DataOps Tech Test: Tennis

Contents

Test Overview

This test is a project made up of functional and non-functional requirements designed to allow you to explore a problem likely to be faced as part of the role and to demonstrate experience and skills that you can bring to the role.

This project is not supposed to be comprehensively completed as part of the interview process, however please take on as much as you can within a reasonable time limit.

The technical stage interview will be approx. 90 mins and you will be expected to present your (partially) implemented solution and describe /discuss how you would take on the remaining requirements.

Explanation of source data

The source data (`keystrokes-for-tech-test.csv`) is a timeline of JSON events which describe a selection of tennis matches (Men's Singles, Professional) with three fields:

- `match_id` is a unique identifier for a match
- `message_id` is a unique identifier for an event in a specific match and also defines the sequential order of the events
- `match_element` is the JSON payload of the event, with a different schema dependent on the event type (defined by `.eventElementType` field within the payload)

Functional Requirements

Requirement: Clean & flatten the data

The current structure of the data has a large number of possible JSON paths and a large number of rows containing state which could be inter-linked. For example a single score increment in the match could result in the following 7 events (event names simplified):

1. Point Started
2. Let
3. Point Started
4. Fault
5. Physio
6. Point Started
7. Point Scored

The first requirement is to flatten the data, pivoting on the initiation of a serve (signified by a `PointStarted` event type) and one-hot encoding the state of the match prior to that serve being initiated (for example a physio being called) and the outcome of that serve being hit (either an unsuccessful serve or a point being played out). The result could look like this (derived from the above 7 events):

.

.

.

state of the world before serve		serveId	outcome of the serve			
server	physio		let	fault	player A scored	player B scored
playerA	0	1	1	0	0	0
playerA	0	2	0	1	0	0
playerA	1	3	0	0	1	0

Requirement: Enrichment

The current data does not contain information on whether the serve being initiated is a first or a second serve. The second requirement is to add this as a column to the flattened data.

Requirement: Transformation

Each PointScored event *should* contain the total number of previously won sets from the fields `.score.overallSetScore.setsA` (for playerA) and `.score.overallSetScore.setsA`. However, this is missing from a selection of the matches. The third requirement is to repair this field by calculating these values from the field which does exist called `.score.previousSetsScore`

Requirement: R&D

Find a way to enrich the data (either with the available JSON data or an external source) by adding a feature which would be interesting to an end user. This could be implemented into the application or presented as an exploratory analysis report.

Non-Functional Requirements

Requirement: Streamable

The application should be able to consume and process the source data one event at a time, with a reasonably low latency. The end user will be watching the sport live and expect the product that this data drives to be approx. up to date with what is occurring on a broadcasted stream.

Requirement: Cloud Architecture

The application should be hosted on an appropriate platform with:

- relevant infrastructure described as code
- an accompanying diagram
- documented decisions as to why components were chosen

Requirement: Data contract

The source data structure should be described and enforced by the application and any new events arriving with a variance in schema should be dismissed and appropriately flagged so the data provider can be invited to take action / fix. A programatically consumable model of the output data should be written so that downstream consumers can have a schema to enforce. A business friendly description of what is happening to the data within the application should be documented.

Requirement: Development Flow

The application should be written with a wider development team in mind, taking care to structure the project in a way which enables code formatting and local builds to remain consistent for multiple developers. Functional requirements should be appropriately described using a testing strategy of your choosing. A continuous integration mechanism should be in place.