Data Engineering Challenge

Nordeus - Job Fair 2026

About the Challenge

At Nordeus sports spirit is something that connects us all. The sport that you play depends on the team you are part of: Top Eleven and Top Goal teams enjoy football, and the Golf Rival team plays golf. Now it's your turn to join golf lovers, at least from a data perspective.

Your task is to prepare the data for Golf Rival. Luckily, players have already spent weeks playing, all while generously producing the data needed to improve their experience.

You are given as input a dataset (JSONL format) where each line is a valid JSON object that represents an event generated by a game. Each event emitted from the game has several fields/parameters that describe the action happening inside the game.

Data description

Raw events (events.jsonl)

Parameter Name	Parameter Type	Parameter Description
id	INT	Unique identifier representing an event
timestamp	INT	Time of event represented as Unix time
event_type	STRING	One of the following: registration, session_ping, match_start, match_finish
user_id	STRING	Unique identifier representing a user
event_data	JSON OBJECT	JSON object containing all event-specific data (check event data below)

For our analysis, we are interested in four types of events: registration, session_ping, match_start, match_finish.

1. Registration (registration)

A registration event is generated when a user creates their account in Golf Rival. At that time, they get a unique identifier that is used for all other events that come from

their account, as a way to keep track of them. This event also contains the username that the user creates, the country they registered from, and the OS of the device they registered from.

Required event_data for **registration**

Parameter Name	Parameter Type	Parameter Description
country	STRING	Country code of the country user registered from
device_os	STRING	OS of the device user registered from (Valid values: iOS or Android)
username	STRING	In-game nickname

2. Session Ping (session_ping)

A session is the continuous period during which a player interacts with a game. A user session is one continuous gameplay period, starting when the player opens or enters the game and ending when they leave or become inactive. We track sessions using session_ping events. This event is sent when a user opens the game, and every 120 seconds after that, as long as the user has the game open. If more than 120 seconds pass between two ping events, it means they belong to different sessions. The first event within a session has the value started in the state column, while the last event is with status ended. These two types are the only ones you need to process if you are using the state column. Another possible value is in_progress, which, as the name suggests, is logged when a session_ping occurs in between started and ended session pings.

Required event_data for **session_ping**

Parameter Name	Parameter Type	Parameter Description
state	STRING	Valid values: “started”, “in_progress”, “ended”
device_os	STRING	OS of the device user pinged his session from (Valid values: iOS or Android)

3. Match Start (match_start)

This event is sent when the user enters a match with a player with ID opponent_id (two events per match, one for each user). Each match is played on a map with ID map_id (see maps.jsonl).

Required event_data for **match_start**

Parameter Name	Parameter Type	Parameter Description
map_id	STRING	Unique identifier representing a map

opponent_id	STRING	user_id of the opponent player

4. Match Finish (match_finish)

Once the match finishes, the match_finish event is sent from users' devices (two events per match, one for each user). This event stores the map_id previously mentioned and the outcome of the match, which is 1 if the user won, 0.5 if it's a draw, and 0 if the user sending the event lost the match.

Required event_data for **match_finish**

Parameter Name	Parameter Type	Parameter Description
map_id	STRING	Unique identifier representing a map
opponent_id	STRING	user_id of the opponent player
outcome	FLOAT	Outcome of the match (1 if win, 0.5 if draw, 0 if loss)

Note: Due to unreliable event delivery, some matches may contain only a partial set of expected events. Consider a match valid if, for the two participating players, the available events collectively include at least one match_start and at least one match_finish, regardless of which player emitted which one.

Maps (maps.jsonl)

maps.jsonl contains basic info about maps (golf courses).

Parameter Name	Parameter Type	Parameter Description
map_id	STRING	Unique identifier representing a map
map_name	STRING	Name of the map

Tasks

Data cleaning

- Discard duplicate events. In these cases, keep the earliest event.
- Discard invalid events. Events are considered invalid if they do not meet the requirements specified above.
- Use common sense to discard other events that don't make any sense.

API

- GET /user-stats

- Input:
  - Countries (list, optional): list of country codes for which to return player statistics
  - OSs (list, optional): A list of operating systems for which to return player statistics (use the session's OS for this). Keep in mind that a user may register on one OS but play sessions on another; this filter should apply only to statistics achieved during sessions played on the specified OS.
- Output:
  - List of entries with fields username, country, fav_map, fav_map_win_ratio, total_playtime, total_win_ratio, avg_matches_per_session, registration_date, ordered by total_playtime (descending).
    - Favorite map and favorite map win ratio (fav_map and fav_map_win_ratio): A favorite map is a map on which the user has the highest win ratio.
    - total_playtime is the time spent playing Golf Rival in seconds
- Example user-stats:

	username	country	fav_map	fav_map_win_ratio	total_playtime	total_win_ratio	avg_matches_per_session	registration_date
1.	Nikola	SRB	Lake	0.6	153226	0.5	0.19	2022-04-03
2.	Mihailo	MNE	Inferno	1	22135	0.6	1.0	2025-11-11
3.	Dejan	SRB	Cobblestone	0.8	8800	0.7	0.8	2018-02-01

- GET /map-stats/{map-name}
  - Input:
    - map-name - name of the map to return stats for
    - date_from and date_to in format YYYY-MM-DD (both optional, if one or both boundaries are not provided, treat it as unbounded on these sides)
  - Output:

        - List of entries with fields date, avg_playtime, best_player_username, match_cnt ordered by date descending.
            - date is the date the data in the row is relevant for.
            - avg_playtime is the average match duration on that map in seconds on a date.
            - best_player_username is the username of the player with the highest win ratio on a map up to and including a given day (date), calculated as that player's total wins on the map up to that day divided by their total matches played on the map up to that day.
            - match_cnt is the number of matches that happened on a map on a date.
            - Consider that the match happened on some date if it ended on that date.
    - Example map-stats

	date	avg_playtime	best_player_username	match_cnt
1.	2026-04-03	130	Nikola	221
2.	2026-04-02	180	Nikola	225
3.	2026-04-01	150	Dejan	196

Expectations

The input file should be processed only once and the target data model should be somehow persisted (in-memory, file, database, etc.) so that queries can be executed efficiently. It is not recommended to process the input file each time a query is executed. You have the freedom to implement the API in any way you find appropriate and in any language (Python, Java, SQL, etc.) for example:

- REST API (recommended) implemented in any language
- CLI APP implemented in any language
- Other

One important thing here is that you should add a documentation file (preferably README.md). This document should explain your chosen approach for the project

and provide clear, user-friendly instructions on how to use the application, including how to start it and install any required dependencies.

Bonus points

- Implementing a detailed and careful data-cleaning process.
- Storing the data model in SQL database and using SQL to query data.
- Implementing the solution as a well-documented REST API.
- Track sessions without using the column state.
- Visualize the match count from the map-stats API route over the last 7 days as a line chart over time (by date), where each map's statistics are represented by a separate line. You can use any framework or library you like.

Submission

- Upload the solution to a new public repository in your **GitHub/GitLab/Bitbucket** profile. Refrain from naming your repository like "nordeus-data-engineering-challenge" ❌so that it's not easily discoverable online (if you want, you can also keep your repository private until the submission deadline, but don't forget to make it public once the submissions are closed).
- Send an email to jobfair@nordeus.com with a link to your repository with the subject **Data Engineering Challenge** and please add your **full name** in the email as well as the link to your **LinkedIn profile**, if you have it.

Advice

- Rely on your best judgment! **We don't expect you to catch all the edge cases, but try to handle as many as possible.** :)
- If you're stuck, feel free to ask for clarification or help by emailing us at jobfair@nordeus.com
- If something in the problem seems unclear, state your assumptions explicitly.
- We encourage you to try out the challenge! It's better to send an incomplete solution than not to send one at all! 🙂

The challenge is open until May 3rd, 2026, end of the day. Good luck!