HRG Data Engineer Test Assignment

Task:

1. **What technology/technologies will be used to implement this storage solution**

Technologies for Storage Solution Implementation

1. Cloud Platform: Microsoft Azure

- **Why chosen**:
  - Provides robust infrastructure for ingestion, processing, and storage.
  - Offers seamless integration with Databricks, Event Hubs, and Snowflake, making it an ideal choice for this architecture.

2. Storage Technologies

Snowflake

- **Why chosen**:
  - Scalable, cloud-native data warehouse designed for high-performance analytics.
  - Optimized for structured data storage.
- **Usage**:
  - Store the structured Data Vault 2.0 model (Hubs, Links, Satellites).
  - Enable time-travel for historical data analysis.
  - Support SQL-based analytics and integration with BI tools.

Azure Data Lake

- **Why chosen**:
  - Cost-effective storage for raw, semi-structured data.
- **Usage**:
  - Store raw JSON events (auth, spins, purchase) for archival and replay purposes.
  - Serve as the staging area for data preprocessing.

3. Data Ingestion Technologies

Azure Event Hub

- **Why chosen**:
  - Highly scalable, real-time data ingestion service for streaming data.
- **Usage**:
  - Ingest events from Appsflyer, Firebase, and backend services.
  - Stream events into Databricks for real-time processing.

4. Data Processing Technologies

Azure Databricks

- **Why chosen**:
  - Unified data engineering platform with Spark-based distributed computing.
- **Usage**:
  - Transform raw JSON data into structured data compliant with Data Vault 2.0 model.
  - Perform batch and real-time processing of event streams.

5. Data Governance and Lineage

Azure Purview

- **Why chosen**:
  - Comprehensive data governance and lineage tracking solution.
- **Usage**:
  - Classify sensitive data (email, phone numbers).
  - Track data lineage from ingestion to analytics for PIPEDA/GDPR compliance.

6. Orchestration and Automation

Azure Data Factory

- **Why chosen**:
  - Orchestrates data movement and transformation workflows.
- **Usage**:
  - Automate pipeline for moving data from Data Lake to Snowflake.
  - Schedule data processing tasks (ETL/ELT workflows).

Azure DevOps

- **Why chosen**:
  - Supports CI/CD pipelines for deployment and automation.
- **Usage**:
  - Automate version control and deployment of Snowflake schema and Databricks notebooks.

7. Analytics and BI Tools

Tableau or Looker Studio

- **Why chosen**:
  - Advanced visualization tools for interactive reports and dashboards.
- **Usage**:
  - Connect to Snowflake for aggregated data analysis and reporting.

8. Monitoring and Logging

Azure Monitor

- **Why chosen**:
  - Provides insights into resource performance and issues.
- **Usage**:
  - Monitor data pipelines, storage usage, and processing performance.

Key Integrations

- **Appsflyer**: Provides event data that integrates with Azure Event Hub for ingestion.
- **Snowflake and Azure Databricks**: Ensure seamless processing and querying of data.

This technology stack forms a scalable, efficient, and compliant storage solution that meets all specified requirements.

---

2. **Describe the Table Structure, Attribute Composition, and Data Types**

Here is a detailed Data Vault 2.0 database structure, designed based on the provided message examples (Authorization, Spins, and Purchase events) with a description of table structure, attribute composition, and data types.

Database Structure

The Data Vault 2.0 model is implemented using Hubs, Links, and Satellites. Each table includes attributes for tracking metadata (*LoadDate, RecordSource*) and ensures historical tracking and scalability.

1. Hubs

Hubs capture unique business keys (immutable) for core business entities.

Hub_Player

Attribute	**Data Type**	**Description**
PlayerID	VARCHAR(10)	Unique identifier for the player (uid).
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

Hub_Game

Attribute	**Data Type**	**Description**
GameID	VARCHAR(10)	Unique identifier for the game (app).
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

2. Links

Links capture relationships between Hubs (e.g., players and games).

Link_Player_Game

Attribute	**Data Type**	**Description**
PlayerID	VARCHAR(10)	Foreign key referencing Hub_Player.
GameID	VARCHAR(10)	Foreign key referencing Hub_Game.
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

3. Satellites

Satellites store descriptive data and track historical changes for Hubs and Links. They include event-specific data such as Authorization, Spins, and Purchase.

Sat_Player_Auth

Attribute	Data Type	Description
PlayerID	VARCHAR(10)	Foreign key referencing Hub_Player.
Email	VARCHAR(50)	Email address of the player.
Phone	VARCHAR(15)	Phone number of the player.
GameID	VARCHAR(10)	Foreign key referencing Hub_Game.
PublishTimestamp	DATETIME	Event generation timestamp from the payload.
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

Sat_Player_Spin

Attribute	Data Type	Description
PlayerID	VARCHAR(10)	Foreign key referencing Hub_Player.
GameID	VARCHAR(10)	Foreign key referencing Hub_Game.
SpinValue	INT	The value of the spin.
PublishTimestamp	DATETIME	Event generation timestamp from the payload.
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

Sat_Player_Purchase

Attribute	Data Type	Description
PlayerID	VARCHAR(10)	Foreign key referencing Hub_Player.
GameID	VARCHAR(10)	Foreign key referencing Hub_Game.
PurchaseAmount	DECIMAL(10,2)	The amount spent by the player.
PublishTimestamp	DATETIME	Event generation timestamp from the payload.
RecordSource	VARCHAR(50)	Source system of the data (e.g., "EventHub").
LoadDate	DATETIME	Timestamp when the record was loaded.

Metadata Attributes

Every table includes:

- Tracks when the data was loaded into the database.
- Tracks the origin of the data for auditability (e.g., "EventHub").

Examples Using the Provided JSON

Example 1: Authorization Event (**auth_msg**)

JSON:

```
{
  "msg_id": 124,
  "publish_ts": "2024-10-12T14:00:00",
  "type": "auth_event",
  "payload": {
    "uid": 453135,
    "email": "SomeEmail@test.com",
    "phone": null,
    "app": "app_3"
  }
}
```

Table Data:

- **Hub_Player**:
  - **PlayerID**: 453135
  - **RecordSource**: EventHub
  - **LoadDate**: Current timestamp
- **Hub_Game**:
  - **GameID**: app_3
  - **RecordSource**: EventHub
  - **LoadDate**: Current timestamp
- **Sat_Player_Auth**:
  - **PlayerID**: 453135
  - **Email**: SomeEmail@test.com
  - **Phone**: NULL
  - **GameID**: app_3
  - **PublishTimestamp**: 2024-10-12T14:00:00
  - **RecordSource**: EventHub
  - **LoadDate**: Current timestamp

Example 2: Purchase Event (**purchase_msg**):

JSON:

```
{
  "msg_id": 2117,
  "publish_ts": "2024-10-12T17:19:00",
  "type": "purchase_event",
  "payload": {
    "uid": "some_uid_3",
   "amount": 1799,
   "app": "app_5"
  }
}
```

Table Data:

- **Hub_Player**:
    - **PlayerID**: some_uid_3
    - **RecordSource**: EventHub
    - **LoadDate**: Current timestamp
- **Hub_Game**:
    - **GameID**: app_5
    - **RecordSource**: EventHub
    - **LoadDate**: Current timestamp
- **Sat_Player_Purchase**:
    - **PlayerID**: some_uid_3
    - **GameID**: app_5
    - **PurchaseAmount**: 1799
    - **PublishTimestamp**: 2024-10-12T17:19:00
    - **RecordSource**: EventHub
    - **LoadDate**: Current timestamp

Example 3: Spin Event (**spins_msg**)

JSON:

```
{
"msg_id": 1275,
"publish_ts": "2024-10-12T14:02:00",
"type": "spin_event",
"payload": {
"uid": 125331,
"spin": 1400,
"app": "app_3"
 }
}
```

Table Data:

- **Hub_Player**
    - **PlayerID**: 125331
    - **RecordSource**: EventHub
    - **LoadDate**: Current timestamp

- **Hub_Game**

    - **GameID**: app_3
    - **RecordSource**: EventHub
    - **LoadDate**: Current timestamp

- **Sat_Player_Spin**
  - **PlayerID**: 125331
  - **GameID**: app_3
  - **SpinValue**: 1400
  - **PublishTimestamp**: 2024-10-12T14:02:00
  - **RecordSource**: EventHub
  - **LoadDate**: Current timestamp

---

3. **What Additional Components Need to Be Developed to Support Your Solution?**

To fully support the proposed Data Vault 2.0 solution, the following additional components need to be developed:

1. **Data Ingestion Layer**

Ingestion Pipelines:

- **Purpose**: Collect events (e.g., auth_msg, spins_msg, purchase_msg) from external systems such as Appsflyer and Firebase.
- **Technology**: Use Azure Event Hub or Azure Data Factory.
- **Functionality**:
  - Connect to event sources (Appsflyer, Firebase).
  - Capture and push raw JSON data into an Azure Data Lake for staging.
- **Example**: Create pipelines for real-time streaming (Event Hub) or scheduled batch ingestion (Data Factory).

Schema Evolution and Validation:

- **Purpose**: Validate incoming JSON events against predefined schemas to ensure consistency.
- **Technology**: Use Azure Databricks.
- **Functionality**:
  - Automatically handle schema changes (e.g., new attributes in JSON payloads).

2. Data Processing and Transformation Layer

Transformation Workflows:

- **Purpose**: Transform raw JSON data into structured Data Vault 2.0 tables.
- **Technology**: Use Azure Databricks.
- **Functionality**:
  - Parse and normalize JSON payloads.
  - Populate Hubs, Links, and Satellites in Snowflake.

Event Deduplication:

- **Purpose**: Ensure unique events are processed (e.g., avoid duplicate msg_id records).
- **Technology**: Built into Databricks processing scripts.
- **Functionality**:
  - Use primary keys (msg_id, uid, app) to filter out duplicates.

3. Data Storage and Querying

Data Lake for Raw Data:

- **Purpose**: Store raw JSON messages for archival and replay purposes.
- **Technology**: Azure Data Lake Storage.
- **Functionality**:
  - Organize raw data by event type (auth_msg, spins_msg, purchase_msg) and ingestion time.
  - Serve as a backup source for reprocessing.

Data Warehouse:

- **Purpose**: Store processed and structured data in Snowflake.
- **Technology**: Snowflake Cloud Data Platform.
- **Functionality**:
  - Enable querying using SQL.
  - Support analytics dashboards and BI tools.

4. Orchestration and Automation

Pipeline Orchestration:

- **Purpose**: Automate data ingestion, transformation, and loading processes.
- **Technology**: Use Azure Data Factory.
- **Functionality**:
  - Schedule ingestion pipelines (e.g., every 5 minutes for real-time data).
  - Monitor pipeline execution and trigger retries in case of failures.

CI/CD Pipelines:

- **Purpose**: Automate deployment of infrastructure, Databricks jobs, and Snowflake schema.
- **Technology**: Use Azure DevOps or GitHub Actions.
- **Functionality**:
  - Version control for Databricks notebooks, Snowflake schema scripts, and ETL workflows.
  - Automate testing and deployment to production.

5. Monitoring and Logging

Pipeline Monitoring:

- **Purpose**: Track data pipeline performance and detect failures.
- **Technology**: Use Azure Monitor or Databricks Dashboards.
- **Functionality**:
  - Monitor Event Hub lag and ingestion rates.
  - Track job execution times in Databricks.

Data Quality Monitoring:

- **Purpose**: Ensure data accuracy and completeness across pipelines.
- **Technology**: Use Great Expectations or custom validation scripts in Databricks.
- **Functionality**:
  - Validate data against predefined business rules (e.g., ensure no NULL values in critical fields).

6. **Data Governance and Security**

Data Governance Framework:

- **Purpose**: Ensure compliance with PIPEDA/GDPR and maintain data lineage.
- **Technology**: Use Azure Purview.
- **Functionality**:
  - Classify PII data (email, phone).
  - Track lineage from raw JSON data to BI reports.

Access Control:

- **Purpose**: Enforce role-based access to sensitive data.
- **Technology**: Use Snowflake's access control features and Azure AD for identity management.
- **Functionality**:
  - Restrict access to PII data at the column level.

7. Analytics and Visualization

BI Dashboards:

- **Purpose**: Provide insights into aggregated metrics (e.g., average purchases, spins trends, time spent on games).
- **Technology**: Use Tableau or Looker Studio.
- **Functionality**:
  - Connect to Snowflake for live data.
  - Create real-time dashboards for business teams.

8. **Real-Time Analytics**

Streaming Analytics:

- **Purpose**: Support real-time analytics for critical metrics (e.g., purchases, spins).
- **Technology**: Use Snowflake Streams and Tasks.
- **Functionality**:
    - Automatically update aggregated tables and dashboards as new events are processed.

9. Data Archival

Purpose:

- Ensure long-term storage of raw and processed data for compliance, audits, and future reprocessing needs.
- Reduce costs by moving older, infrequently accessed data to lower-cost storage solutions.

Technology:

- **Azure Data Lake Storage**: For raw JSON events and intermediate data.
- **Azure Blob Storage (Cool/Archive Tiers)**: For long-term, cost-efficient archival of rarely accessed data.
- **Snowflake Time-Travel and Fail-Safe**: For maintaining historical versions of structured data.

Functionality:

- Use **Azure Storage Lifecycle Policies** to move raw data from hot to archive tiers.
- Export older Snowflake structured data to Azure Blob or Data Lake for long-term storage.
- Enable replay of archived data for future processing or analysis.

Summary of Additional Components

Component	Purpose	Technology
Ingestion Pipelines	Capture and validate JSON events.	Azure Event Hub, Data Factory
Transformation Workflows	Parse, clean, and structure data.	Azure Databricks
Raw Data Storage	Archive JSON messages for reprocessing.	Azure Data Lake
Data Warehouse	Store structured Data Vault 2.0 tables.	Snowflake
Orchestration	Automate ingestion and transformation.	Azure Data Factory
Monitoring	Monitor pipelines and data quality.	Azure Monitor, Great Expectations
Governance	Ensure compliance and track lineage.	Azure Purview
Analytics	Enable real-time insights and reporting.	Tableau, Looker Studio