MENTORSHIP SESSIONS DATA OPTIMIZATION FOR REWARDS PROGRAM EVALUATION REPORT

Reward Data Intern Recruitment Assignment

Abstract

To evaluate and enhance the effectiveness of the Rewards Program by cleaning mentorship session data, allocating points to mentors based on defined criteria, and deriving actionable insights for improving user engagement.

Melvin Ngure

Melvinngure07@gmail.com

Contents

Table of Figures

Task 1: Data Cleaning Report

1. Process Overview:

The cleaning process followed these key steps:

a. Data Ingestion:

The dataset was imported into a Python environment using Pandas, a powerful data manipulation library. This allowed for efficient data inspection and cleaning.

b. Duplicate Removal:

First, the dataset was examined for duplicate rows. These could arise from multiple entries of the same mentorship session due to recording errors or duplication during data entry. No duplicated entries were found as per the figure below.

```
In [3]: # Checking for duplicated records
        duplicate_rows = rewards_data[rewards_data.duplicated()]

        # Print the number of duplicate rows
        print(f"Number of duplicate rows: {len(duplicate_rows)}")

        # Display the duplicate rows
        print("Duplicate Rows:")
        print(duplicate_rows)

        Number of duplicate rows: 0
        Duplicate Rows:
        Empty DataFrame
        Columns: [UID, Mentor_ID, Mentor_Name, Mentee_Name, Session_Number, Session_Duration_Min, Job_Info_Completed, Session_Date, Poi
        nts_Awarded]
        Index: []
```

Figure 1:Duplicated Records

c. Handling Missing Values:

Next, the dataset was checked for any missing or incomplete data using the isnull() function. Missing values can significantly affect data analysis. A handful of columns were found to have records with missing values and were dropped, not taking into consideration the ‘Points Awarded’ column which was to be populated later on.

```
In [5]: # Checking for nulls
        rewards_data.isnull().sum()

Out[5]: UID                    1
        Mentor_ID              1
        Mentor_Name            0
        Mentee_Name            2
        Session_Number         1
        Session_Duration_Min   2
        Job_Info_Completed     1
        Session_Date           1
        Points_Awarded       109
        dtype: int64

In [6]: # Dropping null values excluding those in 'Points Awarded' column

        columns_to_exclude = ['Points_Awarded']
        rewards_data_cleaned = rewards_data.dropna(subset=[col for col in rewards_data.columns if col not in columns_to_exclude])

        print(rewards_data_cleaned.isnull().sum())

        rewards_data_cleaned.head()

        UID                    0
        Mentor_ID              0
        Mentor_Name            0
        Mentee_Name            0
        Session_Number         0
        Session_Duration_Min   0
        Job_Info_Completed     0
        Session_Date           0
        Points_Awarded       106
        dtype: int64
```

Figure 2:Missing Values

d. Dropping of Unnecessary Columns

After meticulously going through the dataset, I noted that the was unique identifier 'UID' and that of mentors' 'Mentor_ID' were not unique at all throughout the dataset and decided to fully drop the columns. I further created new unique identifiers for each mentor, mentee as well as each recorded session.

```
In [11]: # Creating a new column with uniqueIDs for each Mentor

         rewards_data_cleaned['Mentor_Unique_ID'] = rewards_data_cleaned.groupby('Mentor_Name').ngroup() + 1
         print(rewards_data_cleaned.head())

         ...

In [12]: # Creating a new column with uniqueIDs for each Mentee

         rewards_data_cleaned['Mentee_Unique_ID'] = rewards_data_cleaned.groupby('Mentee_Name').ngroup() + 1
         print(rewards_data_cleaned.head())

         ...

In [13]: # Creating a uniqueID relating each Mentor to a Mentee

         rewards_data_cleaned['Unique_ID'] = rewards_data_cleaned['Mentor_Unique_ID'].astype(str) + '_' + rewards_data_cleaned['Mentee_Un
         print(rewards_data_cleaned.head())
```

Figure 3:Unnecessary Columns

e. Data Type Correction:

Numeric columns like 'Session_Duration' and 'Session_Duration_Min' were appropriately assigned to their correct data type format (integer). 'Session_Date' was also re-assigned to datetime format.

```
In [9]: # Assigning appropriate datatypes to various columns

rewards_data_cleaned['Mentor_Name'] = rewards_data_cleaned['Mentor_Name'].astype(str)
rewards_data_cleaned['Mentee_Name'] = rewards_data_cleaned['Mentee_Name'].astype(str)
rewards_data_cleaned['Session_Number'] = rewards_data_cleaned['Session_Number'].round().astype(int)
rewards_data_cleaned['Session_Duration_Min'] = rewards_data_cleaned['Session_Duration_Min'].round().astype(int)
rewards_data_cleaned['Session_Date'] = pd.to_datetime(rewards_data_cleaned['Session_Date'])

rewards_data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 106 entries, 0 to 108
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Mentor_Name           106 non-null    object
 1   Mentee_Name           106 non-null    object
 2   Session_Number        106 non-null    int32
 3   Session_Duration_Min  106 non-null    int32
 4   Job_Info_Completed    106 non-null    object
 5   Session_Date          106 non-null    datetime64[ns]
 6   Points_Awarded        0 non-null      float64
dtypes: datetime64[ns](1), float64(1), int32(2), object(3)
memory usage: 5.8+ KB
```

Figure 4:Datatype Correction

f. Verification and Quality Check:

After cleaning, the dataset was reviewed to ensure all operations were successfully applied. A quick summary of the dataset, including its shape and column details, was generated to verify the data quality and saved to a new Excel

2. Tools Used:

Python: The primary programming language used for this task.

Pandas: A Python library for data analysis and manipulation, used extensively for loading, cleaning, and inspecting the dataset.

Jupyter Notebook: An interactive environment for performing the data cleaning and documenting the process simultaneously.

3. Findings:

Duplicates: No duplicated records were found thus none removed.

Missing Values: A few entries were missing critical information, such as session dates or mentee name, and were excluded from the analysis.

Unnecessary Columns: The columns ‘UID’ and ‘Mentor_ID’ were of no use to the dataset and had to be dropped and new unique IDs recreated for the various columns.

Standardization: Reassignment of columns to their appropriate datatypes to ensure the integrity of the data in further analysis processes.

Task 2: Legacy Point Allocation

Rules for Point Allocation

The points allocation process is governed by three main rules:

Signup Points: Every mentor who participates in the program is awarded 250 points upon signing up.

Two Mentees Rule: Mentors who have mentored at least two different mentees are awarded an additional 1000 points.

Mentorship Points: Mentors receive 500 points for mentoring the same mentee for at least two sessions that meet the following criteria:

Each session must be at least 30 minutes long.

At least one of the sessions must have "Job Info Completed" marked as "Yes."

Brief Report on Point Allocation Results

The point allocation system rewards mentors based on three key aspects: their initial signup, their involvement with multiple mentees, and the quality and quantity of their mentorship sessions.

Signup Points were uniformly awarded to all mentors, ensuring that every participant in the program starts with an equal base score.

Two Mentees Points were awarded to mentors who mentored at least two different mentees, incentivizing mentors to engage with more participants in the program. Mentors who focused on a single mentee did not receive these points, reinforcing the program's objective of broadening mentorship reach.

Mentorship Points rewarded mentors for conducting at least two sessions of significant duration (30 minutes or more) and providing job-related guidance ("Job Info Completed"). This highlights the program's focus on both quantity (multiple sessions) and quality (significant duration and job-related mentorship).

In total, mentors who demonstrated consistent engagement across multiple mentees and high-quality mentorship received the highest scores. This points allocation approach ensures that mentors are rewarded for broadening their impact and providing meaningful, job-related advice, aligning with the program's goals of enhancing professional development through mentorship.

Step-by-Step Outline for Point Allocation Process

Data Preparation:

Import the dataset and ensure that all necessary columns (Mentor_Unique_ID, Mentee_Name, Session_Duration_Min, Job_Info_Completed, etc.) are properly formatted.

Clean the data by removing duplicates and handling any missing values.

Rule Implementation:

Assign 250 points to all mentors for signing up by adding a Signup_Points column.

Group by Mentor_Unique_ID and count the unique mentees. Award 1000 points to mentors who have mentored at least two different mentees by adding a Two_Mentees_Points column.

For each mentor-mentee pair, apply the session duration and job info criteria. If both conditions are met, award 500 points in the Mentorship_Points column.

Total Points Calculation:

Create a Total_Points column by summing the Signup_Points, Two_Mentees_Points, and Mentorship_Points.

Testing and Validation:

Unit Testing: For each rule, manually verify a few cases to ensure that the points are being awarded correctly (e.g., check if mentors with two mentees received 1000 points).

Edge Case Testing: Test scenarios with mentors who have exactly one mentee, or whose sessions are shorter than 30 minutes, to confirm no points are incorrectly awarded.

Task 2: Legacy Point Allocation

In [16]:
```
# Assigning initial points for signing up (250 points)
rewards_data_cleaned['Signup_Points'] = 250

# Allocating 1000 points for mentors who have mentored at least 2 different mentees
mentee_count = rewards_data_cleaned.groupby('Mentor_Unique_ID')['Mentee_Name'].nunique()
mentors_with_two_mentees = mentee_count[mentee_count >= 2].index
rewards_data_cleaned['Two_Mentees_Points'] = rewards_data_cleaned['Mentor_Unique_ID'].apply(lambda x: 1000 if x in mentors_with_

# Allocating 500 points for mentoring the same mentee for two sessions, meeting criteria
# a. Sessions must be >= 30 minutes and
# b. At least one session must have job info completed ("Yes")

def allocate_points(group):
    if len(group) >= 2:
        if all(group['Session_Duration_Min'] >= 30) and any(group['Job_Info_Completed'] == 'Yes'):
            return 500
    return 0

mentee_group = rewards_data_cleaned.groupby(['Mentor_Unique_ID', 'Mentee_Name']).apply(allocate_points).reset_index(name='Mentor

# Merging the mentorship points back to the main DataFrame
rewards_data_cleaned = pd.merge(rewards_data_cleaned, mentee_group, on=['Mentor_Unique_ID', 'Mentee_Name'], how='left')

# Calculating Total Points for each row
rewards_data_cleaned['Total_Points'] = rewards_data_cleaned['Signup_Points'] + rewards_data_cleaned['Two_Mentees_Points'] + rewa

rewards_data_cleaned[['Unique_ID','Signup_Points', 'Two_Mentees_Points', 'Mentorship_Points', 'Total_Points']]
```

Out[16]:

	Unique_ID	Signup_Points	Two_Mentees_Points	Mentorship_Points	Total_Points
0	5_2	250	1000	0	1250
1	2_3	250	1000	0	1250
2	3_4	250	1000	0	1250
3	2_2	250	1000	500	1750
4	1_4	250	1000	0	1250

Figure 5:Legacy Point Allocation

Task 3: Deriving Reward Dashboard and Insights

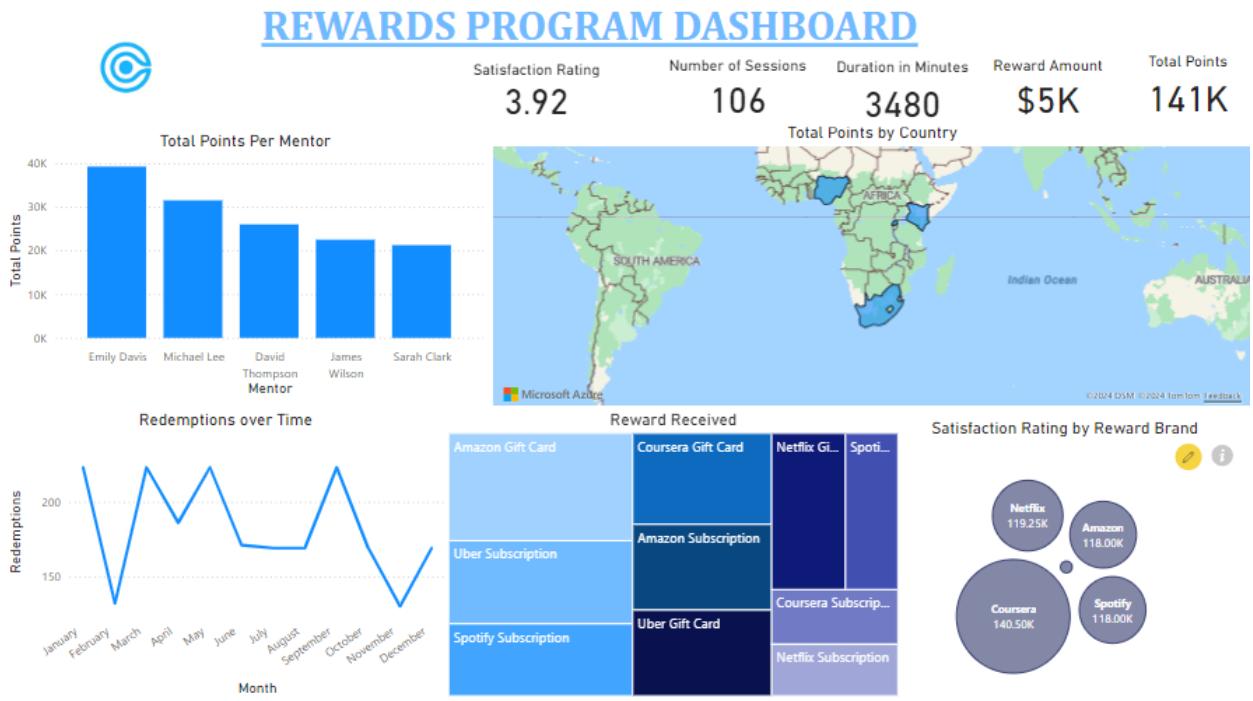

Figure 6:Dashboard

Actionable Recommendations

1. Enhance the Speed of Reward Fulfillment to Improve User Satisfaction

- **Objective**: Decrease the time it takes for users to receive rewards, particularly for new users and low engagement segments, where delays are associated with lower satisfaction.
- **Implementation**:

  Streamline the reward fulfillment process, especially for digital rewards, which can be distributed instantly.

  Partner with logistics providers to optimize delivery times for physical rewards in regions experiencing delays.

2. Introduce Regionalized Reward Offerings to Meet Country-Specific Preferences

- **Objective**: Cater to the geographic preferences observed in different regions, which can increase engagement by offering rewards that are more relevant to users in specific countries.
- **Implementation**:

  Based on data, adjust reward offerings to include region-specific options. For example, in Country A, expand the selection of digital rewards, while in Country B, focus on physical goods or vouchers that align with user preferences.

Incorporate localized promotional campaigns to highlight rewards that resonate most with users in each country.

3. Utilize Data Analytics to Personalize Reward Recommendations

- **Objective**: Improve user engagement by providing personalized reward suggestions based on user behavior, preferences, and past activity.
- **Implementation**:

  Leverage user data: Analyze user engagement patterns, session length, and reward redemption history to predict preferences and recommend personalized rewards. For example, if a mentor frequently redeems digital vouchers, prioritize showing them similar options.

  Dynamic reward catalog**:** Tailor the reward catalog each user sees to highlight the most relevant rewards, based on their engagement level and geographic location. This personalization can increase the likelihood of redemption and satisfaction.

  Continuous feedback loop: Collect feedback on user satisfaction with recommended rewards and adjust the algorithms accordingly to improve future personalization efforts.