datetime'] = pd.to_datetime(gameplay_df['Datetime'])
deposit_df['Datetime']= pd.to_datetime(deposit_df['Datetime'])
withdrawal_df['Datetime'] = pd.to_datetime(withdrawal_df['Datetime'])
```

	User_Id	Games_Played	Datetime
0	851	1	2022-01-10 00:00:00
1	717	1	2022-01-10 00:00:00
2	456	1	2022-01-10 00:00:00
3	424	1	2022-01-10 00:00:00
4	845	1	2022-01-10 00:00:00

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

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

⚙ METHODOLOGY

Part A **01**

🎯 Loyalty Points by Time Slots

- <u>Calculating loyalty points:</u>
- 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

- Calculate loyalty points per player in the following time slots:

🕛 2nd Oct – Slot 1 (12:00 AM – 12:00 PM)
🕛 16th Oct – Slot 2 (12:00 PM – 11:59 PM)
🕛 18th Oct – Slot 1 (12:00 AM – 12:00 PM)
🕛 26th Oct – Slot 2 (12:00 PM – 11:59 PM)

- *There are no gameplay activity, deposit & withdrawal transactions on 2nd October.*
- *Hence, loyalty point computation for this specific slot is not possible.*

```
# Filter for 16th Oct Slot 2

slot_start = pd.to_datetime("2022-10-16 12:00:00")
slot_end = pd.to_datetime("2022-10-16 23:59:00")

slot2_gameplay = gameplay_df[(gameplay_df['GameDate'] ⩾ slot_start) &
                             (gameplay_df['GameDate'] ⩽ slot_end)]
```

- Repeat the same for withdrawal_df and deposit_df.

METHODOLOGY

Aggregating & Merging Player Data

Objective:

Aggregate player-level data for:

- Total Deposit Amount
- Total Withdrawal Amount
- Count of Deposits & Withdrawals
- Number of Games Played

```
withdrawal_summary = slot2_withdrawals.groupby('User_Id').agg(
    total_withdrawal_amount=('Withdraw_Amount', 'sum'),
    num_withdrawals=('Withdraw_Amount', 'count')).reset_index()

withdrawal_summary.head()
```

- Use .groupby() to aggregate
- .merge() to combine data across sources.

```
deposit_summary = slot2_deposits.groupby('User_Id').agg(
    total_deposit_amount=('Deposit_Amount', 'sum'),
    num_deposits=('Deposit_Amount', 'count'))
deposit_summary.head()
```

```
gameplay_summary = slot2_gameplay.groupby('User_Id')['Games_Played'].count().reset_index()
gameplay_summary.head()
```

```
# Merge All Summaries

merged = pd.merge(deposit_summary, withdrawal_summary, on='User_Id', how='outer')
merged_df = pd.merge(merged, gameplay_summary, on='User_Id', how='outer')
merged_df.head()
```

METHODOLOGY

Calculating Loyalty points

- Replace NaN values with 0 to avoid errors in math calculations

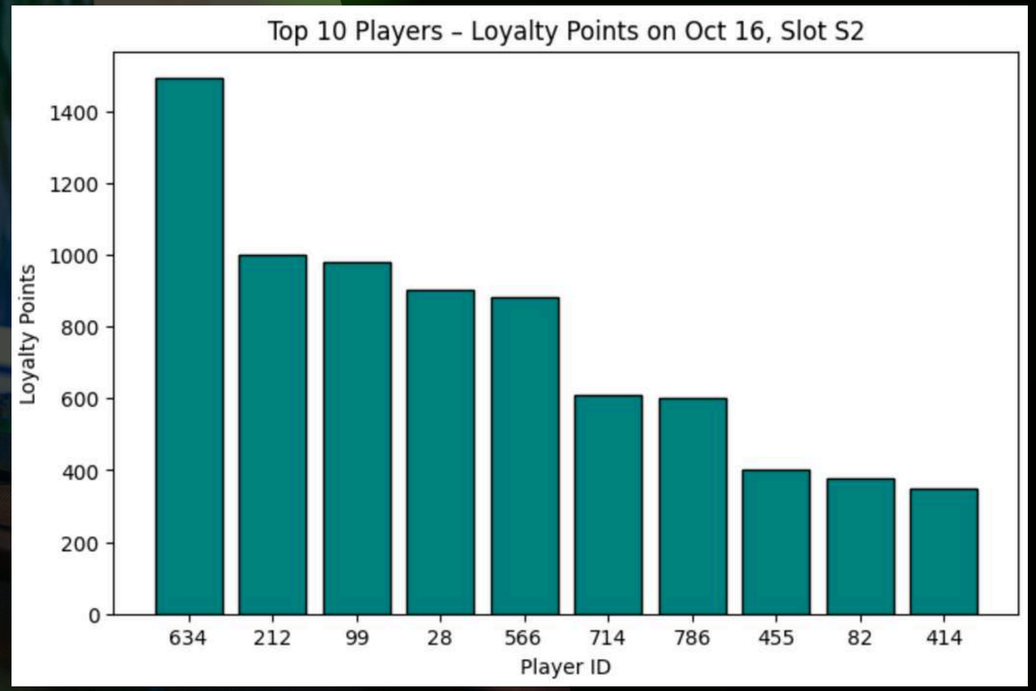
```
merged_df.fillna(0, inplace=True)
```

- Use the formula to calculate Loyalty points.

```
merged_df['extra_deposit_actions'] = (merged_df['num_deposits'] - merged_df['num_withdrawals']).clip(lower=0)

merged_df['Loyalty_Points'] = (
    0.01 * merged_df['total_deposit_amount'] +
    0.005 * merged_df['total_withdrawal_amount'] +
    0.001 * merged_df['extra_deposit_actions'] +
    0.2 * merged_df['Games_Played']
)
```

- Repeat the same for other dates and time slots.

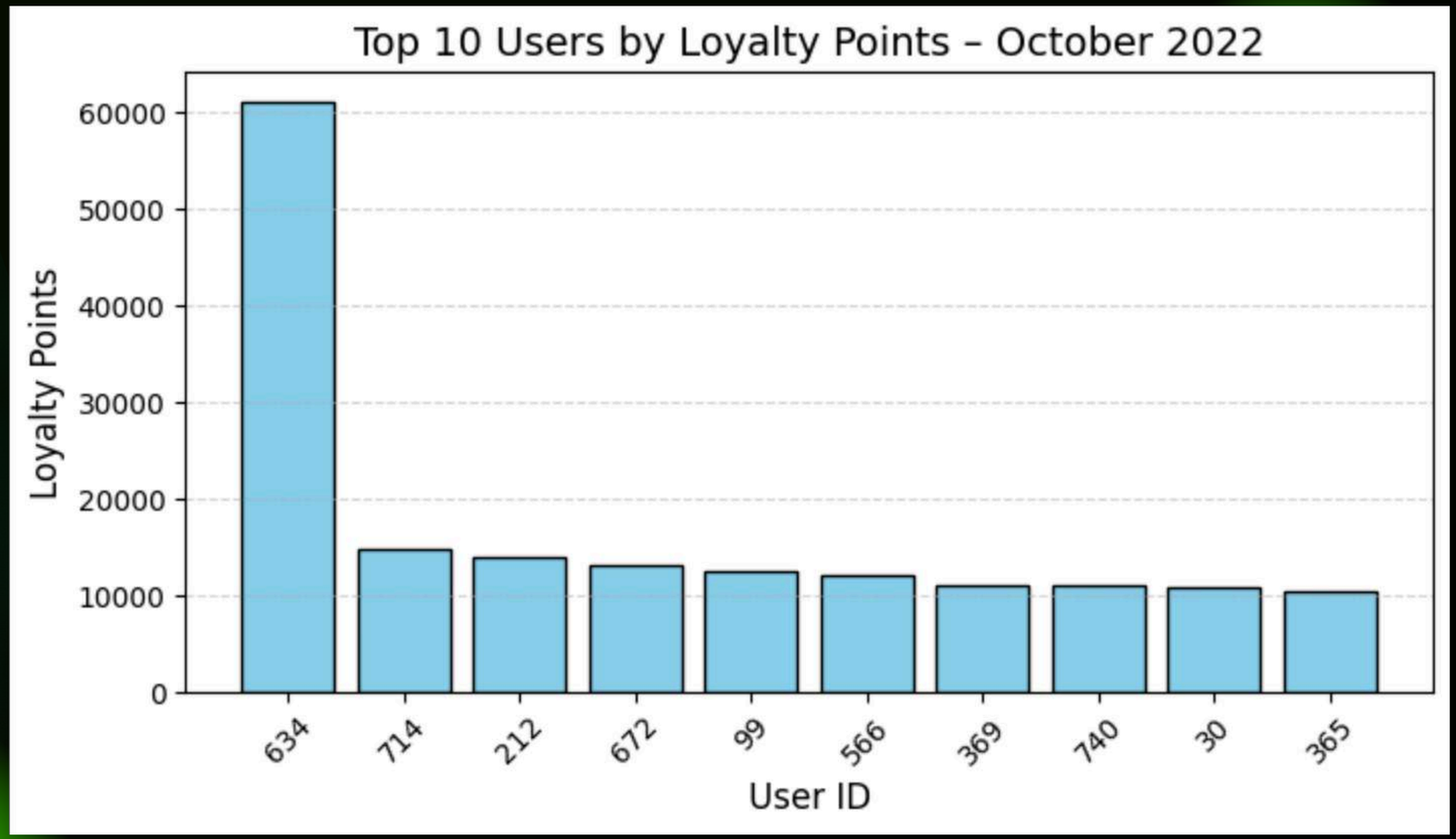


METHODOLOGY

02 Player Ranking Based on Loyalty Points – October

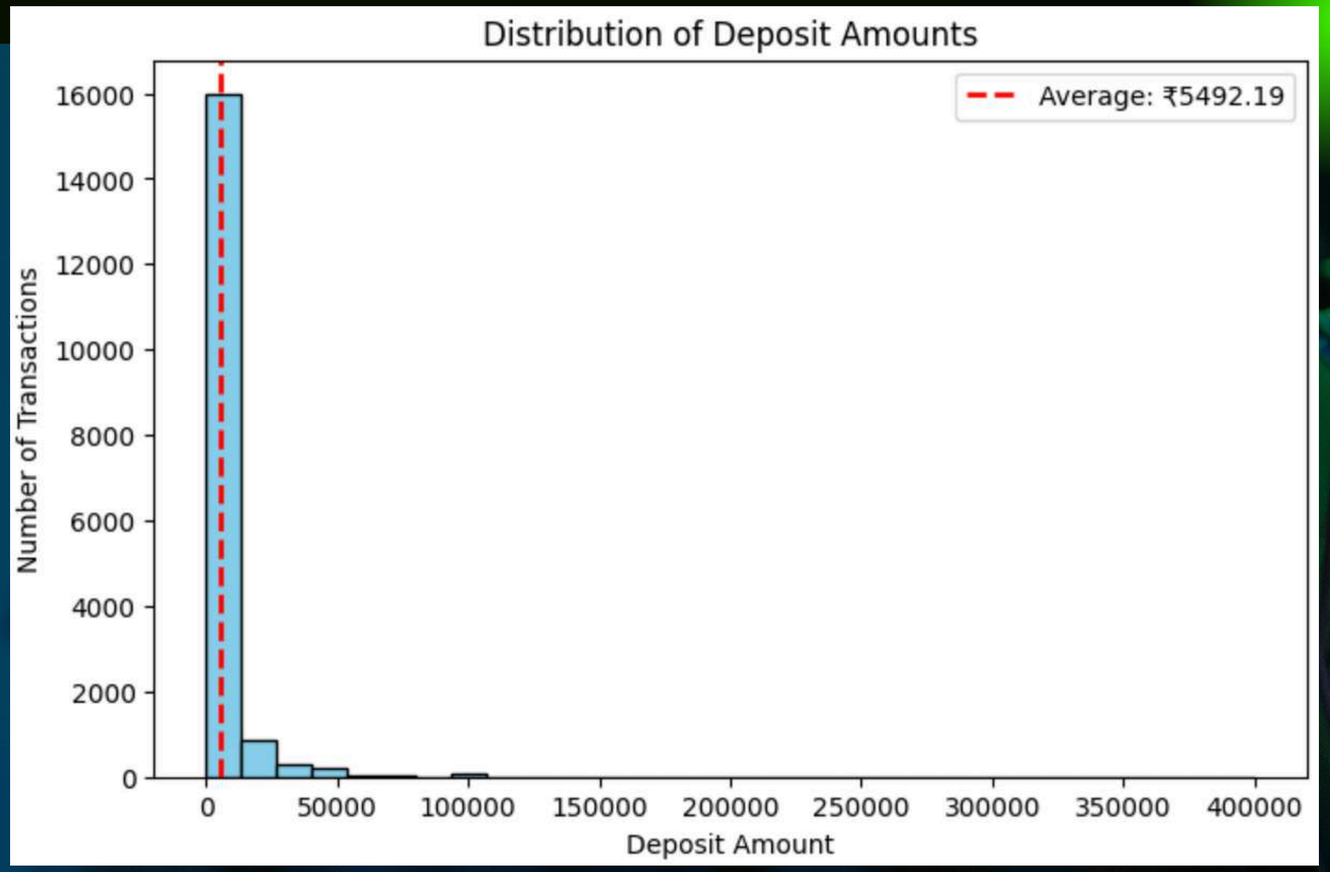
Repeat the same steps used earlier —

✅ Filter the data for the month of October (no specific date).
✅ Aggregate deposit, withdrawal, and gameplay data by PlayerID.
✅ Apply the loyalty formula.
✅ Rank players using loyalty points as primary and games played as tie-breaker.



METHODOLOGY

03 Average Deposit Amount Across All Transactions




```
average_deposit = deposit_df['Deposit_Amount'].mean()

print(f"Average Deposit Amount: ₹{average_deposit:.2f}")
```

The average deposit amount across all transactions is **₹5492.19**.

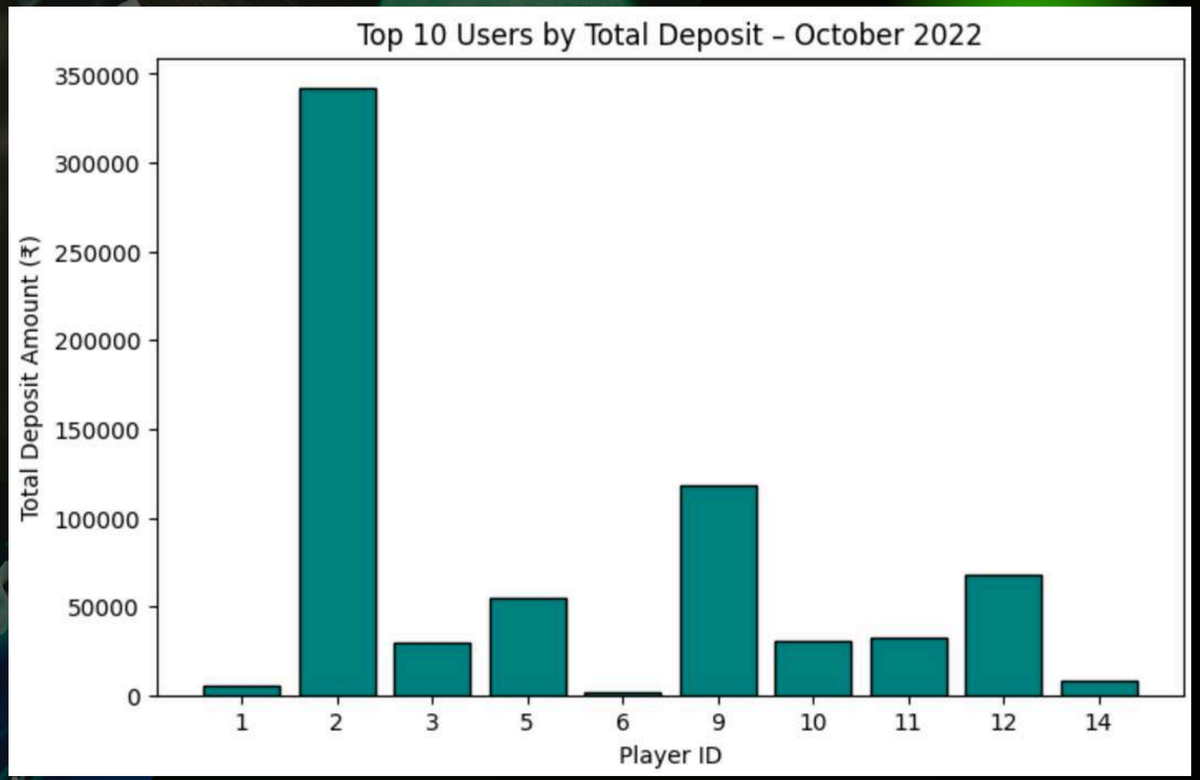
METHODOLOGY

04 Average Monthly Deposit per User

```
avg_deposit_per_user = user_total_october['Deposit_Amount'].mean()

#Print the result
print(f"Average Deposit Amount per User in October: ₹{avg_deposit_per_user:.2f}")
```

Average Deposit Amount per User in October: ₹72533.98



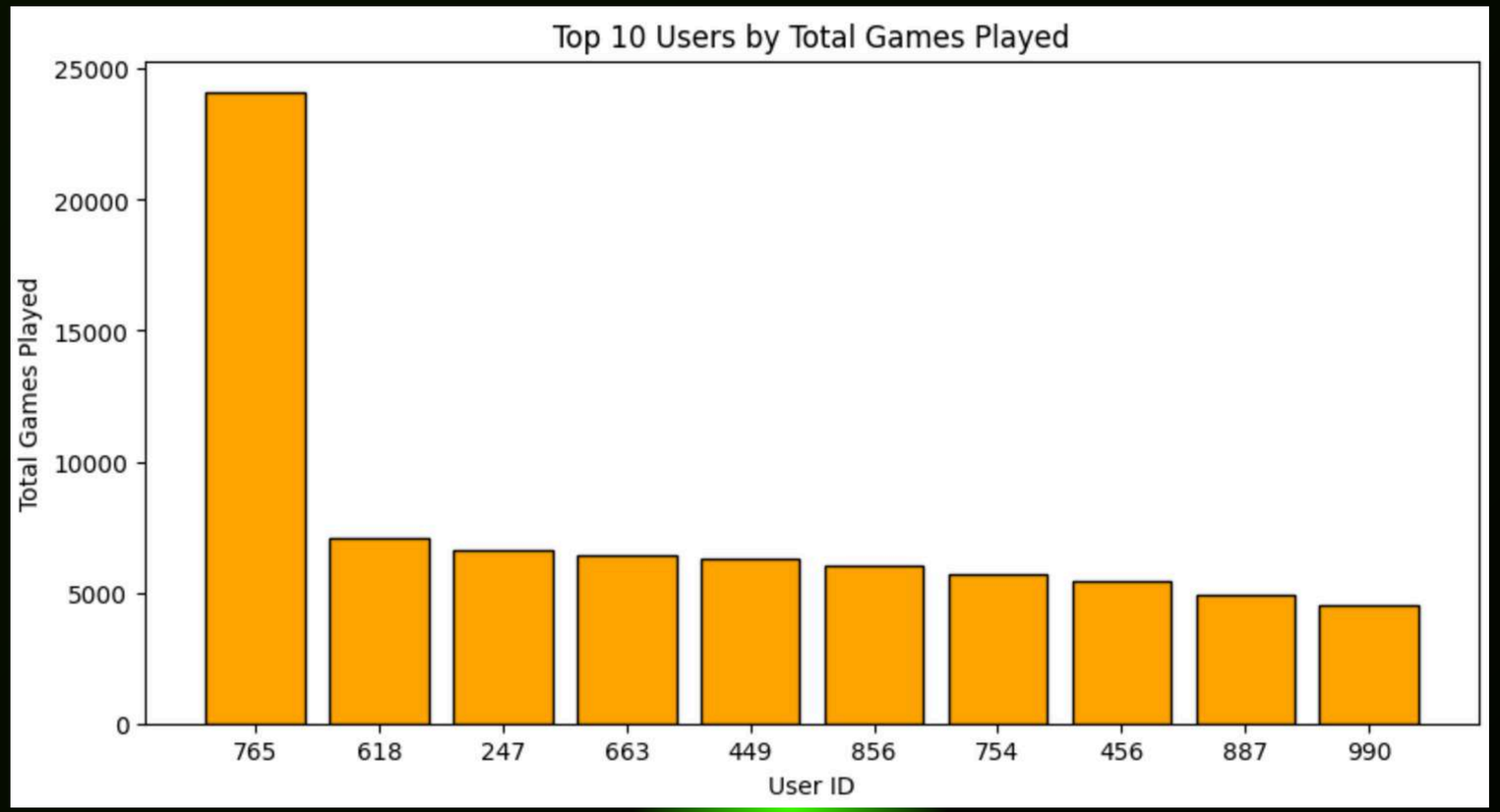
METHODOLOGY

05 Average Games Played per User

```
total_games_played = gameplay_df.groupby('User_Id')['Games_Played'].sum().reset_index()

avg_games_per_user = total_games_played['Games_Played'].mean()

print(f"Average number of games played per user: {avg_games_per_user:.2f}")
```



Average number of games played per user:
355.27

BONUS STRATEGY

Part B

Bonus Allocation Strategy for Top 50 Players

🎯 Objective:

Distribute ₹50,000 bonus among the top 50 players ranked by loyalty points in October.

💡 Chosen Strategy:

Bonus is allocated proportionally based on loyalty points earned by each player.

This ensures:

- More active and high-contributing players get higher rewards
- Encourages consistent engagement with the platform.

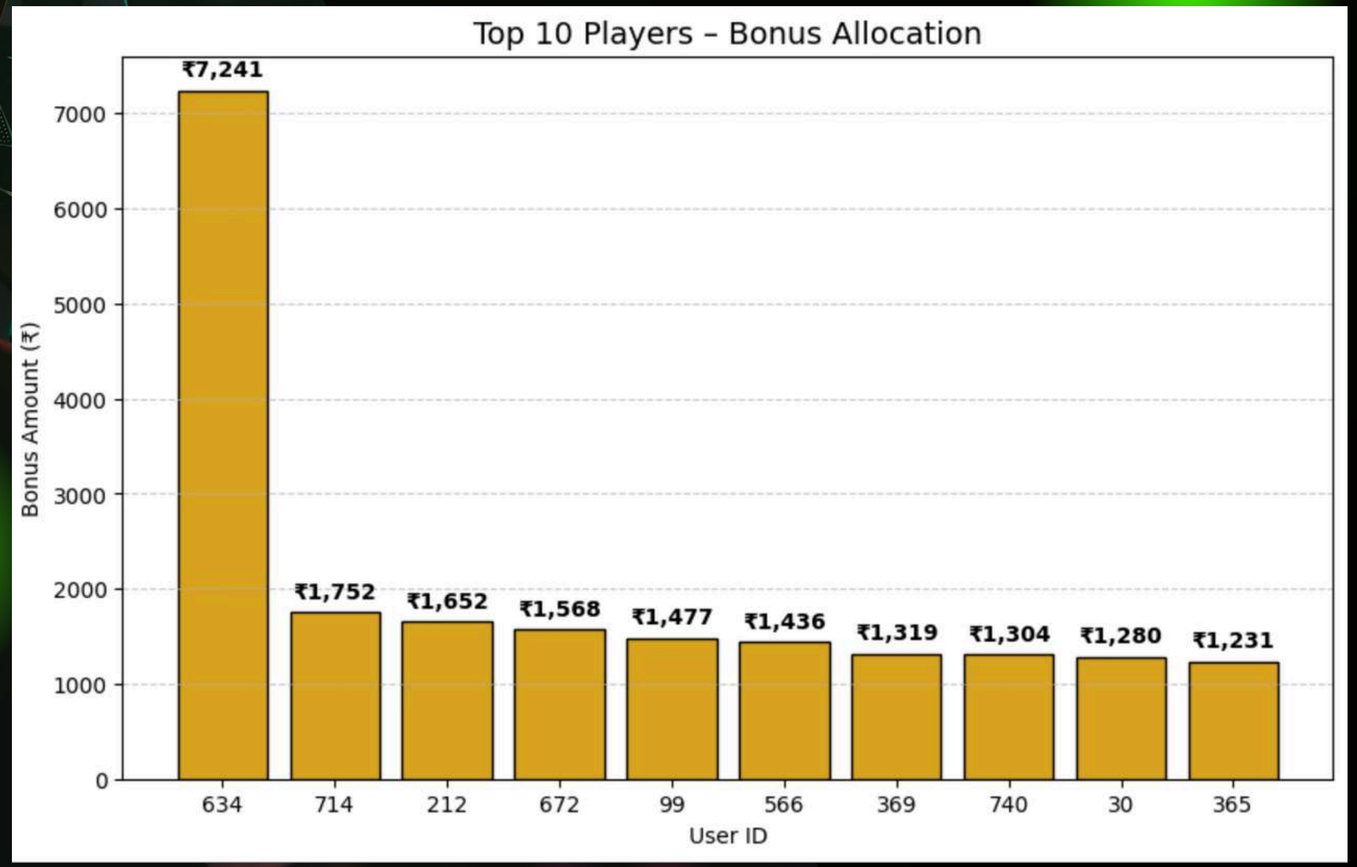
📌 Formula Used:

Bonus = (Player's Loyalty Points / Total Loyalty Points of Top 50) * ₹50,000

```
total_loyalty_top_50 = top_50['Loyalty_Points'].sum()

bonus_pool = 50000
top_50['Bonus'] = ((top_50['Loyalty_Points'] / total_loyalty_top_50) * bonus_pool).round(2

top_50[['User_Id', 'Loyalty_Points', 'games_played', 'Bonus']].head()
```



FAIRNESS EVALUATION & SUGGESTED IMPROVEMENT

Problem

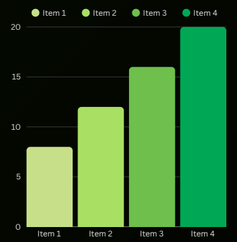
The loyalty formula gives too much weight to the number of games played, ignoring how much money was involved in those games.

Suggestion

Instead of counting only the number of games, include the value of each game (entry fee or bet amount) in the formula. This ensures that high-stake players get rewarded fairly, not just those who play more often.

KEY FINDINGS & TAKEAWAYS

- Loyalty points increase with both gameplay and transactions.
- Top 50 players account for significant loyalty share.



- Bonus allocation using proportional method ensures fairness.
- Formula improvement canmake reward system more robust.

THANK YOU

priyankajinka23@gmail.com

Priyanka Jinka

Project Portfolio