Data Analyst Assignment Report – ABC Gaming Platform

Objective-

This report presents a player loyalty analysis for ABC, a real-money gaming platform. Using player activity data (deposits, withdrawals, and games), the goal is to calculate loyalty points, rank users, and propose a bonus distribution method. Additionally, this report evaluates the fairness of the existing loyalty formula.

Dataset Description

The analysis is based on three separate datasets, each representing a specific aspect of user activity on the ABC gaming platform:

1. deposit_data

Contains records of money deposited by users on the platform.

Key columns:

1. User id – Unique identifier for each user
2. datetime – Timestamp of when the deposit occurred
3. deposit amount – Amount deposited in INR

An additional column loyalty_points was calculated using the formula:
loyalty_points = 0.01 × deposit amount

2. withdrawal_data

Includes records of money withdrawn by users from the platform.

Key columns:

1. Use rid – Unique identifier for each user
2. datetime – Timestamp of when the withdrawal occurrer
3. withdrawal amount – Amount withdrawn in INR

A new column loyalty_points was calculated using the formula:
loyalty_points = 0.005 × withdrawal amount

3. user_gameplay_data

Tracks gameplay activity by users.

- Key columns:

1.User id – Unique identifier for each user

2.datetime – Timestamp of the gameplay activity

3.number of games played by the user at particular time

Loyalty points were calculated as:
loyalty_points = 0.2 × number of games played

4. Bonus for More Deposits than Withdrawals

An additional loyalty component was calculated based on the difference between the number of deposits and number of withdrawals for each user:

bonus_points = 0.001 × max((number of deposits − number of withdrawals), 0)

Database Setup (PostgreSQL)

The data was imported and analyzed using PostgreSQL, a powerful open-source relational database system.

A dedicated database named ABC_Gaming_Database was created, containing the following tables:

Tables Created:

1. deposit_data

Stores user deposit transactions with datetime, amount, and calculated loyalty points.

2. withdrawal_data

Stores user withdrawal transactions with datetime, amount, and calculated loyalty points.

3. user_gameplay_data

Contains gameplay logs with datetime and number of games played, along with calculated loyalty points.

1. **Temporary and Derived Tables**

For each slot or analysis step **temporary tables** were created using CREATE TEMP TABLE to modularize logic and improve readability.

The database was designed to support step-wise SQL transformations, filtering by time slots, and aggregations for player-level and month-level loyalty calculations.

Part A - Calculating loyalty points (Approach and Logic)

1. Loyalty Points by Specific Time Slots

To calculate loyalty points earned by players in specific date + time slots, the following 4 slots were analyzed:

- 2nd October – Slot S1 (12:00 AM to 12:00 PM)
- 16th October – Slot S2 (12:00 PM to 12:00 AM)
- 18th October – Slot S1 (12:00 AM to 12:00 PM)
- 26th October – Slot S2 (12:00 PM to 12:00 AM)

My Approach:

1.Calculated Loyalty Points at the Table Level:

I first added a new column loyalty_points inside each base table (deposit_data, withdrawal_data, and user_gameplay_data) using ALTER TABLE.

The values were computed as per the formula:

1. 0.01 × deposit amount
2. 0.005 × withdrawal amount
3. 0.2 × number of games played

2. Slot-Wise Aggregation Using Temporary Tables:

For each slot, I filtered the data by datetime and created temporary tables to store:

1.deposit_points

2.withdrawal_points

3.game_points

I then calculated bonus points using:

bonus_points = 0.001 × max(deposit count − withdrawal count, 0)

This was also handled using temporary tables for clarity and reusability.

Final Loyalty Point Calculation:

1. All relevant temporary tables were **joined on user id** to calculate the final total_loyalty_points per user per slot.
2. These results were then used to analyze user activity and performance during specific time windows.

Note: The complete SQL queries for this section are included in the attached PostgreSQL .sql file

2. Monthly Loyalty Points Calculation and Ranking

This section addresses the task of calculating the overall loyalty points earned by each player for the month of October, and ranking them accordingly. In case of a tie in loyalty points, the number of games played was used as the tie-breaker.

My Approach:

To solve this, I followed a similar method as used in the slot-wise loyalty calculation, with the only difference being the time range.

1.Changed the Timeline:

I used the same logic from the slot-wise queries, but updated the time filter to cover the entire month:

```
WHERE EXTRACT(MONTH FROM datetime) = 10
```

2.Player Ranking:

Than i ranked player on the basis of loyalty points first and if that is tie then on the basis of number of game played by each of the players

Players were ranked using a SQL RANK() window function:

DENSE_RANK() OVER (ORDER BY total_loyalty_points DESC,
total_games_played DESC)

This ensures that if two users had the same loyalty score, the one with more games played received the higher rank.

3. KPI Calculations

To better understand user behavior on the platform, I calculated several key metrics using simple yet meaningful SQL aggregations. These metrics provide insights into how users interact with the platform in terms of money and gameplay.

Average Deposit Amount

- **Goal:** Find the average amount deposited per transaction.
- **Approach:**

1.Used the AVG(deposit amount) function on the entire deposit_data table.

2.This gives a platform-wide sense of how much users typically deposit in one go.

Average Deposit Amount Per User Per Month

- **Goal:** Determine the average monthly deposit value for each user.
- **Approach:**

1.First, grouped deposits by user and month and calculated the sum of deposits done by the user.
2.Then, calculate the **average of those monthly totals** across all users.
3.This helps identify average user spending behavior on a monthly basis.

Average Number of Games Played Per User

- **Goal:** Measure average gameplay activity per user.
- **Approach:**

1.Aggregated the total number of games played per user using SUM(number of games played).
2.Then averaged that across all users to get platform-wide engagement levels.

Part B - Bonus Distribution Strategy

The company has allocated a pool of **₹50,000** to be distributed among the **top 50 players** based on their loyalty points. The challenge is to determine a fair and logical way to distribute the bonus money among these players.

My Approach:

After calculating the **total loyalty points** for each user during October (as explained in Part A), I ranked all users and selected the **top 50 players**. These players were then considered eligible for the bonus distribution.

I chose to distribute the ₹50,000 **in proportion to each player's loyalty points** among the top 50 users.

Why Proportional Distribution Based on Loyalty Points?

1. **Loyalty points represent combined effort**: If we see the final loyalty points calculation formulae it is clear that it has taken all the metrics into account while calculation of loyalty points i.e deposit amount,withdrawal amount and number of game played by each player
2. **Fair and performance-driven**: Players who engage more receive more bonus.

Formula Used:

bonus_awarded = (player_loyalty_points / total_loyalty_points_among_top_50) × ₹50,000

Alternative Approaches Considered:

While loyalty points were the primary basis for distribution, I also considered:

1.Using **gameplay activity** (number of games played) as a secondary metric

2.A hybrid model: Ex-. 70% weight to loyalty points and 30% to games played
However, to maintain clarity and direct alignment with the loyalty scoring system, I proceeded with a **purely loyalty point-based allocation**.

Part C - Evaluation of the Loyalty Point Formula

The current loyalty point formula used by the platform is as follows:

Loyalty Points =

0.01 × deposit amount +

0.005 × withdrawal amount +

0.001 × max(deposit count − withdrawal count, 0) +

0.2 × number of games played

Is the Formula Fair?

Overall, the formula does a good job of:

1.Incentivizing users to **deposit money**, which supports the platform's revenue.

2.Encouraging **gameplay activity** with a fixed reward per game.

3.Giving a small **bonus** to users who deposit more frequently than they withdraw.

This makes the formula **easy to understand**, **simple to implement**, and aligns well with basic engagement goals.

But There Are Some Limitations:

Gameplay undervalued - 0.2 points per game gives less weight to consistent engagement .

No reward for retention - There's no incentive for daily activity, streaks, or referring others

Rewarding withdrawals - Even though it's lower (0.005), rewarding money leaving the platform may not align with business goals.

Suggestions to Make It More Robust

1. Increase the weight of gameplay

- Boost(0.2 × games to 0.5) to reward users who consistently engage.

2. Reduce or remove points for withdrawals

- Since withdrawals reduce user balance, rewarding them (even lightly) may not be beneficial.

3. Incorporate streaks or session frequency

- Award bonus points for consecutive days of login or play to drive retention.

4. Referral-based bonuses

- Encourage word-of-mouth growth by awarding points for verified referrals.