Vignette 2: シンプルなデータパイプライン (Simple Data Pipelines)

概要

この Vignette では、Snowflake でシンプルかつ自動化されたデータパイプラインを構築する方法を学びます。外部ステージから生の半構造化データを取り込むことから始め、Snowflake のダイナミックテーブルの力を使ってそのデータを変換および強化し、新しいデータが到着すると自動的に最新の状態に保たれるパイプラインを作成します。

学習内容

- 外部 S3 ステージからデータを取り込む方法。
- 半構造化 VARIANT データをクエリおよび変換する方法。
- 配列を解析するために FLATTEN 関数を使用する方法。
- ダイナミックテーブルを作成および連鎖させる方法。
- ELT パイプラインが新しいデータを自動的に処理する方法。
- 有向非巡回グラフ（DAG）を使用してパイプラインを視覚化する方法。

構築するもの

- データ取り込み用の外部ステージ。
- 生データ用のステージングテーブル。
- 3 つの連鎖したダイナミックテーブルを使用したマルチステップデータパイプライン。

SQLコードを取得してSQLファイルに貼り付ける

この ファイル から SQL コードをコピーして、新しい SQL ファイルに貼り付け、Snowflake でフォローしてください。SQL ファイルの最後に到達したら、ステップ 16 - Snowflake Cortex AI に進むことができます。

外部ステージからの取り込み (External Stage Ingestion)

概要

生のメニューデータは現在、Amazon S3 バケットに CSV ファイルとして保存されています。パイプラインを開始するには、まずこのデータを Snowflake に取り込む必要があります。これを行うには、S3 バケットを指すステージを作成し、`COPY` コマンドを使用してデータをステージングテーブルにロードします。

ステップ 1 - コンテキストの設定

まず、正しいデータベース、ロール、ウェアハウスを使用するようにセッションコンテキストを設定しましょう。SQL ファイルの最初のいくつかのクエリを実行します。

```
ALTER SESSION SET query_tag = '{"origin":"sf_sit-is","name":"tb_zts","version":{"major":1,
"minor":1},"attributes":{"is_quickstart":1, "source":"tastybytes", "vignette":
"data_pipeline"}}';

USE DATABASE tb_101;
USE ROLE tb_data_engineer;
USE WAREHOUSE tb_de_wh;
```

ステップ 2 - ステージとステージングテーブルの作成

ステージは、データファイルが保存されている外部の場所を指定する Snowflake オブジェクトです。パブリック S3 バケットを指すステージを作成します。次に、この生データを保持するテーブルを作成します。

```
-- メニューステージを作成する
CREATE OR REPLACE STAGE raw_pos.menu_stage
COMMENT = 'Stage for menu data'
URL = 's3://sfquickstarts/frostbyte_tastybytes/raw_pos/menu/'
FILE_FORMAT = public.csv_ff;

CREATE OR REPLACE TABLE raw_pos.menu_staging
(
    menu_id NUMBER(19,0),
    menu_type_id NUMBER(38,0),
    menu_type VARCHAR(16777216),
    truck_brand_name VARCHAR(16777216),
    menu_item_id NUMBER(38,0),
    menu_item_name VARCHAR(16777216),
    item_category VARCHAR(16777216),
    item_subcategory VARCHAR(16777216),
    cost_of_goods_usd NUMBER(38,4),
    sale_price_usd NUMBER(38,4),
    menu_item_health_metrics_obj VARIANT
);
```

ステップ 3 - ステージングテーブルへのデータのコピー

ステージとテーブルが配置されたので、`COPY INTO` コマンドを使用してステージから `menu_staging` テーブルにデータをロードしましょう。

```
COPY INTO raw_pos.menu_staging
FROM @raw_pos.menu_stage;
```

> **COPY INTO TABLE**: この強力なコマンドは、ステージングされたファイルから Snowflake テーブルにデータをロードします。これは、一括データ取り込みの主要な方法です。

半構造化データ (Semi-Structured Data)

概要

Snowflake は、ネイティブの `VARIANT` データ型を使用して JSON などの半構造化データを処理することに優れています。取り込んだ列の 1 つである `menu_item_health_metrics_obj` には JSON が含まれています。これをクエリする方法を見てみましょう。

ステップ 1 - VARIANT データのクエリ

生の JSON を見てみましょう。ネストされたオブジェクトと配列が含まれていることに注意してください。

```
SELECT menu_item_health_metrics_obj FROM raw_pos.menu_staging;
```

特別な構文を使用して JSON 構造をナビゲートできます。コロン（`:`）は名前でキーにアクセスし、角括弧（`[]`）はインデックスで配列要素にアクセスします。また、`CAST` 関数または二重コロンの省略形（`::`）を使用して、結果を明示的なデータ型にキャストすることもできます。

```
SELECT
    menu_item_name,
    CAST(menu_item_health_metrics_obj:menu_item_id AS INTEGER) AS menu_item_id, -- 'AS' を使用してキャスト
    menu_item_health_metrics_obj:menu_item_health_metrics[0]:ingredients::ARRAY AS ingredients -- 二重コロン (::) 構文を使用してキャスト
FROM raw_pos.menu_staging;
```

ステップ 2 - FLATTEN による配列の解析

`FLATTEN` 関数は、配列のネストを解除するための強力なツールです。これは、配列内の要素ごとに新しい行を生成します。これを使用して、すべてのメニュー項目のすべての成分のリストを作成しましょう。

```
SELECT
    i.value::STRING AS ingredient_name,
    m.menu_item_health_metrics_obj:menu_item_id::INTEGER AS menu_item_id
FROM
    raw_pos.menu_staging m,
    LATERAL FLATTEN(INPUT =>
m.menu_item_health_metrics_obj:menu_item_health_metrics[0]:ingredients::ARRAY) i;
```

> **半構造化データ型**: Snowflake の VARIANT、OBJECT、および ARRAY 型を使用すると、事前に厳密なスキーマを定義しなくても、半構造化データを直接保存およびクエリできます。

ダイナミックテーブル (Dynamic Tables)

概要

フランチャイズは常に新しいメニュー項目を追加しています。この新しいデータを自動的に処理する方法が必要です。このために、クエリの結果を宣言的に定義し、Snowflake に更新を処理させることでデータ変換パイプラインを簡素化するように設計された強力なツールであるダイナミックテーブルを使用できます。

ステップ 1 - 最初のダイナミックテーブルの作成

ステージングテーブルからすべての一意の成分を抽出するダイナミックテーブルを作成することから始めます。`LAG` を「1 minute」に設定します。これは、このテーブルのデータがソースデータより最大でどれだけ遅れることができるかを Snowflake に伝えます。

```
CREATE OR REPLACE DYNAMIC TABLE harmonized.ingredient
    LAG = '1 minute'
    WAREHOUSE = 'TB_DE_WH'
AS
    SELECT
    ingredient_name,
    menu_ids
FROM (
    SELECT DISTINCT
        i.value::STRING AS ingredient_name,
        ARRAY_AGG(m.menu_item_id) AS menu_ids
    FROM
        raw_pos.menu_staging m,
        LATERAL FLATTEN(INPUT =>
menu_item_health_metrics_obj:menu_item_health_metrics[0]:ingredients::ARRAY) i
    GROUP BY i.value::STRING
);
```

> **ダイナミックテーブル**: ダイナミックテーブルは、基になるソースデータが変更されると自動的に更新され、ELT パイプラインを簡素化し、手動の介入や複雑なスケジューリングなしでデータの鮮度を確保します。

ステップ 2 - 自動更新のテスト

自動化が実際に動作しているところを見てみましょう。トラックの 1 つにバインミーサンドイッチが追加されました。これには、フランスパンと大根の酢漬けという新しい成分が含まれています。この新しいメニュー項目をステージングテーブルに挿入しましょう。

```
INSERT INTO raw_pos.menu_staging
SELECT
    10101, 15, 'Sandwiches', 'Better Off Bread', 157, 'Banh Mi', 'Main', 'Cold Option',
9.0, 12.0,
    PARSE_JSON('{"menu_item_health_metrics": [{"ingredients": ["French
Baguette","Mayonnaise","Pickled Daikon","Cucumber","Pork Belly"],"is_dairy_free_flag":
"N","is_gluten_free_flag": "N","is_healthy_flag": "Y","is_nut_free_flag":
"Y"}],"menu_item_id": 157}');
```

次に、`harmonized.ingredient` テーブルをクエリします。1 分以内に、新しい成分が自動的に表示されるはずです。

```
-- 最大1分待ってからこのクエリを再実行する必要がある場合があります
SELECT * FROM harmonized.ingredient
WHERE ingredient_name IN ('French Baguette', 'Pickled Daikon');
```

パイプラインの構築 (Build Out the Pipeline)

概要

これで、他のダイナミックテーブルから読み取るダイナミックテーブルをさらに作成することで、マルチステップパイプラインを構築できます。これにより、チェーン、つまり有向非巡回グラフ (DAG) が作成され、更新はソースから最終出力へと自動的に流れます。

ステップ 1 - ルックアップテーブルの作成

成分をそれらが使用されているメニュー項目にマッピングするルックアップテーブルを作成しましょう。このダイナミックテーブルは、 `harmonized.ingredient` ダイナミックテーブルから読み取ります。

```
CREATE OR REPLACE DYNAMIC TABLE harmonized.ingredient_to_menu_lookup
    LAG = '1 minute'
    WAREHOUSE = 'TB_DE_WH'
AS
SELECT
    i.ingredient_name,
    m.menu_item_health_metrics_obj:menu_item_id::INTEGER AS menu_item_id
FROM
    raw_pos.menu_staging m,
    LATERAL FLATTEN(INPUT =>
m.menu_item_health_metrics_obj:menu_item_health_metrics[0]:ingredients) f
JOIN harmonized.ingredient i ON f.value::STRING = i.ingredient_name;
```

ステップ 2 - トランザクションデータの追加

注文テーブルにレコードを挿入して、バインミーサンドイッチ 2 つの注文をシミュレートしましょう。

```
INSERT INTO raw_pos.order_header
SELECT
    459520441, 15, 1030, 101565, null, 200322900,
    TO_TIMESTAMP_NTZ('08:00:00', 'hh:mi:ss'),
    TO_TIMESTAMP_NTZ('14:00:00', 'hh:mi:ss'),
    null, TO_TIMESTAMP_NTZ('2022-01-27 08:21:08.000'),
    null, 'USD', 14.00, null, null, 14.00;

INSERT INTO raw_pos.order_detail
SELECT
    904745311, 459520441, 157, null, 0, 2, 14.00, 28.00, null;
```

ステップ 3 - 最終パイプラインテーブルの作成

最後に、最終的なダイナミックテーブルを作成しましょう。これは、注文データを成分ルックアップテーブルと結合して、トラックごとの月間成分使用量のサマリーを作成します。このテーブルは他のダイナミックテーブルに依存しており、パイプラインを完成させます。

```
CREATE OR REPLACE DYNAMIC TABLE harmonized.ingredient_usage_by_truck
    LAG = '2 minute'
    WAREHOUSE = 'TB_DE_WH'
    AS
    SELECT
        oh.truck_id,
        EXTRACT(YEAR FROM oh.order_ts) AS order_year,
        MONTH(oh.order_ts) AS order_month,
        i.ingredient_name,
        SUM(od.quantity) AS total_ingredients_used
    FROM
        raw_pos.order_detail od
        JOIN raw_pos.order_header oh ON od.order_id = oh.order_id
        JOIN harmonized.ingredient_to_menu_lookup iml ON od.menu_item_id =
iml.menu_item_id
        JOIN harmonized.ingredient i ON iml.ingredient_name = i.ingredient_name
        JOIN raw_pos.location l ON l.location_id = oh.location_id
    WHERE l.country = 'United States'
    GROUP BY
        oh.truck_id,
        order_year,
        order_month,
        i.ingredient_name
    ORDER BY
        oh.truck_id,
        total_ingredients_used DESC;
```

ステップ 4 - 最終出力のクエリ

パイプラインの最後のテーブルをクエリしてみましょう。更新が完了するまで数分待つと、前のステップで挿入した注文からの 2 つのバインミーの成分使用量が表示されます。パイプライン全体が自動的に更新されました。

```
-- 最大2分待ってからこのクエリを再実行する必要がある場合があります
SELECT
    truck_id,
    ingredient_name,
    SUM(total_ingredients_used) AS total_ingredients_used
FROM
    harmonized.ingredient_usage_by_truck
WHERE
    order_month = 1
    AND truck_id = 15
GROUP BY truck_id, ingredient_name
ORDER BY total_ingredients_used DESC;
```

パイプラインの可視化 (Visualize the Pipeline)

概要

最後に、パイプラインの有向非巡回グラフ、つまり DAG を視覚化しましょう。DAG は、データがテーブル間をどのように流れるかを示し、パイプラインの健全性とラグを監視するために使用できます。

ステップ 1 - グラフビューへのアクセス

Snowsight で DAG にアクセスするには:

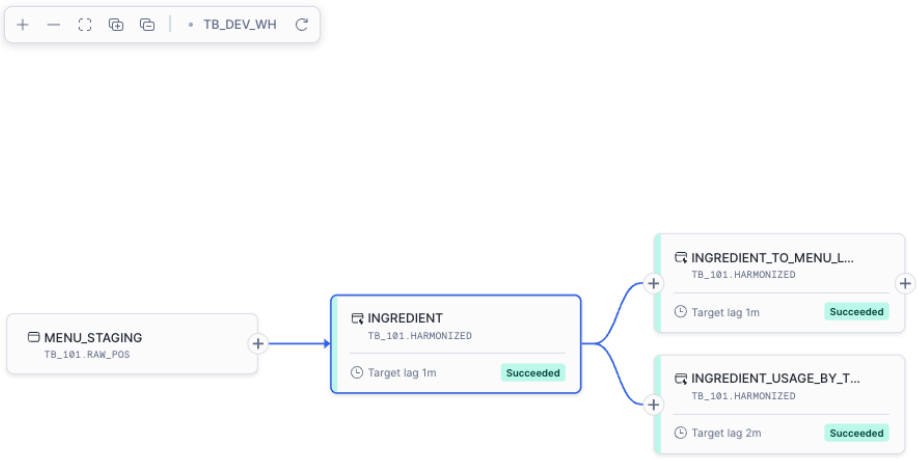
1. **Data** » **Database** に移動します。
2. データベースオブジェクトエクスプローラーで、データベース **TB_101** とスキーマ **HARMONIZED** を展開します。
3. **Dynamic Tables** をクリックします。
4. 作成したダイナミックテーブルのいずれか（例： `INGREDIENT_USAGE_BY_TRUCK`）を選択します。
5. メインウィンドウの **Graph** タブをクリックします。

これで、ベーステーブルがダイナミックテーブルにどのように流れるかを示す、パイプラインの視覚化が表示されます。

TB_101 / HARMONIZED / **INGREDIENT**

Dynamic Table | ACCOUNTADMIN | 1 day ago | 173 | TB_DE_WH | Full

Table Details | Columns | Data Preview | Graph | Refresh History



INGREDIENT

TB_101.HARMONIZED

Details | Definition

Lag Metrics

Last Refresh	Succeeded
Time Wit...	100%
Current Lag	12s
Target Lag	1m
Max Lag	1m 19s

Configuration

State	Active
Refresh Mode	Full
Object Type	Dynamic Table
Rows	173
Warehouse	TB_DE_WH
Owner	ACCOUNTAD...