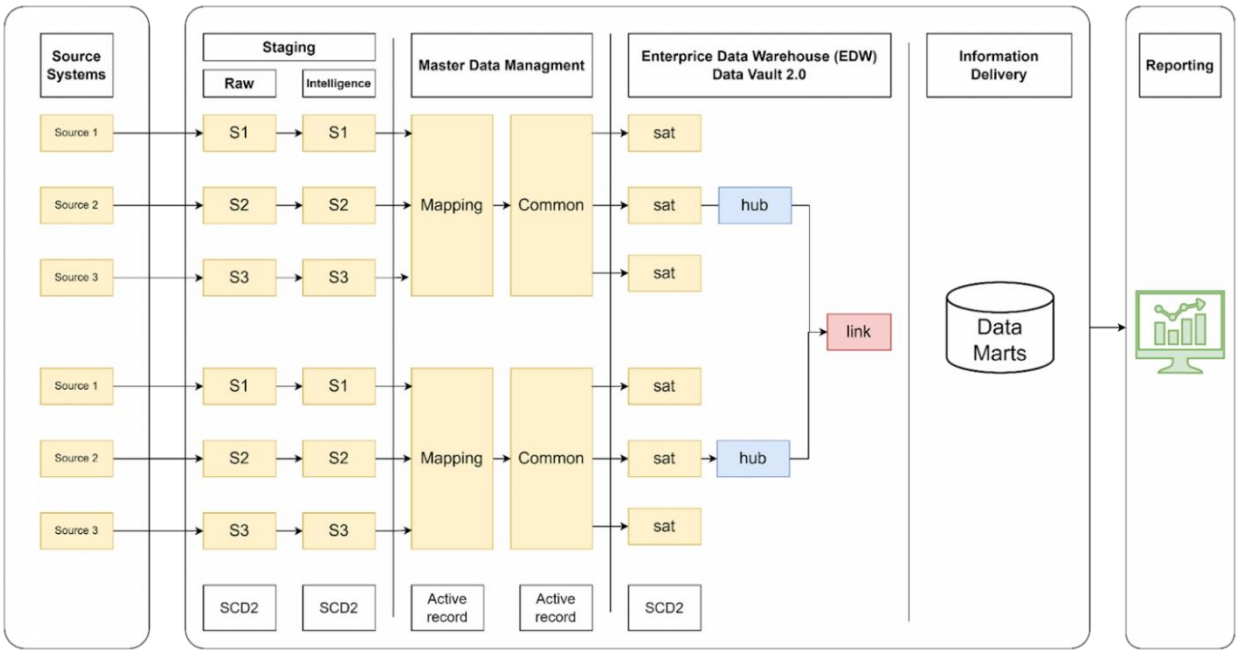
1. **What technology/technologies will be used to implement this storage solution?**

The high-level simplified architecture for Data Vault:



According to the tusk we are interested in layers from Master Data Management to Information Delivery.

The proposed solution

Master Data Management:

1. **Data Storage:**

Cloud Data Warehouses (e.g., Snowflake, BigQuery, or Amazon Redshift)
Reason: Scalability, native support for semi-structured data (JSON), and performance optimization for analytics.

2. Data Ingestion

- *Kafka for event streaming and near real-time data capture.*
- *Apache NiFi or AWS Glue for batch and streaming ETL pipelines.*

3. Data Processing and Transformation

- *Apache Spark on Databricks for large-scale processing and incremental ETL operations.*
- *DBT (Data Build Tool) for managing transformations in the Data Vault model.*

4. Metadata and Orchestration

- *Airflow or Prefect for workflow orchestration.*
- *Data Catalog (AWS Glue Data Catalog or DataHub) for metadata management.*

5. Data Storage Format

JSON stored in a structured or semi structured format for raw event storage.

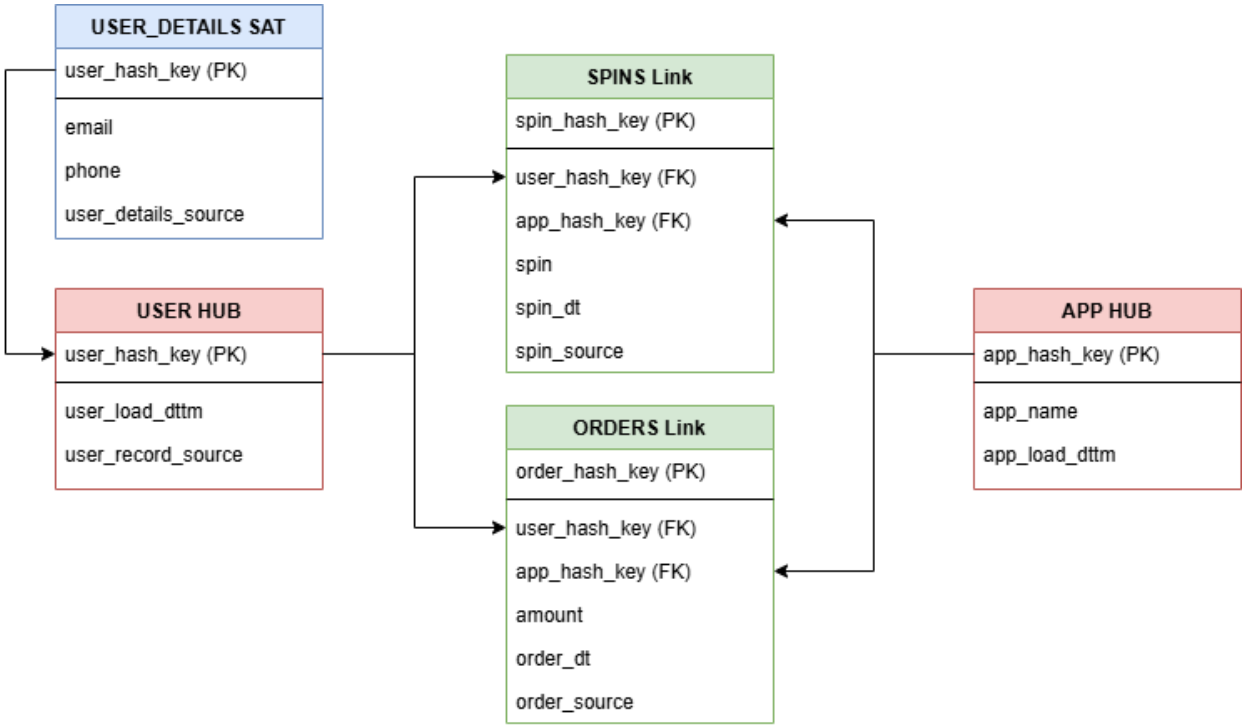
Enterprise Data Warehouse

1. **Data Modeling.** Data Vault 2.0 methodology with staging, raw vault and business vault layers. ETL processes to parse data, generate surrogate keys and separate data into hubs, links and satellites tables.
2. **Aggregation and Enrichment.** Creating business layer, aggregation and enrichment data with metrics and business logic for reporting and analytics
3. **Data Quality and Monitoring.** Duplication checking, ensuring integrity between Data Vaults elements and creating logs

Information delivery

Optimizing business layer for BI tools.
Data transformed into final performance-oriented and user-friendly form.

2. Describe the table structure, attribute composition, and data types. The format of the description is open-ended; use whichever is most convenient or familiar for you.



1. **Hubs**
   Hubs capture unique business keys, which you've implemented correctly for users and apps.

   Hub Table Suggestions:

   - User Hub

Attribute	Data Type	Description
user_hash_key	VARCHAR(255)	PK, Hash of the uid
user_load_dttm	TIMESTAMP	Load timestamp
user_record_source	VARCHAR(255)	Source of the record

- App Hub

Attribute	Data Type	Description
user_hash_key	VARCHAR(255)	PK, Hash of the app
user_load_dttm	TIMESTAMP	Load timestamp
user_record_source	VARCHAR(255)	Source of the record

ETL Example:

```
--Create USER HUB table
DROP TABLE IF EXISTS company_x_test.user;
CREATE TABLE company_x_test.user (
   user_hash_key VARCHAR(255) UNIQUE NOT NULL,
   user_load_dttm timestamp NOT NULL,
   user_record_source VARCHAR(255) NOT NULL
);

ALTER TABLE company_x_test.user company_x_test.auth_msg
ADD CONSTRAINT pk_d_user PRIMARY KEY (user_hash_key);

--Fill USER HUB data from auth_msg source
INSERT INTO company_x_test.user (user_ hash_key, user_load_dttm, user_
record_source)

SELECT DISTINCT
          MD5(uid::varchar) AS user_hash_key
        , CURRENT_TIMESTAMP as user_load_dttm
        , 'auth_msg' user_record_source
FROM company_x_test.auth_msg
WHERE NOT EXISTS
(SELECT 1 FROM company_x_test.user WHERE user_hash_key = MD5(uid::varchar))


--Create APP HUB table
DROP TABLE IF EXISTS company_x_test.app;
CREATE TABLE company_x_test.app (
   app_hash_key VARCHAR(255) UNIQUE NOT NULL,
   app_name text UNIQUE NOT NULL,
   app_load_dttm timestamp NOT NULL
);

ALTER TABLE company_x_test.app
ADD CONSTRAINT pk_d_app PRIMARY KEY (app_hash_key);

--Fill APP HUB data from auth_msg, spins_msh and purchase_msg source
INSERT INTO company_x_test.app (app_hash_key, app_name, app_load_dttm)
SELECT DISTINCT
          MD5(app::varchar) AS app_hash_key
        , app AS app_name
        , CURRENT_TIMESTAMP as app_load_dttm
FROM (
        SELECT app
        FROM company_x_test.auth_msg
        UNION
        SELECT  app
        FROM company_x_test.spins_msg
        UNION
```

```
        SELECT  app
        FROM company_x_test.purchase_msg) sub
WHERE NOT EXISTS
(SELECT 1 FROM company_x_test.app WHERE app_hash_key = MD5(app::varchar))
```

2. Links

Links represent relationships between hubs and are time-variant.

Links Table Suggestions:

- Spin Link

Attribute	Data Type	Description
spin_hash_key	VARCHAR(255)	PK, Hash of the uid
user_hash_key	VARCHAR(255)	FK to user hub
app_hash_key	VARCHAR(255)	FK to app hub
spin	INT	Spin value
spin_dt	TIMESTAMP	Event datetime
spin_source	VARCHAR(255)	Data source

- Order Link

Attribute	Data Type	Description
order_hash_key	VARCHAR(255)	PK, Hash of the uid
user_hash_key	VARCHAR(255)	FK to user hub
app_hash_key	VARCHAR(255)	FK to app hub
amount	INT	Purchase amount
order_dt	TIMESTAMP	Event datetime
order_source	VARCHAR(255)	Data source

ETL Example:

```
--Create SPIN LINK table
DROP TABLE IF EXISTS company_x_test.spins;
CREATE TABLE company_x_test.spins (
        spin_hash_key VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
        user_hash_key VARCHAR(255) NOT NULL,
        app_hash_key VARCHAR(255) NOT NULL,
        spin int NOT NULL,
        spin_dt timestamp NOT NULL,
        spin_source VARCHAR(255) NOT NULL,
        CONSTRAINT fk_user_spins FOREIGN KEY (user_hash_key) REFERENCES
company_x_test.user(user_hash_key),
        CONSTRAINT fk_app_spins FOREIGN KEY (app_hash_key) REFERENCES
company_x_test.app(app_hash_key)
);

--Fill SPIN LINK data from spins_msh
INSERT INTO company_x_test.spins (spin_hash_key, user_hash_key, app_hash_key,
spin, spin_dt, spin_source)
SELECT DISTINCT
          MD5(uid::varchar || app::varchar || publish_ts::varchar) AS spin_hash_key
        , MD5(uid::varchar) AS user_hash_key
        , MD5(app::varchar) AS app_hash_key
        , spin
        , publish_ts spin_dt
        , 'spins_msg ' spin_source
```

```
FROM company_x_test.spins_msg
WHERE NOT EXISTS
(SELECT 1 FROM company_x_test.spins WHERE spin_hash_key = MD5(uid::varchar ||
app::varchar|| publish_ts::varchar))

--Create ORDER LINK table
DROP TABLE IF EXISTS company_x_test.orders;
CREATE TABLE company_x_test.orders (
        order_hash_key VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
        user_hash_key VARCHAR(255) NOT NULL,
        app_hash_key VARCHAR(255) NOT NULL,
        amount int NOT NULL,
        order_dt timestamp NOT NULL,
        order_source VARCHAR(255) NOT NULL,
        CONSTRAINT fk_user_orders FOREIGN KEY (user_hash_key) REFERENCES
company_x_test.user(user_hash_key),
        CONSTRAINT fk_app_orders FOREIGN KEY (app_hash_key) REFERENCES
company_x_test.app(app_hash_key)
);

--Fill ORDER LINK data from purchase_msg source
INSERT INTO company_x_test.orders (order_hash_key, user_hash_key,
app_hash_key, amount, order_dt, order_source)
SELECT DISTINCT
          MD5(uid::varchar || app::varchar || publish_ts::varchar) AS order_hash_key
        , MD5(uid::varchar) AS user_hash_key
        , MD5(app::varchar) AS app_hash_key
        , amount
        , publish_ts order_dt
        , 'purchase_msg ' order_source
FROM company_x_test.purchase_msg
WHERE NOT EXISTS
(SELECT 1 FROM company_x_test.orders WHERE order_hash_key = MD5(uid::varchar
|| app::varchar|| publish_ts::varchar))
```

3. Satellites

Satellites store descriptive information with a direct relationship to a hub or link.

3. Satellites Table Suggestions:

- User_details Satellite

Attribute	Data Type	Description
user_hash_key	VARCHAR(255)	PK, FK, Hash of the uid
email	TEXT	User PII email
phone	text	User PII phone
user_details_source	VARCHAR(255)	Data source

ETL Example:

```
--Create USER_DETAILS SATELLITE
DROP TABLE IF EXISTS company_x_test.user_details;
CREATE TABLE company_x_test.user_details (
        user_hash_key VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
        email text,
        phone text,
```

```
        user_details_source VARCHAR(255) NOT NULL
);

--Fill USER_DETAILS SATELLITE data from purchase_msh
INSERT INTO company_x_test.user_details (user_hash_key, email, phone,
user_details_source)
SELECT DISTINCT
          MD5(uid::varchar) AS user_hash_key
        , email
        , phone
        , 'auth_msg' user_details_source
FROM company_x_test.auth_msg
WHERE NOT EXISTS
(SELECT 1 FROM company_x_test.user_details WHERE user_hash_key =
MD5(uid::varchar))
```

For business layer we're going to aggregate data from tables using joins by hash keys to create and fill views with the data.

3. What additional components need to be developed to support your solution?

In addition to the technologies listed in the first paragraph and for creating qualitative Data Mart layer I would suggest to add some components:

1. Data Quality Framework for checking data integrity, accuracy across all layers, especially the final one. For this purpose it is necessary to create some Data Validation Rules for missing and duplicated data. Then to create data quality dashboard for monitoring data pipline metrics and anomaly.
2. Data Mart Development. Creating Data Mart layer itself with developing ETL process to extract data from Business layer, transform and load it.
3. Reporting and visualizing layer. Use BI tools to create dashboards from Data Mart layer