Mobile Gaming Analytics Bootcamp Project

**JUNE 2024**

**MİNE KARA**

# PART 1

**QUESTION 1: Calculate D14 LTV. Interpret it.**

Normally, we need 14 days of data to calculate the D14 LTV value. However, we only have 8 days of data. D14 LTV represents the total revenue generated by a user up until the 14th day after installing the app.

**LTV = Total cohort revenue / Cohort Size**

For D14, LTV is the total revenue over 14 days divided by the cohort size. Since we don’t have exact values from our data, we need to make an estimate. Using the monetization curve, we can estimate revenue and retention rates for day 14.

**QUESTION 2: What is the share of interstitial ad revenue in the total daily revenue? Interpret.**

The ratio of interstitial ad revenue to total daily revenue is shown in the graph below. According to the graph, on some days, interstitial ads constitute a large share of the total revenue, while on other days, this share decreases, possibly due to factors like ad campaigns. Overall, within these 8 days, interstitial ads comprise a larger percentage of total ad revenue compared to other ad types like banner and rewarded ads. This ratio ranges between appr. 76% and 92%.

A graph with blue lines and numbers

Description automatically generated

# 

PART 2

Suppose a marketing campaign with a $10 CPI acquires 1000 players, and Day 1 retention is 50% (i.e., half of the players reopen the game the day after installing it). Day 2 retention is 49%, and Day 3 retention is 48%, decreasing by 1% daily until no one plays again (retention is 0% by Day 51).

**QUESTION 1: What should the average daily ARPDAU (Average Revenue Per Daily Active User) be to reach the break-even point on Day 50?**

The break-even point occurs on the day when the LTV value equals CPI, meaning we need to find the day when **LTV = CPI**. Since ARPDAU is assumed to be constant:

**ARPDAU = revenue / Daily Active Users (DAU)**

**LTV = total cohort revenue / cohort size**

Here, the cohort size is 1000 users. Daily revenue can be derived from the first equation:

**Daily Revenue = ARPDAU (A) \* DAU**

**Example Table for Daily Retention Rates, Users, ARPDAU, and Daily Revenue:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Day** | **Retention Rate** | **Kullanıcı Sayısı** | **ARPDAU** | **Revenue (daily)** |
| D1 | 0.5 | 500 | A | 500 x A |
| D2 | 0.49 | 245 | A | 245 x A |
| D3 | 0.48 | 118 | A | 118 x A |
| D4 | 0.47 | 56 | A | 56 x A |
| D5 | 0.46 | 26 | A | 26 x A |
| D6 | 0.45 | 12 | A | 12 x A |
| D7 | 0.44 | 6 | A | 6 x A |
| D8 | 0.43 | 3 | A | 3 x A |
| D9 | 0.42 | 1 | A | 1 x A |
| D10 | 0.41 | 1 | A | 1 x A |
| **….** | **….** | **….** | **….** | **….** |
|  | **Total** | **1008** | **A** | **1008 x A** |

Total cohort revenue for 50 days would be **1008 x ARPDAU (A)**. To solve:

**1008 x A / 1000 = $10,** So **A (ARPDAU) = $9.92**.

**QUESTION 2: How much does assuming ARPDAU is constant reflect reality? If you think it does, why? If you think it doesn't, what would be your assumptions that could cause a change?**

Frankly, I think that assuming ARPDAU as constant does not fully reflect reality. Because ARPDAU value is not a metric that can change depending on a single factor. Also, even if it were only based on the number of daily active users, even instant changes in human behavior could easily affect ARPDAU. Apart from this, there may be seasonal or trend effects in users' interactions with the game. For example, people may interact with the game more during their free time on the weekend. In addition, in-game dynamics that may change with updates may also cause changes in users' in-game spending.

# PART 3

**QUESTION 1: If you want to detect cheaters in the game, which table data would you use? Perform a simple cheat detection.**

1 - I analyzed the character health and attack values according to the levels players are on. I calculated the average and standard deviations of character health and attack values and flagged players with values more than three standard deviations from the mean as cheaters.

**QUERY:**

WITH stage\_stats AS (

    SELECT

        stage\_index,

        AVG(CAST(character\_health AS FLOAT64)) AS avg\_health,

        STDDEV(CAST(character\_health AS FLOAT64)) AS std\_health,

        AVG(CAST(character\_attack AS FLOAT64)) AS avg\_attack,

        STDDEV(CAST(character\_attack AS FLOAT64)) AS std\_attack

    FROM game-analysis-01.project\_game\_v2.stage\_events

    GROUP BY stage\_index

)

SELECT

    se.user\_id,

    se.stage\_index,

    CAST(se.character\_health AS FLOAT64) AS character\_health,

    CAST(se.character\_attack AS FLOAT64) AS character\_attack,

    se.event\_date

FROM game-analysis-01.project\_game\_v2.stage\_events se

JOIN stage\_stats ss ON se.stage\_index = ss.stage\_index

WHERE

    (CAST(se.character\_health AS FLOAT64) > ss.avg\_health + 3 \* ss.std\_health OR

     CAST(se.character\_health AS FLOAT64) < ss.avg\_health - 3 \* ss.std\_health) OR

    (CAST(se.character\_attack AS FLOAT64) > ss.avg\_attack + 3 \* ss.std\_attack OR

     CAST(se.character\_attack AS FLOAT64) < ss.avg\_attack - 3 \* ss.std\_attack)

ORDER BY stage\_index;

**OUTPUT: (İlk 15 satır)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| user\_id | stage\_index | character\_health | character\_attack | event\_date |
| 13357580 | 10 | 4.81E+41 | 8.29E+100 | 10/29/2023 |
| 13357580 | 10 | 4.81E+41 | 8.29E+100 | 10/29/2023 |
| 12465913 | 10 | 6.71E+96 | 2.56E+96 | 10/3/2023 |
| 12465913 | 10 | 6.71E+96 | 2.56E+96 | 10/3/2023 |
| 13357580 | 11 | 4.81E+41 | 8.29E+100 | 10/29/2023 |
| 13357580 | 11 | 4.81E+41 | 8.29E+100 | 10/29/2023 |
| 12465913 | 11 | 6.71E+96 | 2.56E+96 | 10/3/2023 |
| 12465913 | 11 | 6.71E+96 | 2.56E+96 | 10/3/2023 |
| 13357580 | 12 | 2.55E+42 | 3.81E+101 | 10/29/2023 |
| 13357580 | 12 | 2.55E+42 | 3.81E+101 | 10/29/2023 |
| 12465913 | 12 | 8.12E+96 | 2.82E+96 | 10/4/2023 |
| 12465913 | 12 | 8.12E+96 | 2.82E+96 | 10/4/2023 |
| 13256277 | 13 | 4.56E+31 | 1.30E+30 | 10/25/2023 |
| 13256277 | 13 | 4.56E+31 | 1.30E+30 | 10/25/2023 |

As shown, it’s improbable for players in level 11 to have 4.81x10^41 character health.

2 - I performed another analysis. This time, I examined the currency change amount according to the level the players were at. Again, using the same logic, I found the points that were far from the average and obtained the output of the players who showed abnormal behavior.

**QUERY:**

WITH currency\_stats AS (

    SELECT

        se.stage\_index,

        AVG(CAST(cc.currency\_change\_amount AS INT64)) AS avg\_currency,

        STDDEV(CAST(cc.currency\_change\_amount AS INT64)) AS std\_currency

    FROM game-analysis-01.project\_game\_v2.stage\_events se

    JOIN game-analysis-01.project\_game\_v2.currency\_changes cc

    ON se.user\_id = CAST(cc.user\_id AS INT64)

    GROUP BY se.stage\_index

)

SELECT

    cc.user\_id,

    se.stage\_index,

    cc.currency\_change\_amount,

    cc.event\_date

FROM game-analysis-01.project\_game\_v2.currency\_changes cc

JOIN game-analysis-01.project\_game\_v2.stage\_events se

ON CAST(cc.user\_id AS INT64) = se.user\_id

JOIN currency\_stats cs

ON se.stage\_index = cs.stage\_index

WHERE

    CAST(cc.currency\_change\_amount AS INT64) > cs.avg\_currency + 3 \* cs.std\_currency OR

    CAST(cc.currency\_change\_amount AS INT64) < cs.avg\_currency - 3 \* cs.std\_currency

ORDER BY stage\_index;

**OUTPUT: (İlk 15 satır)**

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id | stage\_index | currency\_change\_amount | event\_date |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |
| 12927055 | 1 | 87699872 | 10/19/2023 |

A graph with blue squares

Description automatically generated

As seen in the graph above, the levels where the currency change is high for those at level 100 (which is a big currency change even for them in my opinion) are generally the first 10 levels. It is obvious that the currency changes of users at these levels cannot be this high. Therefore, these players are also included in the category of fraudulent players.

**QUESTION 2: Based on current data and information, what do you think is the best source for players to earn gems?**

We can examine the resources that earn the most gems by looking at the reason, currency change amount, currency type and change type data in the "currency changes" table.

**QUERY:**

SELECT

    reason,

    COUNT(\*) AS num\_transactions,

    AVG(CAST(cc.currency\_change\_amount AS INT64)) AS avg\_gems\_per\_transaction,

    SUM(CAST(cc.currency\_change\_amount AS INT64)) AS total\_gems

FROM game-analysis-01.project\_game\_v2.currency\_changes cc

WHERE currency\_type = 'Gem' AND change\_type = 'Gain'

GROUP BY reason

ORDER BY total\_gems DESC;

**OUTPUT:**

|  |  |  |  |
| --- | --- | --- | --- |
| **reason** | **num\_transactions** | **avg\_gems\_per\_transaction** | **total\_gems** |
| login | 36811 | 228267.1913 | 8402743580 |
| quest\_new\_user | 4885 | 839466.8393 | 4100795510 |
| market\_offer | 5703 | 526115.5935 | 3000437230 |
| other\_01 | 15541 | 135226.4333 | 2101554000 |
| quest | 532374 | 3763.420941 | 2003547460 |
| enemy\_kill\_02 | 25824 | 77273.71577 | 1995516436 |
| game\_mode\_01 | 22838 | 44327.36229 | 1012348300 |
| chest\_05 | 93991 | 4484.956006 | 421545500 |
| chest\_02 | 129758 | 2337.281709 | 303281000 |

A graph with red bars

Description automatically generated

A graph with red bars

Description automatically generated

According to the graphs above;

The highest total gem earnings are provided by the “login” process. In other words, we can say that logging in regularly is an important gem source for players.

The source that provides the highest average gem earnings per transaction is the “quest\_new\_user” process. It is seen that important rewards are provided to new users in the game.

In the “quest” process, although the total gem count is high, it is seen that it provides low earnings when looked at the average. In other words, quests are done but they do not provide a very large gem income.

**QUESTION 3: What is the most likely stage for a player still active on Day 5?**

**QUERY:**WITH day5\_active\_users AS (

    SELECT CAST(ud.user\_id AS INT64) AS user\_id

    FROM game-analysis-01.project\_game\_v2.users\_daily ud

    JOIN game-analysis-01.project\_game\_v2.users u

    ON CAST(ud.user\_id AS INT64)= u.user\_id

    WHERE ud.event\_date = DATE\_ADD(u.install\_date, INTERVAL 5 DAY)

),

stage\_counts AS (

    SELECT

        se.user\_id,

        se.stage\_index,

        COUNT(\*) AS stage\_count

    FROM game-analysis-01.project\_game\_v2.stage\_events se

    JOIN day5\_active\_users dau ON se.user\_id = dau.user\_id

    GROUP BY se.user\_id, se.stage\_index

)

SELECT

    stage\_index,

    COUNT(\*) AS num\_users

FROM stage\_counts

GROUP BY stage\_index

ORDER BY num\_users DESC

LIMIT 5;

A graph with a bar

Description automatically generated with medium confidence

According to the graph above, the most likely stage for a player still in the game on Day 5 is Stage 1. In other words, the stage that players are most often in is Stage 1. If I understand correctly, the Stage index number indicates the stage the player is in. At this point, I was surprised to see players still in Stage 1 on Day 5. If the game has a long tutorial process, perhaps players are spending the first few days playing the tutorial.

**QUESTION 4: Which stage is a player most likely on after their 15th attempt?**

**QUERY:**

WITH stage\_attempts AS (

    SELECT

        user\_id,

        stage\_index,

        SUM(CAST(stage\_attempt AS INT64)) AS total\_attempts

    FROM game-analysis-01.project\_game\_v2.stage\_events

    GROUP BY user\_id, stage\_index

),

cumulative\_att AS (

    SELECT

        user\_id,

        stage\_index,

        total\_attempts,

        SUM(total\_attempts) OVER (PARTITION BY user\_id ORDER BY stage\_index) AS cumulative\_attempts

    FROM stage\_attempts

)

SELECT

    stage\_index,

    COUNT(\*) AS num\_users

FROM cumulative\_att

WHERE cumulative\_attempts >= 15

GROUP BY stage\_index

ORDER BY num\_users DESC

LIMIT 1;

**OUTPUT:**

|  |  |
| --- | --- |
| stage\_index | num\_users |
| 1 | 47028 |

The output suggests that players on their 15th attempt are most likely to be on stage index = 1.

**QUESTION 5: Create and answer your own question based on the data.**

**Question: I will examine the relationship between play duration and gem and gold accumulation.**

**QUERY:**

WITH player\_retention AS (

    SELECT

        u.user\_id,

        MIN(ud.event\_date) AS first\_play\_date,

        MAX(ud.event\_date) AS last\_play\_date,

        DATE\_DIFF(MAX(ud.event\_date), MIN(ud.event\_date), DAY) AS days\_active

    FROM game-analysis-01.project\_game\_v2.users u

    JOIN game-analysis-01.project\_game\_v2.users\_daily ud

    ON u.user\_id = CAST(ud.user\_id AS INT64)

    GROUP BY u.user\_id

),

gem\_gold AS (

    SELECT

        pr.user\_id,

        pr.days\_active,

        us.current\_gem,

        us.current\_gold,

    FROM player\_retention pr

    JOIN game-analysis-01.project\_game\_v2.user\_states us

    ON pr.user\_id = us.user\_id

    WHERE us.event\_date = pr.last\_play\_date

)

SELECT

    gg.days\_active,

    AVG(SAFE\_CAST(gg.current\_gem AS FLOAT64)) as avg\_gem,

    AVG(SAFE\_CAST(gg.current\_gold AS FLOAT64)) as avg\_gold,

FROM gem\_gold gg

JOIN game-analysis-01.project\_game\_v2.user\_states us

ON gg.user\_id = us.user\_id

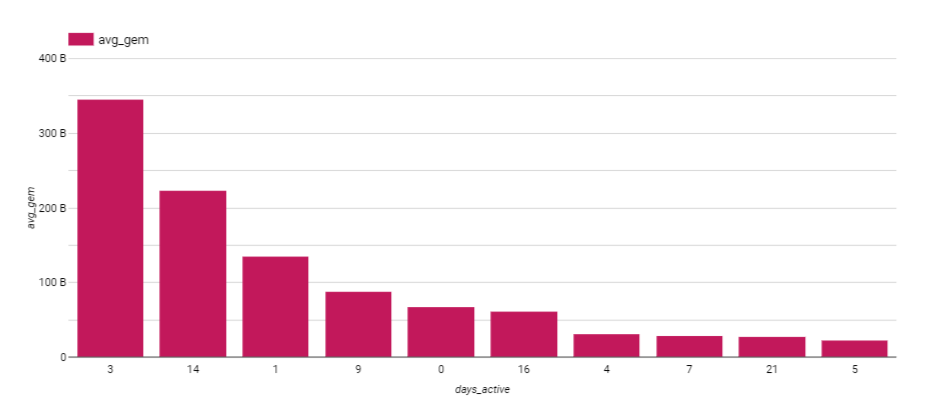
GROUP BY gg.days\_active

ORDER BY gg.days\_active DESC;

OUTPUT:

|  |  |  |
| --- | --- | --- |
| days\_active | avg\_gem | avg\_gold |
| 3 | 345794.8998 | 180632.9947 |
| 14 | 223560.9891 | 29193.5753 |
| 1 | 134972.2953 | 123154.4956 |
| 9 | 88025.86874 | 54976.62768 |
| 0 | 68159.86226 | -60702.34941 |
| 16 | 61311.37464 | 8086.483947 |
| 4 | 31761.97471 | 35617.57268 |
| 7 | 29425.78517 | 48220.59729 |
| 21 | 27541.31225 | 23760.55467 |

**Days active - Average Gem grafiği:**



The average gem amount of players who stayed in the game for the first three days seems quite high (345,794.8998). This may indicate that new players are getting more gems at the beginning or that these players are making in-game purchases.

The average gem amount of players who stayed in the game for 14 days is also good, although not as high as the first three days, but the gem amount of players who stayed in the game for 16 days is relatively low. Strategies can be developed for players who joined the game on the 14th day to increase the gems in the game in the next 2 days.

**Days active - Average Gold graph:**

A graph with lines and numbers

Description automatically generated

Players who are active for 41 days are seen to earn a high amount of gold. This may be because special events or promotions were offered to players on that day. Or, the game developers may have distributed extra gold to those who reached the 41st day.