GAME PERFORMANCE & AB TEST INSIGHTS PROJECTS

Below you'll find two tasks:

1. Game performance analysis for a soft launch game — to assess your knowledge of key game metrics used to track game health, growth, and performance

2. A/B test analysis for a new feature — to assess your knowledge of A/B testing, both product and statistics parts of it, and game analytics overall.

Both tasks are similar to what we have to tackle every day and should give you a high-level understanding of the work we do.

What we are looking for:

- Your understanding of the objective.
- Your approach to data manipulation.
- Your approach to the knowledge discovery process from data.
- Your intuition about the data given.
- Your intuition and approach to product decisions.
- The tools that you use in the process.
- How you share your results.

You can apply any methods to come up with insights based on the data given. Beyond these, please feel free to come up with your own metrics, and include any additional you believe should be contributed to the analysis, if not provided in the data set.

It is expected that you use SQL along with a programming language of your choice (Python preferred) for this assignment. Please do not forget to send your code and scripts you used for your calculations along with your test.

Remember to focus on the main question in each task but make sure to provide both WHAT is happening in data and WHY it is happening when answering it.

TASK 1: Soft Launch Performance

Main Question: Should we release the game? Why or why not?

Attached you will find data for a soft launch IAP title “Merge Data!”. In every game we release globally there are 2 prior launches, Fake launch and Soft launch. Soft Launches take place if the game successfully passes the fake launch and to see if the game is ready for a global launch.
We would like to know how the game performed during the Soft Launch and given the performance should we release the game globally or not.

The game “Merge Data!” is similar to what we have in store like Merge Dragons! or Merge Magic! but in a different concept. You can play any of those games in order to make sense of the data we’ve provided.

The dataset has 7 tables: install, login, revenue, endgame, startgame, econ_spend_hc, cost and all of them have a prefix of “dataset_sl_” in the database. You will find the definitions of these tables and corresponding columns below.
The dataset is stored in a PostgreSQL database and the credentials you need to access the database are in the attachment that is provided to you in a text file. If you would like to create new tables and manipulate the data in any way you want, you have full control over creating new tables.

TASK 2: Wonder Crafters Feature A/B test

Main Question: Should we release the feature? Why or why not?

In the database you have given access, you will find tables regarding an A/B test of a new feature called Wonder Crafters. We A/B test every feature before / if we release it so the insights derived from each experiment are crucial for product development. We always look into various metrics, covering monetisation, engagement, interactions with the feature, and other factors, **based on which we make a final decision <u>whether the feature should be rolled out to everyone in the game or not</u>.**

Wonder Crafters is a feature designed to address Resource Management issues in the game. A player can convert wonders into crafters, which will unlock crafting recipes. These recipes will allow players to create new items by using their existing items in their camp, the main area of the game. Crafting serves as both a means to use everyday items and a way to clean up clutter in their camp.

Players need to have at least 700 Dragon Power to unlock this feature.

Since you're not able to play the feature, here's detailed explanation of how it works:

- The feature is accessible from both Wonder Crafter beacon & the crafters which you transformed from your wonders.
- You first need to choose a crafting recipe from a crafter.
    - You have a limited number of slots for the recipes so you should choose a recipe which you are likely to complete.
    - Once the recipe is in one of your slots, you cannot remove it until you complete the recipe.
    - You can purchase more extra slots, if you need.
- You add crafting ingredients into the recipe. You can create the ingredients by merging in your camp.
    - If you don't have enough ingredients, you can buy them with gems.
- After collecting the ingredients, using your lumens, you can start crafting.
    - You can earn free lumens daily.
    - If you want more lumens, you can spend gems.
- After crafting is over (each recipe takes a different amount of time), you enter the crafter and obtain your final item.
    - If you don't want to wait, you can accelerate crafting by spending gems.
- You can earn more crafting recipes and more item options to craft by upgrading crafters.

- To upgrade your crafters, you need to craft a certain number of items through the feature. This depends on your crafter and your crafter's level,
- You can also spend gems to accelerate the upgrading process.

The dataset has 7 tables: a/b test state, login, revenue, duration, economy spend and earn. All of them have a prefix of "DATASET_CASE202425_ABTEST_" in the database. You will find the definitions of these tables and corresponding columns in the appendix.

The dataset is stored in a PostgreSQL database and the credentials you need to access the database are in the attachment that is provided to you in a text file. If you would like to create new tables and manipulate the data in any way you want, you have full control over creating new tables.

APPENDIX: DATA DEFINITIONS

Task 1:

- **Common Fields in all tables**
  - **uid**: A user identification key that is unique for each user.
  - **event_timestamp**: Unix timestamp of the event
  - **dt**: date of the event
  - **s_index**: Incremental session index. Starts from 1 for each user and increases by 1 for each new session. A session is defined as the period of time between game launch and close.
  - **s_time**: The exact time of session (in seconds) the event took place.
  - **app_version**: Version of the game when the event was sent.
  - **platform**: Platform of the game (ios/android)
  - **gold_amount:** amount of gold (soft currency) users have at the time of the event.
  - **gem_amount:** amount of gem (hard currency) users have at the time of the event.
  - **stone_amount:** amount of stone (soft currency) users have at the time of the event.
  - **chalice_amount:** amount of chalice (level currency) users have at the time of the event.

- **Endgame** (Sends whenever a game round is finished.)
  - **Type**:
    - **success**: if player successfully finishes the level
    - **leave**: if player leaves without completing the level
    - **lose**: if player is loses the level
    - **camp**: if level is camp **for the new users as "trial/training"?**
    - **restart**: if level is restarted
  - **level**: id of the level
  - **Level_index**: the number of times the user played this specific level before, including this time. e.g. if this is the 5th time the user has started this level, this should be 5.
  - **Duration**: the duration of the level. for the camp, this counter should reset every time the user leaves camp.
  - **score**: the score of the game that has finished
  - **merge_count**: the number of merges made during a level. combos are also counted here.
  - **moves_count**: the number of times the user changed the position of an object
  - **coin_earned**: number of coins earned during level
  - **gem_earned**: number of gems earned during level

- **Startgame** (Sends whenever a game round has started.)

- **Type**:
  - **camp**: if player successfully starts camp level
  - **newg: if player successfully starts other levels**
- **level**: id of the level
- **Level_index**: the number of times the user played this specific level before, including this time. e.g. if this is the 5th time the user has started this level, this should be 5.

- **Install** (Sends whenever the user opens the application after an install)
  - **Country:** country of the user
  - **Network:** network where the user is acquired

- **Login** (Sends every time the user starts the application)

- **Revenue** (Sends whenever the user makes an in-app purchase)
  - **Currency:** currency of the purchase
  - **revenue_usd**: USD amount the user paid for the item

- **Econ Spend HC** (Sends whenever the user makes a purchase in the game with hard currency - gem)
  - **Currency**: currency type, gem
  - **Type**: type of the spend
  - **Level**: id of the level.
  - **Amount**: the number of currency spent.

- **Cost** (Cost table about user acquisition)
  - **Country**: country of the acquired users
  - **Network**: the source of the acquisition
  - **Mapped Platform**: Platform for the acquisition
  - **Cost**: amount of money in USD currency spent for user acquisition.

Task 2:

- **A/B test state**
  - **user_id**: unique id of a user **(also in other tables)**
  - **platform**: platform they played on
  - **variant**: their a/b test variant
  - **test_entry_date**: date when they joined the test
  - **install_date**: date when they installed the game

- **Login** (Sends every time the user starts the application)
  - **login_date**: date of a login
  - **d_power**: how much dragon power they had

- **Revenue** (Sends whenever the user makes an in-app purchase)

- **purchase_date:** date when the purchase was made.
- **purchase_timestamp:** unix timestamp when the purchase was made.
- **purchase_usd_amount**: USD amount the user paid for the item

- **Economy Earn (Sends with each currency gain) — similar for Economy Spend**
  - **currency**: currency type: coin, gem, etc.
  - **earn_context** (**spend_context**): how and where the transaction was made
  - **amount**: the number of currency earned.

- **Duration (measures time spent in different game areas)**
  - **total_duration**: overall time spent in game, sum of other parts
  - **camp_duration**: time spent in the main game area, camp
  - **level_duration**: time spent playing levels
  - **event_duration**: time spent playing live events (liveops)

- **Feature Metrics - Wonder Crafters Funnel** (Triggers when player interacts with any of the crafters at the wonder crafter beacon)
  - **wonder_list**: all craftable wonders player have, JSON format like: {prefab_id_1: amount, prefab_id_2: amount}
  - **step**:
    - **visible**: when the beacon becomes visible for the very first time (this event will only send once when the player first sees the beacon)
    - **locked**: when player clicks on the locked beacon
    - **unlock**: when player unlocks wonder crafter beacon
    - **close_beacon**: when player dismisses and enters in the unlocked wonder crafter beacon by clicking on the beacon
    - **open_wonder**: when player clicks and enters in the unlocked wonder crafter beacon by clicking on the beacon
    - **close**: when player exits from the unlocked wonder crafter beacon
    - **crafter_transform**: when player transforms one of their wonders into a crafter
    - **crafter_upgraded_normal**: when player upgrades their crafter via completing mission
    - **crafter_upgraded_premium**: when player upgrades their crafter via gems
    - **recipe_unlocked**: when player unlocks a recipe (sends separately for each recipe unlocked) - may not be needed
    - **recipe_selected**: when player adds recipe to one of their slots (when begin crafting)
    - **recipe_slot_upgraded**: when player upgrades a recipe - may not be needed
    - **ingredient_added**: when player adds ingredient into the recipe from camp
    - **crafting_start**: when player starts crafting

        - **crafting_skipped**: when player spends gems to skip crafting process
        - **crafting_quit**: when player crafting ends
        - **item_collected**: when player collects the crafted item
        - **daily_currency_collected**: when player collects daily currency from the beacon
        - **currency_purchased**: when player purchases more crafting currency
        - **purchase_missing_ingredient**: when player purchases missing ingredient through gem
        - **crafter_mission_complete**: when player completes a crafter mission
        - **crafter_mission_start**: when player sends gem to skip crafter mission
        - **crafter_mission_skip**: when player receives crafter mission reward
    - **recipe_slots**: recipes in slots, JSON format like: {slot_1: recipe_identifier, slot_2: recipe_identifier}
    - **crafter_level_id**: prefab_id of the crafter
    - **crafter_level**: level of the crafter
    - **recipe_id**: recipe_identifier of the interacted recipe
    - **item_id**: prefab_id of the ingredient or the crafted item in the result
    - **item_amount**: amount of the item given or collected