User Insights, Data Discrepancies and Scalability

User Insights

1. There are a total of 17419 users for 20 distinct games.
2. Game with game_id = 55107008, appears to be the most popular game, with a user base of 11495.
3. Surprisingly, with highest user base of 11495, game_id 55107008 is not the most addictive. The most addictive game is game_id 10655437. With user base of only 797 users, game_id 10655437 is most addictive, because 33 of its users lie in top 35 users who have played most of any of the games. This is a nice game to post advertisements which would give fruitful results.
4. Games with game_ids 94868282, 82196685 and 85159873 do not have considerable sessions.
5. SDK version 7.6 seems to be most popular among the users with 13163 distinct users

Data discrepancies

There are number of data discrepancies in the provided dataset. Please find some of them below:

1. **Scenario 1:** For a game-device combination, the total number of start time is more than number of stop time.
   Example: game_id – 55107008, Device-id - 15c25555d36b7877551ada2863193db3
   Fix 1: We consider the minimum of the stop time which is just greater than a start time. With this fix, the latest start time of a game-device combination will be neglected. (Step 1 of the stored procedure)

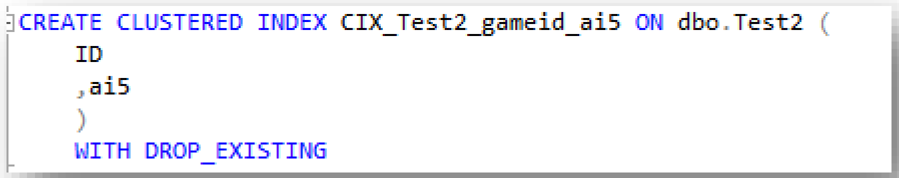
2. **Scenario 2:** For a game-device combination, the total number of stop time is more than number of start time.
   Example: game_id – 32950170, Device-id - 0c054efac80f96a9e7dc62a81482a228
   Fix: With Fix1 for Scenario1, this issue is also fixed. (Step 1 of the stored procedure)

3. **Scenario 3:** After fixing above two scenarios, we get multiple start times for a single stop time.
   Example: game-id – 30900473, Device-id - 4b53782b166270536859512da41f56d7
   Fix: We consider the minimum start time for the stop time. (Step 2 of the stored procedure)

Scalability

Considering the raw traffic file contains data equal to 100 times that present in current log file. We can follow steps below:

1. Create a stored procedure which contains all the code compiled at one place. The stored procedures are compiled once and stored in executable form, therefore providing quick and efficient execution and processing.
2. Store the [ai5] column (the device id) column in a uniqueidentifier type of column. Storing [ai5] as guid is preferable since, it is only 16 bytes and storing it in varchar or nvarchar would require 32 bytes atleast. Therefore, if the table is large, saving 16 bytes (32 -16) per row is considerable.
3. Further we can create a clustered index on game_id and device_id(ai5) column such as follows:



```
CREATE CLUSTERED INDEX CIX_Test2_gameid_ai5 ON dbo.Test2 (
    ID
    ,ai5
    )
    WITH DROP_EXISTING
```

4. In case of further data increase, we can create index on a file group other than the file group of the base table. This file group (.ndf file) can be kept in some other drive than the drive which contains file group(.mdf file) of base table.
5. Also, we can create a mapping table of ai5 (uniqueidentifier) with integer values, i.e. generate identity for each ai5. We can use this integer value in our stored procedure, resulting in further decrease in processing time.