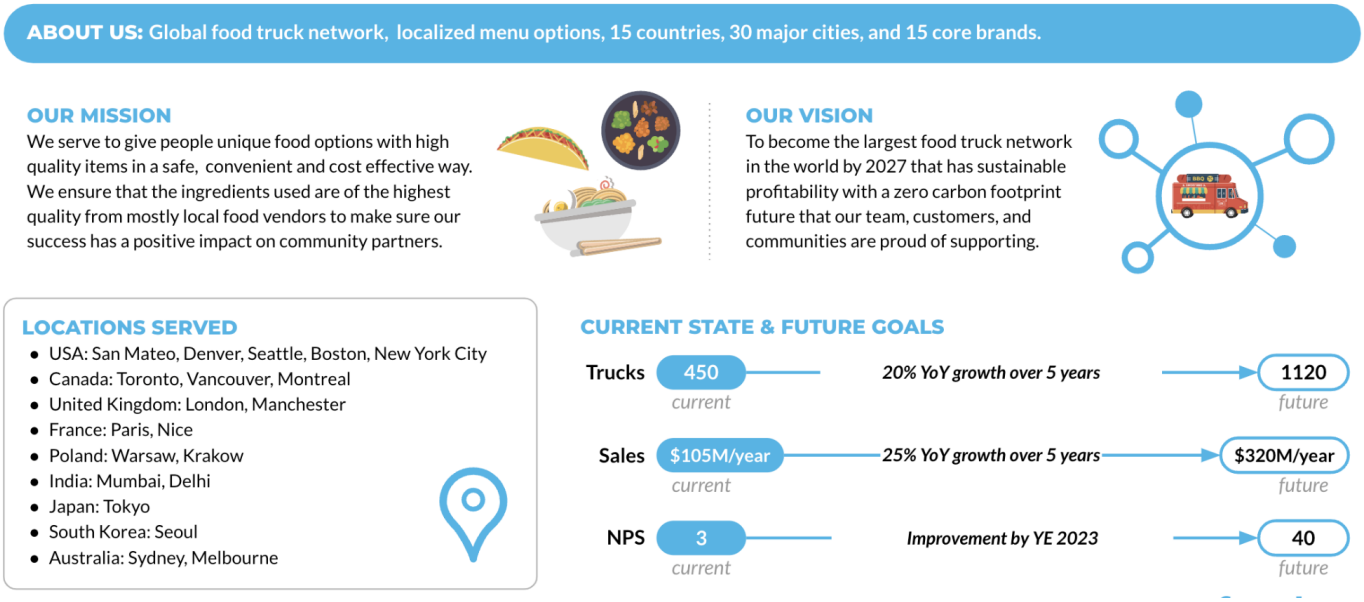
Vignette 1: Snowflakeを始めよう (Getting Started with Snowflake)

概要

Zero to Snowflake クイックスタートへようこそ！このガイドは、Snowflake AI データクラウドの主要分野を巡る統合された旅です。ウェアハウジングとデータ変換の基礎から始め、自動化されたデータパイプラインを構築し、Cortex Playground を使用して LLM を実験し、テキスト要約のためのさまざまなモデルを比較します。また、AISQL 関数を使用してシンプルな SQL コマンドで顧客レビューの感情を即座に分析し、Cortex Search を活用してインテリジェントなテキスト検索を行い、Cortex Analyst を利用して対話型ビジネスインテリジェンスを実現します。最後に、強力なガバナンス制御でデータを保護し、シームレスなデータコラボレーションを通じて分析を強化する方法を学びます。

これらの概念を、架空のフードトラック「Tasty Bytes」のサンプルデータセットを使用して適用し、データ操作を改善および合理化します。いくつかのワークロード固有のシナリオを通じてこのデータセットを調査し、Snowflake がビジネスに提供する利点を実証します。

Tasty Bytesとは?



私たちの使命は、地元のベンダーからの新鮮な食材の使用を重視し、ユニークで高品質な食品オプションを便利かつ費用対効果の高い方法で提供することです。彼らのビジョンは、二酸化炭素排出量ゼロで世界最大のフードトラックネットワークになることです。

前提条件

- サポートされている Snowflake ブラウザ
- Enterprise または Business Critical の Snowflake アカウント
- Snowflake アカウントをお持ちでない場合は、30日間無料トライアルアカウントにサインアップ してください。サインアップの際は、必ず **Enterprise** エディションを選択してください。Snowflake のクラウド/リージョン は自由に選択できます。

- 登録後、アクティベーションリンクと Snowflake アカウントの URL が記載されたメールが届きます。
- **Snowflake Cortex AI 機能について:** このラボでは、Snowflake Cortex AI を利用する機能を紹介する場合があり、一部の Cortex AI モデルはリージョン固有です。このラボに必要な機能またはモデルが Snowflake アカウントのプライマリリージョンで利用できない場合、クロスリージョン推論を有効にする必要があります。**これを有効にするには、`ACCOUNTADMIN` ロールを持つユーザーが ワークスペース SQL ファイルで次の SQL コマンドを実行する必要があります:**

```
ALTER ACCOUNT SET CORTEX_ENABLED_CROSS_REGION = 'AWS_US';
```

- 上記の行を SQL ファイルにコピー＆ペーストし、 `ACCOUNTADMIN` ロールでログインした状態で実行するだけです。

学習内容

- **Vignette 1: Snowflakeを始める:** Snowflake ウェアハウス、キャッシュ、クローン、タイムトラベルの基礎。
- **Vignette 2: シンプルなデータパイプライン:** ダイナミックテーブルを使用して半構造化データを取り込み、変換する方法。
- **Vignette 3: Snowflake Cortex AI:** 実験、スケーラブルな分析、AI 支援開発、対話型ビジネスインテリジェンスのために、Snowflake の包括的な AI 機能を活用する方法。
- **Vignette 4: Horizon によるガバナンス:** ロール、分類、マスキング、行アクセスポリシーを使用してデータを保護する方法。
- **Vignette 5: アプリとコラボレーション:** Snowflake マーケットプレイスを活用して、サードパーティのデータセットで内部データを強化する方法。

構築するもの

- Snowflake プラットフォームの中核に関する包括的な理解。
- 構成済みの仮想ウェアハウス。
- ダイナミックテーブルを使用した自動 ELT パイプライン。
- Snowflake AI を活用した完全なインテリジェンス顧客分析プラットフォーム。
- ロールとポリシーを備えた堅牢なデータガバナンスフレームワーク。
- ファーストパーティデータとサードパーティデータを組み合わせた強化された分析ビュー。

セットアップ

概要

このクイックスタートでは、Snowflake ワークスペース を使用して、このコースに必要なすべての SQL スクリプトを整理、編集、実行します。セットアップと各 Vignette 用に専用の SQL ファイルを作成します。これにより、コードが整理され、管理しやすくなります。

最初の SQL ファイルを作成し、必要なセットアップコードを追加して実行する方法を見ていきましょう。

ステップ 1 - セットアップ SQL ファイルの作成

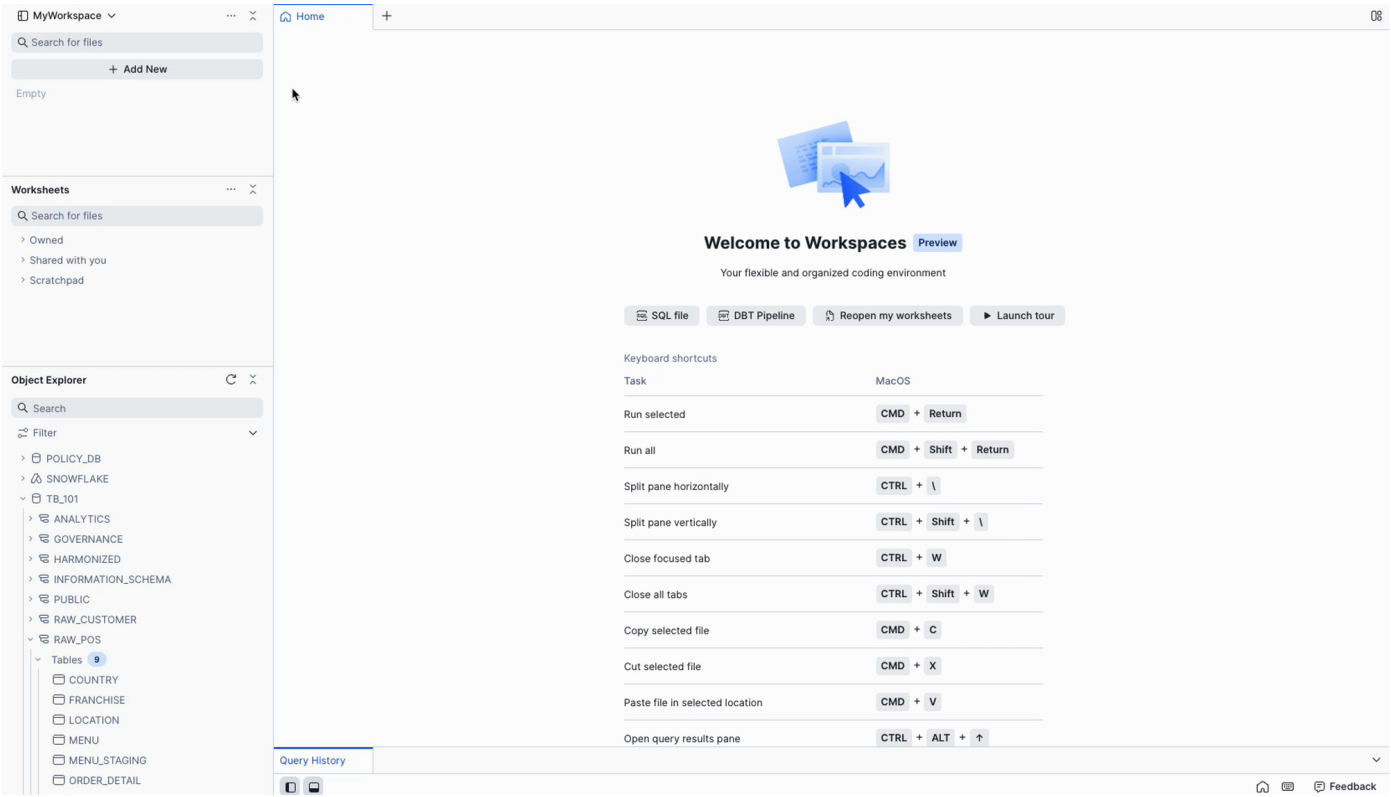
まず、セットアップスクリプトを配置する場所が必要です。

1. **ワークスペース に移動:** Snowflake UI の左側のナビゲーションメニューで、 **Projects** » **Workspaces** をクリックします。これは、すべての SQL ファイルの中心的なハブです。
2. **新しい SQL ファイルの作成:** ワークスペース 領域の左上にある **+ Add New** ボタンを見つけてクリックし、**SQL File** を選択します。これにより、新しい空の SQL ファイルが生成されます。
3. **SQL ファイルの名前変更:** 新しい SQL ファイルには、作成されたタイムスタンプに基づいた名前が付けられます。**Zero To Snowflake - Setup** のようなわかりやすい名前を付けてください。

ステップ 2 - セットアップスクリプトの追加と実行

SQL ファイルができたので、セットアップ SQL を追加して実行します。

1. **SQL コードのコピー:** セットアップファイル のリンクをクリックし、クリップボードにコピーします。
2. **SQL ファイルへの貼り付け:** Snowflake の Zero To Snowflake Setup SQL ファイルに戻り、スクリプト全体をエディタに貼り付けます。
3. **スクリプトの実行:** SQL ファイル内のすべてのコマンドを順番に実行するには、エディタの左上にある **"Run All"** ボタンをクリックします。これにより、今後の Vignette で必要となるロール、スキーマ、ウェアハウスの作成など、必要なすべてのセットアップアクションが実行されます。



今後の手順

新しい SQL ファイルを作成するために完了したばかりのプロセスは、このコースの以降のすべての Vignette で使用するのとまったく同じワークフローです。

新しい Vignette ごとに、以下を行います。

1. **新しい** SQL ファイルを作成する。
2. わかりやすい名前を付ける（例：Vignette 1 - Getting Started with Snowflake）。
3. その特定の Vignette 用の SQL スクリプトをコピーして貼り付ける。
4. 各 SQL ファイルには、フォローするための必要なすべての手順とコマンドが含まれています。

Snowflakeを始めよう (Get Started with Snowflake)

概要

この Vignette では、仮想ウェアハウスの探索、クエリ結果キャッシュの使用、基本的なデータ変換の実行、タイムトラベルによるデータ復旧の活用、リソースモニターと予算によるアカウントの監視を通じて、Snowflake のコアコンセプトについて学びます。

学習内容

- 仮想ウェアハウスを作成、構成、スケーリングする方法。
- クエリ結果キャッシュを活用する方法。
- 開発にゼロコピークローンを使用する方法。
- データを変換およびクリーンアップする方法。
- UNDROP を使用してドロップされたテーブルを即座に復元する方法。
- リソースモニターを作成して適用する方法。
- コストを監視するための予算を作成する方法。
- ユニバーサル検索を使用してオブジェクトや情報を検索する方法。

構築するもの

- Snowflake 仮想ウェアハウス
- ゼロコピークローンを使用したテーブルの開発用コピー
- リソースモニター
- 予算

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 10 - シンプルなデータパイプラインに進むことができます。

仮想ウェアハウスと設定 (Virtual Warehouses and Settings)

概要

仮想ウェアハウスは、Snowflake データ分析を実行するための動的でスケーラブルかつ費用対効果の高いコンピューティングパワーです。その目的は、基礎となる技術的な詳細を心配することなく、すべてのデータ処理ニーズを処理することです。

ステップ 1 - コンテキストの設定

まず、セッションコンテキストを設定しましょう。クエリを実行するには、SQL ファイルの上部にある 3 つのクエリを強調表示し、"▶ Run" ボタンをクリックします。

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

USE DATABASE tb_101;
USE ROLE accountadmin;
```

ステップ 2 - ウェアハウスの作成

最初のウェアハウスを作成しましょう！このコマンドは、最初は一時停止状態の新しい X-Small ウェアハウスを作成します。

```
CREATE OR REPLACE WAREHOUSE my_wh
    COMMENT = 'My TastyBytes warehouse'
    WAREHOUSE_TYPE = 'standard'
    WAREHOUSE_SIZE = 'xsmall'
    MIN_CLUSTER_COUNT = 1
    MAX_CLUSTER_COUNT = 2
    SCALING_POLICY = 'standard'
    AUTO_SUSPEND = 60
    INITIALLY_SUSPENDED = true
    AUTO_RESUME = false;
```

> **仮想ウェアハウス**: 仮想ウェアハウス（単に「ウェアハウス」と呼ばれることが多い）は、Snowflake 内のコンピューティングリソースのクラスターです。ウェアハウスは、クエリ、DML 操作、およびデータのロードに必要です。詳細については、ウェアハウスの概要を参照してください。

ステップ 3 - ウェアハウスの使用と再開

ウェアハウスができたので、それをセッションのアクティブなウェアハウスとして設定する必要があります。次のステートメントを実行します。

```
USE WAREHOUSE my_wh;
```

次のクエリを実行しようとすると失敗します。これは、ウェアハウスが一時停止しており、 `AUTO_RESUME` が有効になっていないためです。

```
SELECT * FROM raw_pos.truck_details;
```

再開して、将来的に自動再開するように設定しましょう。

```
ALTER WAREHOUSE my_wh RESUME;
ALTER WAREHOUSE my_wh SET AUTO_RESUME = TRUE;
```

これで、もう一度クエリを試してください。正常に実行されるはずです。

```
SELECT * FROM raw_pos.truck_details;
```

ステップ 4 - ウェアハウスのスケーリング

Snowflake のウェアハウスは、弾力性を持つように設計されています。ウェアハウスをオンザフライでスケールアップして、より集中的なワークロードを処理できます。ウェアハウスを X-Large にスケーリングしましょう。

```
ALTER WAREHOUSE my_wh SET warehouse_size = 'XLarge';
```

より大きなウェアハウスを使用して、トラックブランドごとの総売上高を計算するクエリを実行しましょう。

```
SELECT
    o.truck_brand_name,
    COUNT(DISTINCT o.order_id) AS order_count,
    SUM(o.price) AS total_sales
FROM analytics.orders_v o
GROUP BY o.truck_brand_name
ORDER BY total_sales DESC;
```

クエリ結果キャッシュ (Query Result Cache)

概要

これは、Snowflake のもう 1 つの強力な機能であるクエリ結果キャッシュを実演するのに最適な場所です。「トラックごとの売上」クエリを最初に実行したとき、数秒かかった可能性があります。まったく同じクエリをもう一度実行すると、結果はほぼ瞬時に表示されます。これは、クエリ結果が Snowflake のクエリ結果キャッシュにキャッシュされたためです。

ステップ 1 - クエリの再実行

前のステップと同じ「トラックごとの売上」クエリを実行します。クエリ詳細ペインで実行時間に注目してください。はるかに高速になるはずです。

```
SELECT
    o.truck_brand_name,
    COUNT(DISTINCT o.order_id) AS order_count,
    SUM(o.price) AS total_sales
FROM analytics.orders_v o
GROUP BY o.truck_brand_name
ORDER BY total_sales DESC;
```

> **クエリ結果キャッシュ**: 結果は、任意のクエリに対して 24 時間保持されます。結果キャッシュへのヒットにはコンピューティングリソースがほとんど不要なため、頻繁に実行されるレポートやダッシュボードに最適です。キャッシュはクラウドサービスレイヤーに存在するため、アカウント内のすべてのユーザーとウェアハウスからグローバルにアクセスできます。詳細については、 永続化されたクエリ結果の使用に関するドキュメントを参照してください。

ステップ 2 - スケールダウン

これからはより小さなデータセットを扱うことになるため、クレジットを節約するためにウェアハウスを X-Small にスケールダウンできます。

```
ALTER WAREHOUSE my_wh SET warehouse_size = 'XSmall';
```

基本的な変換テクニック (Basic Transformation Techniques)

概要

このセクションでは、データをクリーンアップするための基本的な変換テクニックと、ゼロコピークローンを使用して開発環境を作成する方法を見ていきます。目標はフードトラックの製造元を分析することですが、このデータは現在 `VARIANT` 列内にネストされています。

ステップ 1 - ゼロコピークローンによる開発テーブルの作成

まず、`truck_build` 列を見てみましょう。

```
SELECT truck_build FROM raw_pos.truck_details;
```

このテーブルには各トラックのメーカー、モデル、年式に関するデータが含まれていますが、VARIANT と呼ばれる特別なデータ型にネスト（埋め込み）されています。この列に対して操作を実行してこれらの値を抽出できますが、まずテーブルの開発用コピーを作成します。

`truck_details` テーブルの開発用コピーを作成しましょう。Snowflake のゼロコピークローンを使用すると、追加のストレージを使用せずに、テーブルの同一の完全に独立したコピーを即座に作成できます。

```
CREATE OR REPLACE TABLE raw_pos.truck_dev CLONE raw_pos.truck_details;
```

> ゼロコピークローン: クローニングは、ストレージを複製せずにデータベースオブジェクトのコピーを作成します。元のオブジェクトまたはクローンに加えられた変更は新しいマイクロパーティションとして保存され、もう一方のオブジェクトは変更されません。

ステップ 2 - 新しい列の追加とデータの変換

安全な開発テーブルができたので、`year`、`make`、`model` の列を追加しましょう。その後、`truck_build` `VARIANT` 列からデータを抽出し、新しい列に入力します。

```
-- 新しい列を追加する
ALTER TABLE raw_pos.truck_dev ADD COLUMN IF NOT EXISTS year NUMBER;
ALTER TABLE raw_pos.truck_dev ADD COLUMN IF NOT EXISTS make VARCHAR(255);
ALTER TABLE raw_pos.truck_dev ADD COLUMN IF NOT EXISTS model VARCHAR(255);

-- データを抽出して更新する
UPDATE raw_pos.truck_dev
SET
    year = truck_build:year::NUMBER,
    make = truck_build:make::VARCHAR,
    model = truck_build:model::VARCHAR;
```

ステップ 3 - データのクレンジング

クエリを実行して、トラックメーカーの分布を確認しましょう。

```
SELECT
    make,
    COUNT(*) AS count
FROM raw_pos.truck_dev
GROUP BY make
ORDER BY make ASC;
```

最後のクエリの結果で何か奇妙なことに気づきましたか？データ品質の問題が見られます。「Ford」と「Ford_」が別々の製造元として扱われています。単純な `UPDATE` ステートメントでこれを簡単に修正しましょう。

```
UPDATE raw_pos.truck_dev
    SET make = 'Ford'
    WHERE make = 'Ford_';
```

ここでは、`Ford_` であるすべての行の make 値を `Ford` に設定するように指示しています。これにより、どの Ford メーカーにもアンダースコアがなくなり、統一されたメーカー数が得られます。

ステップ 4 - SWAP による本番環境へのプロモーション

開発テーブルは現在クリーンアップされ、正しくフォーマットされています。 `SWAP WITH` コマンドを使用して、これを新しい本番テーブルとして即座にプロモートできます。これにより、2 つのテーブルがアトミックにスワップされます。

```
ALTER TABLE raw_pos.truck_details SWAP WITH raw_pos.truck_dev;
```

ステップ 5 - クリーンアップ

スワップが完了したので、新しい本番テーブルから不要な `truck_build` 列を削除できます。また、現在は `truck_dev` という名前になっている古い本番テーブルも削除する必要があります。しかし、次のレッスンのために、メインテーブルを「誤って」削除してみましょう。

```
ALTER TABLE raw_pos.truck_details DROP COLUMN truck_build;

-- 本番テーブルを誤って削除してしまう！
DROP TABLE raw_pos.truck_details;
```

ステップ 6 - UNDROP によるデータ復旧

大変！本番の `truck_details` テーブルを誤って削除してしまいました。幸いなことに、Snowflake のタイムトラベル機能を使用すると、即座に復旧できます。`UNDROP` コマンドは、削除されたオブジェクトを復元します。

ステップ 7 - ドロップの確認

テーブルに対して `DESCRIBE` コマンドを実行すると、存在しないというエラーが表示されます。

```
DESCRIBE TABLE raw_pos.truck_details;
```

ステップ 8 - UNDROP によるテーブルの復元

`truck_details` テーブルを、削除される前とまったく同じ状態に復元しましょう。

```
UNDROP TABLE raw_pos.truck_details;
```

> **タイムトラベル & UNDROP**: Snowflake タイムトラベルを使用すると、定義された期間内の任意の時点の履歴データにアクセスできます。これにより、変更または削除されたデータを復元できます。`UNDROP` はタイムトラベルの機能であり、誤った削除からの回復を簡単にします。

ステップ 9 - 復元の確認とクリーンアップ

テーブルから選択して、テーブルが正常に復元されたことを確認します。その後、実際の開発テーブル `truck_dev` を安全に削除できます。

```
-- テーブルが復元されたことを確認する
SELECT * from raw_pos.truck_details;

-- これで、本物の truck_dev テーブルを削除します
DROP TABLE raw_pos.truck_dev;
```

リソースモニター (Resource Monitors)

概要

コンピューティング使用量の監視は重要です。Snowflake は、ウェアハウスのクレジット使用量を追跡するためのリソースモニターを提供します。クレジット割り当てを定義し、しきい値に達したときにアクション（通知や一時停止など）をトリガーできます。

ステップ 1 - リソースモニターの作成

`my_wh` のリソースモニターを作成しましょう。このモニターの月間割り当ては 100 クレジットで、75% で通知を送信し、90% と 100% でウェアハウスを一時停止します。まず、ロールが `accountadmin` であることを確認してください。

```
USE ROLE accountadmin;

CREATE OR REPLACE RESOURCE MONITOR my_resource_monitor
    WITH CREDIT_QUOTA = 100
    FREQUENCY = MONTHLY
    START_TIMESTAMP = IMMEDIATELY
    TRIGGERS ON 75 PERCENT DO NOTIFY
             ON 90 PERCENT DO SUSPEND
             ON 100 PERCENT DO SUSPEND_IMMEDIATE;
```

ステップ 2 - リソースモニターの適用

モニターを作成したら、それを `my_wh` に適用します。

```
ALTER WAREHOUSE my_wh
    SET RESOURCE_MONITOR = my_resource_monitor;
```

> 各構成が何を処理するかについての詳細は、リソースモニターの操作 のドキュメントを参照してください。

予算の作成 (Create a Budget)

概要

リソースモニターはウェアハウスの使用量を追跡しますが、予算 (Budgets) はすべての Snowflake コストを管理するためのより柔軟なアプローチを提供します。予算は、任意の Snowflake オブジェクトへの支出を追跡し、金額のしきい値に達したときにユーザーに通知できます。

ステップ 1 - SQL による予算の作成

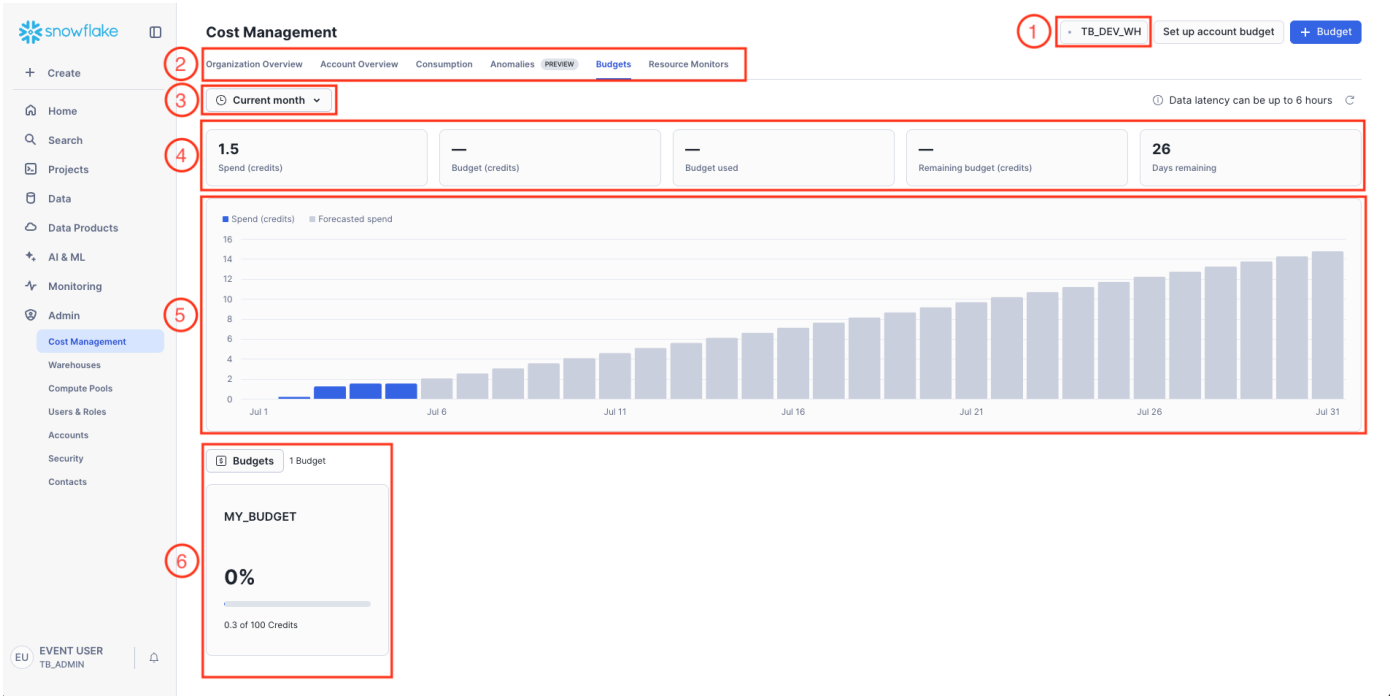
まず、SQL で予算オブジェクトを作成しましょう。

```
CREATE OR REPLACE SNOWFLAKE.CORE.BUDGET my_budget()
    COMMENT = 'My Tasty Bytes Budget';
```

ステップ 2 - Snowsight の予算ページ

Snowsight の予算ページを見てみましょう。

Admin » **Cost Management** » **Budgets** に移動します。



キー:

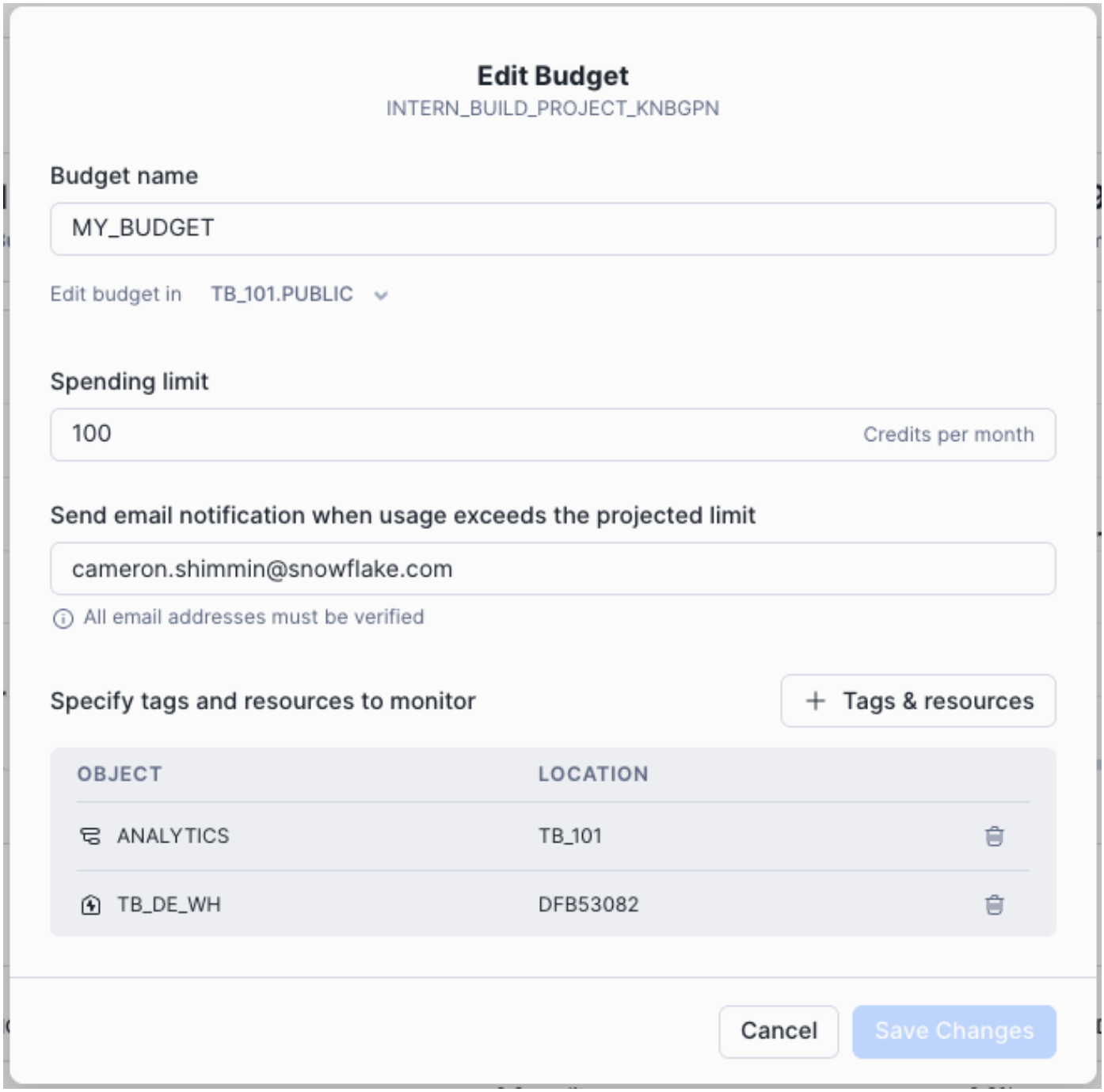
1. ウェアハウスコンテキスト
2. コスト管理ナビゲーション
3. 期間フィルター
4. 主要メトリクスの概要
5. 支出と予測の傾向チャート
6. 予算の詳細

ステップ 3 - Snowsight での予算の構成

予算の構成は、Snowsight UI を通じて行われます。

1. アカウントロールが `ACCOUNTADMIN` に設定されていることを確認します。左下隅でこれを変更できます。
2. 作成した **MY_BUDGET** 予算をクリックします。

3. **Budget Details** をクリックして予算詳細パネルを開き、右側の予算詳細パネルで **Edit** をクリックします。
4. **Spending Limit** を `100` に設定します。
5. 確認済みの通知用メールアドレスを入力します。
6. **+ Tags & Resources** をクリックし、監視対象として **TB_101.ANALYTICS** スキーマと **TB_DE_WH** ウェアハウスを追加します。
7. **Save Changes** をクリックします。



> 予算の詳細なガイドについては、Snowflake 予算のドキュメントを参照してください。

ユニバーサル検索 (Universal Search)

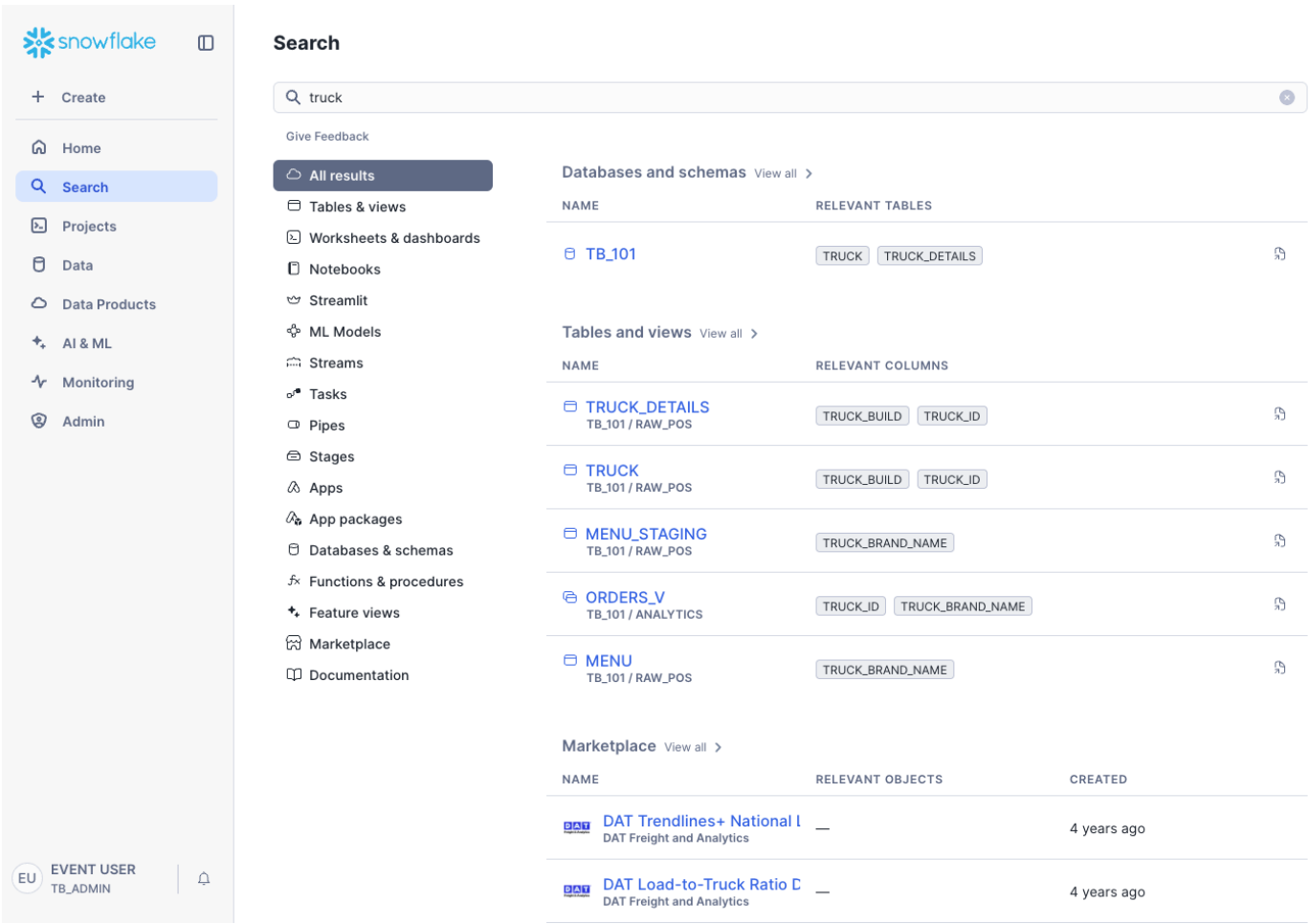
概要

ユニバーサル検索を使用すると、アカウント内の任意のオブジェクトを簡単に見つけることができるほか、マーケットプレイスのデータ製品、関連する Snowflake ドキュメント、コミュニティナレッジベースの記事を探索できます。

ステップ 1 - オブジェクトの検索

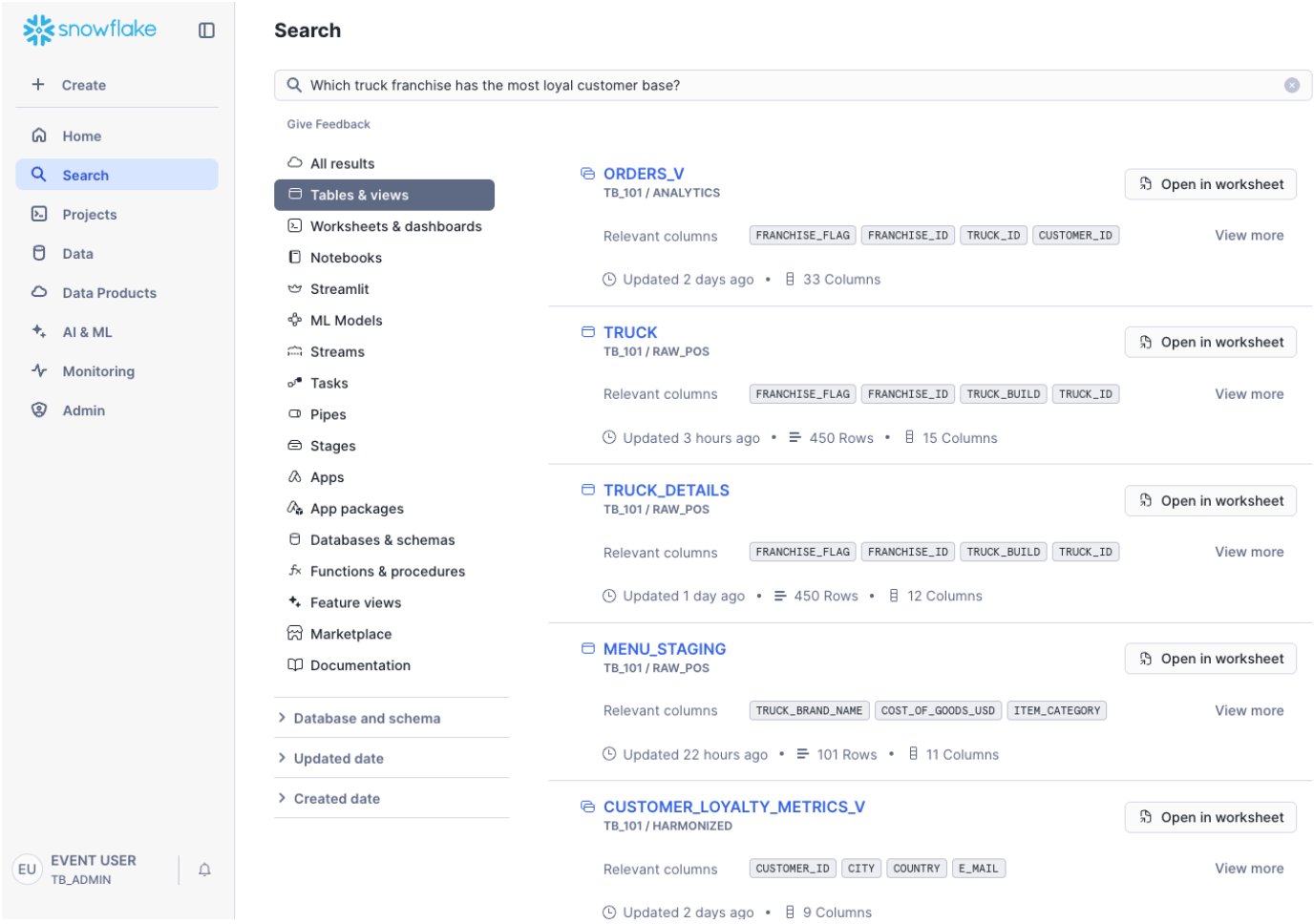
今すぐ試してみましょう。

1. 左側のナビゲーションメニューで **Search** をクリックします。
2. 検索バーに `truck` と入力します。
3. 結果を確認します。テーブルやビューなどのアカウント上のオブジェクトのカテゴリや、関連ドキュメントが表示されます。



ステップ 2 - 自然言語検索の使用

自然言語を使用することもできます。たとえば、次のように検索します: `Which truck franchise has the most loyal customer base?` (最も忠実な顧客基盤を持つトラックフランチャイズはどれですか？)
ユニバーサル検索は関連するテーブルとビューを返し、質問への回答に役立つ可能性のある列を強調表示して、分析の優れた出発点を提供します。


snowflake
Create
Home
Search
Projects
Data
Data Products
AI & ML
Monitoring
Admin
EU
EVENT USER
TB_ADMIN
Search
Which truck franchise has the most loyal customer base?
Give Feedback
All results
Tables & views
Worksheets & dashboards
Notebooks
Streamlit
ML Models
Streams
Tasks
Pipes
Stages
Apps
App packages
Databases & schemas
Functions & procedures
Feature views
Marketplace
Documentation
Database and schema
Updated date
Created date
ORDERS_V
TB_101 / ANALYTICS
Open in worksheet
Relevant columns
FRANCHISE_FLAG
FRANCHISE_ID
TRUCK_ID
CUSTOMER_ID
View more
Updated 2 days ago • 33 Columns
TRUCK
TB_101 / RAW_POS
Open in worksheet
Relevant columns
FRANCHISE_FLAG
FRANCHISE_ID
TRUCK_BUILD
TRUCK_ID
View more
Updated 3 hours ago • 450 Rows • 15 Columns
TRUCK_DETAILS
TB_101 / RAW_POS
Open in worksheet
Relevant columns
FRANCHISE_FLAG
FRANCHISE_ID
TRUCK_BUILD
TRUCK_ID
View more
Updated 1 day ago • 450 Rows • 12 Columns
MENU_STAGING
TB_101 / RAW_POS
Open in worksheet
Relevant columns
TRUCK_BRAND_NAME
COST_OF_GOODS_USD
ITEM_CATEGORY
View more
Updated 22 hours ago • 101 Rows • 11 Columns
CUSTOMER_LOYALTY_METRICS_V
TB_101 / HARMONIZED
Open in worksheet
Relevant columns
CUSTOMER_ID
CITY
COUNTRY
E_MAIL
View more
Updated 2 days ago • 9 Columns