HRG Data Engineer Test Assignment

In online gaming, every spin, purchase, and login generates valuable data, creating a complex stream of events. Capturing and analyzing these structured JSON events is key to understanding player behavior and improving monetization strategies.

Below, I describe the optimal data architecture to efficiently store and process user interaction data.

Data Processing

Technologies Used:

- **Apache Kafka** for event streaming
- **Apache Spark** for batch processing

Kafka Streaming captures raw events in real-time, but these events may arrive out of order or contain errors. In such cases, batch processing (via Spark) helps clean, deduplicate, and transform data for long-term storage. While Kafka handles simple real-time calculations, deeper insights (e.g., player retention, lifetime value) require batch aggregation.

Additionally, if an issue occurs in the streaming pipeline, batch jobs can reprocess old data and fill in missing values.

Data Storage

Since our data arrives as JSON, the best storage solution depends on querying requirements, schema evolution, and storage format. The architecture consists of three layers:

Raw Storage Layer (Landing Zone)

- Store raw JSON in **Apache Iceberg** or **Delta Lake**

- These formats support **schema evolution, ACID transactions, and efficient query optimization**

Processed Storage Layer (Data Vault 2.0)

- Transform JSON into structured tables using **Apache Spark**

- Schema mapping (JSON fields → Data Vault tables) ensures **normalized storage**

- dbt can be used for further data modeling and transformations within the analytical database

Query & Analytics Layer

- Load transformed data into **Snowflake**

- Snowflake is an industry-leading **cloud-based database** widely used for implementing Data Vault models due to its **scalability, flexibility, and support for large-scale changing data**

Data Vault 2.0 Model

This model ensures **auditability, scalability, and flexibility** in handling event-driven architectures.

1. Hub Tables (Unique Business Entities)

Hub tables store unique business keys and serve as the foundation of the model.

Hub_Player

This hub captures unique players in the gaming system.

Column	Data Type	Description
player_id	BIGINT	Unique identifier for the player (uid in JSON)
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

Hub_Game

This hub represents unique games within the ecosystem.

Column	Data Type	Description
game_id	VARCHAR	Unique identifier for the game (mapped from app in JSON)
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

Hub_Transaction

This hub captures all unique transactions, whether they are authentications, spins, or purchases.

Column	Data Type	Description
transaction_id	BIGINT	Unique transaction identifier (mapped from msg_id in JSON)
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

2. Link Tables (Relationships Between Entities)

Link tables establish relationships between different business entities.

Link_Player_Game

Column	Data Type	Description
link_id	BIGINT	Unique link identifier
player_id	BIGINT	FK to Hub_Player
game_id	VARCHAR	FK to Hub_Game
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

Link_Transaction_Game

Column	Data Type	Description
link_id	BIGINT	Unique link identifier
transaction_id	BIGINT	FK to Hub_Transaction
game_id	VARCHAR	FK to Hub_Game
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

Link_Transaction_Player

Column	Data Type	Description
link_id	BIGINT	Unique link identifier
player_id	BIGINT	FK to Hub_Player
transaction_id	BIGINT	FK to Hub_Transaction
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

3. Satellite Tables (Attributes That Change Over Time)

Satellite tables store evolving attributes related to business keys.

Sat_Player_Details

Stores **player-related details** that can change over time (e.g., email, phone).

Column	Data Type	Description
player_id	BIGINT	FK to Hub_Player
email	VARCHAR	Player's email address (from JSON)
phone_number	VARCHAR	Player's phone number (from JSON)
last_active_game	VARCHAR	The last game played by the player
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

Sat_Game_Stats

Stores **aggregated stats per game** (total spins, purchases, etc.).

Column	Data Type	Description
game_id	VARCHAR	FK to Hub_Game
total_spins	INTEGER	Total spins in the game
total_purchases	FLOAT	Total purchase value in the game
load_timestamp	TIMESTAMP	Record load timestamp
Record_source	VARCHAR	Source system name

Sat_Transaction_Details

Column	Data Type	Description
transaction	BIGINT	FK to Hub_Transaction
transaction_type	VARCHAR	Type of transaction (auth, spin, purchase) (type in JSON)
amount	FLOAT	Purchase amount (if applicable, amount in JSON)
game_id	VARCHAR	FK to Hub_Game
load_timestamp	TIMESTAMP	Record load timestamp
record_source	VARCHAR	Source system name

The proposed **Data Vault 2.0 model** provides a **scalable, auditable, and high-performance storage solution** for Company X's event-driven architecture. By implementing this model, the organization will efficiently aggregate and analyze gaming transaction data for **business intelligence and monetization strategies**.