

Vignette 5: アプリとコラボレーション (Apps & Collaboration)

APPS & COLLABORATION

概要

この Vignette では、Snowflake マーケットプレイスを通じて Snowflake がシームレスなデータコラボレーションをどのように促進するかを探ります。ライブでクエリ可能なサードパーティのデータセットを取得し、それらを独自の内部データと即座に結合して、従来の ETL パイプラインを必要とせずに新しいインサイトを引き出すことがいかに簡単かを見ていきます。

学習内容

- Snowflake マーケットプレイスからデータを見つけて取得する方法。
- ライブ共有データを即座にクエリする方法。
- マーケットプレイスデータを自分のアカウントデータと結合して、強化されたビューを作成する方法。
- サードパーティの Point-of-Interest (POI) データを活用して詳細な分析を行う方法。
- 共通テーブル式 (CTE) を使用して複雑なクエリを構造化する方法。

構築するもの

- 内部売上データと外部の気象データおよび POI データを組み合わせた強化された分析ビュー。

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。

Snowflake マーケットプレイスからデータを取得する (Acquire Data from Snowflake Marketplace)

概要

アナリストの 1 人が、天候がフードトラックの売上にどのように影響するかを確認したいと考えています。これを行うために、Snowflake マーケットプレイスを使用して Weather Source からライブ気象データを取得します。これは、独自の売上データと直接結合できます。マーケットプレイスを使用すると、データの複製や ETL を行うことなく、サードパーティプロバイダーからのライブでクエリ可能なデータにアクセスできます。

Snowflake マーケットプレイスの紹介: マーケットプレイスは、さまざまなサードパーティのデータ、アプリケーション、AI 製品を見つけてアクセスするための一元化されたハブを提供します。

ステップ 1 - 初期コンテキストの設定

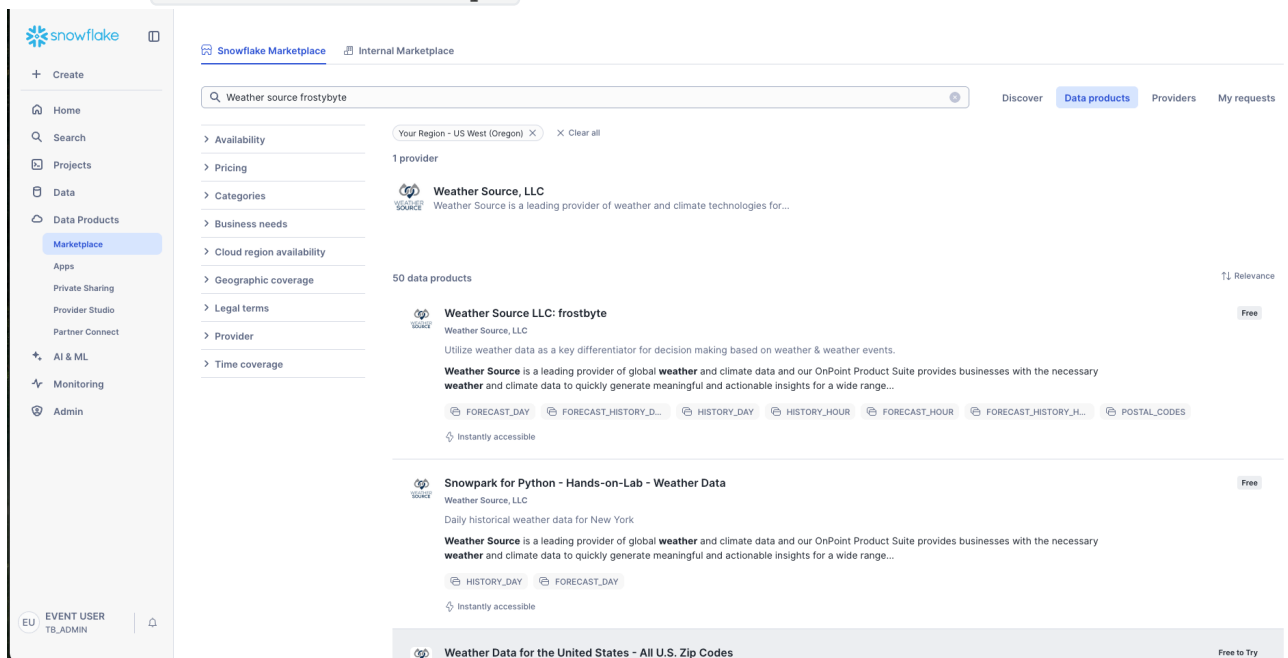
まず、マーケットプレイスからデータを取得するために必要な `accountadmin` ロールを使用するようにコンテキストを設定しましょう。

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

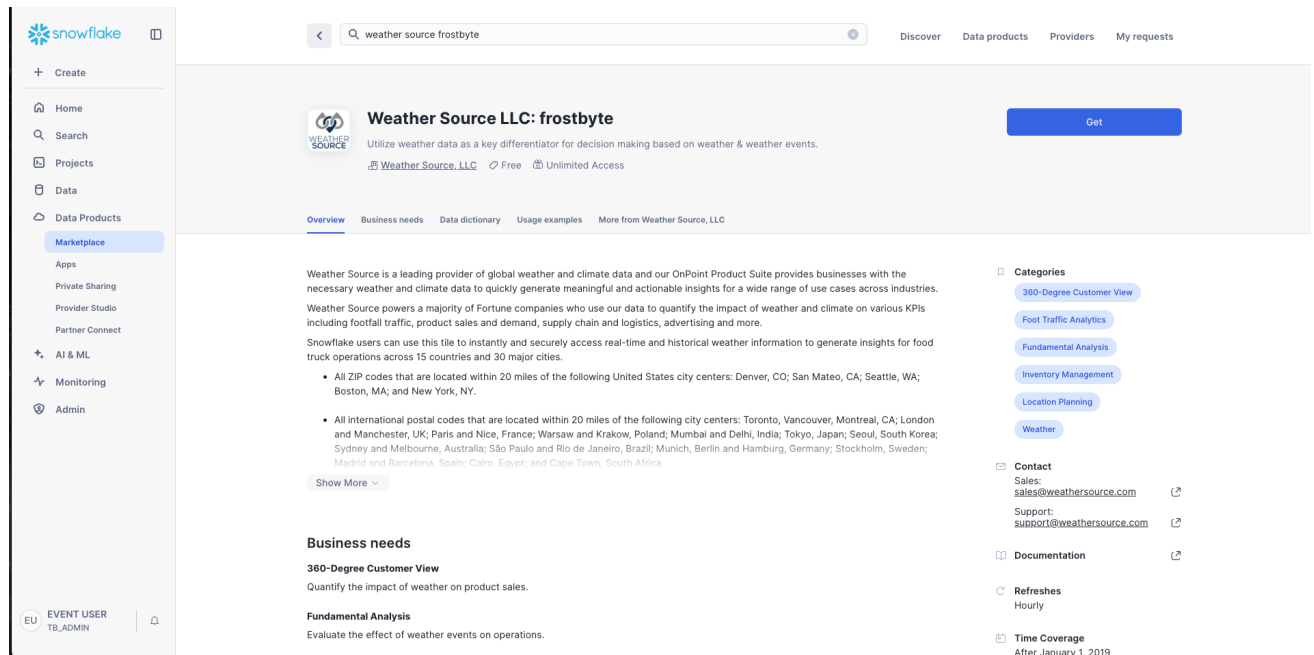
ステップ 2 - Weather Source データの取得

Snowsight UI で次の手順に従って、Weather Source データを取得します。

1. `ACCOUNTADMIN` ロールを使用していることを確認します。
2. 左側のナビゲーションメニューから **Data Products** » **Marketplace** に移動します。
3. 検索バーに `Weather Source frostbyte` と入力します。



4. **Weather Source LLC: frostbyte** リスティングをクリックします。



5. **Get** ボタンをクリックします。
6. オプションをクリックして展開し、**Database name** を `ZTS_WEATHERSOURCE` に変更します。
7. **PUBLIC** ロールにアクセス権を付与します。
8. **Get** をクリックします。

このプロセスにより、Weather Source データが新しいデータベースとしてアカウントですぐに利用可能になり、クエリを実行できるようになります。

アカウントデータと共有データの統合 (Integrate Account Data with Shared Data)

概要

Weather Source データがアカウントにあるので、アナリストはすぐに既存の Tasty Bytes データとの結合を開始できます。ETL ジョブの実行を待つ必要はありません。

ステップ 1 - 共有データの探索

`tb_analyst` ロールに切り替えて、新しい気象データの探索を始めましょう。共有で利用可能なすべての米国の都市のリストと、いくつかの平均的な気象メトリクスを取得することから始めます。

```
USE ROLE tb_analyst;

SELECT
    DISTINCT city_name,
    AVG(max_wind_speed_100m_mph) AS avg_wind_speed_mph,
    AVG(avg_temperature_air_2m_f) AS avg_temp_f,
    AVG(tot_precipitation_in) AS avg_precipitation_in,
    MAX(tot_snowfall_in) AS max_snowfall_in
FROM zts_weathersource.onpoint_id.history_day
WHERE country = 'US'
GROUP BY city_name;
```

ステップ 2 - 強化されたビューの作成

次に、生の `country` データと Weather Source 共有からの過去の毎日の気象データを結合するビューを作成しましょう。これにより、Tasty Bytes が運営されている都市の気象メトリクスの統合ビューが得られます。

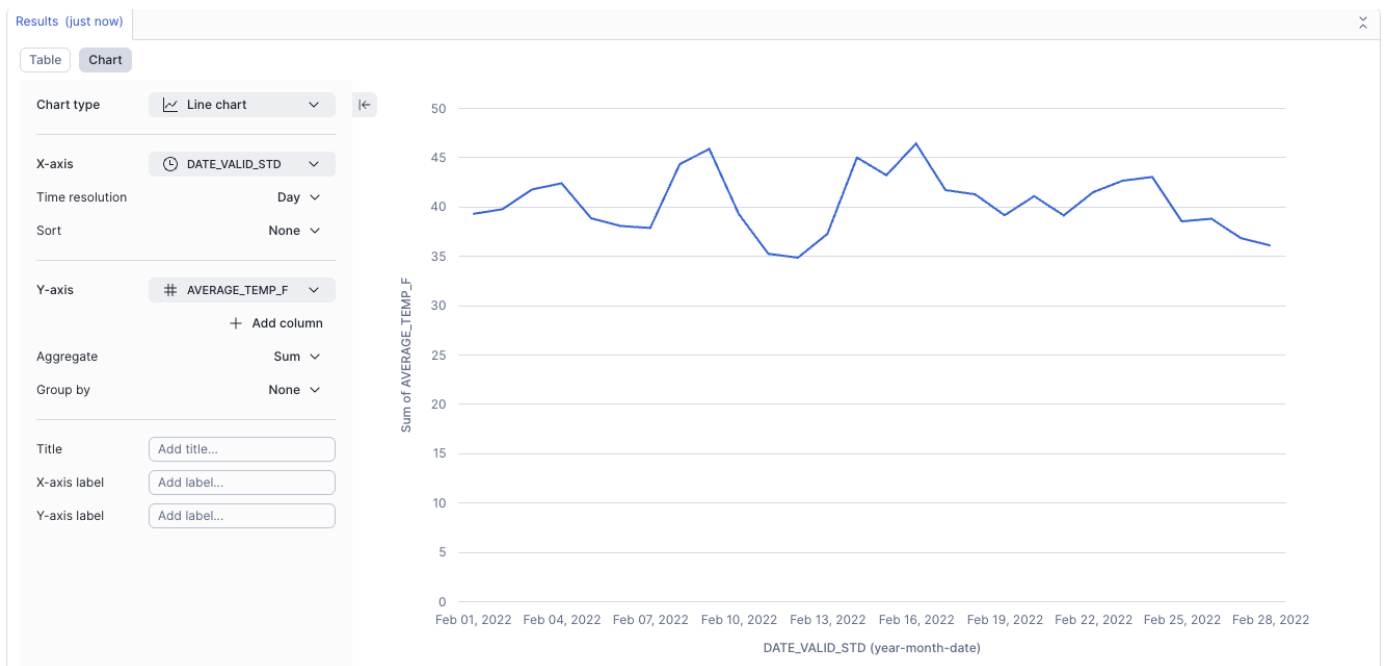
```
CREATE OR REPLACE VIEW harmonized.daily_weather_v
COMMENT = 'Weather Source Daily History filtered to Tasty Bytes supported Cities'
AS
SELECT
    hd.*,
    TO_VARCHAR(hd.date_valid_std, 'YYYY-MM') AS yyyy_mm,
    pc.city_name AS city,
    c.country AS country_desc
FROM zts_weathersource.onpoint_id.history_day hd
JOIN zts_weathersource.onpoint_id.postal_codes pc
    ON pc.postal_code = hd.postal_code
    AND pc.country = hd.country
JOIN raw_pos.country c
    ON c.iso_country = hd.country
    AND c.city = hd.city_name;
```

ステップ 3 - 強化されたデータの分析と可視化

新しいビューを使用して、アナリストは 2022 年 2 月のドイツ・ハンブルクの平均気温をクエリできます。以下のクエリを実行し、Snowsight で折れ線グラフとして直接視覚化します。

```
SELECT
    dw.country_desc,
    dw.city_name,
    dw.date_valid_std,
    AVG(dw.avg_temperature_air_2m_f) AS average_temp_f
FROM harmonized.daily_weather_v dw
WHERE dw.country_desc = 'Germany'
    AND dw.city_name = 'Hamburg'
    AND YEAR(date_valid_std) = 2022
    AND MONTH(date_valid_std) = 2
GROUP BY dw.country_desc, dw.city_name, dw.date_valid_std
ORDER BY dw.date_valid_std DESC;
```

1. 上記のクエリを実行します。
2. **Results** ペインで、**Chart** をクリックします。
3. **Chart Type** を **Line** に設定します。
4. **X-Axis** を **DATE_VALID_STD** に設定します。
5. **Y-Axis** を **AVERAGE_TEMP_F** に設定します。



ステップ 4 - 売上と天気のビューの作成

さらに一歩進んで、`orders_v` ビューと新しい `daily_weather_v` を組み合わせて、売上が気象条件とどのように相関するかを見てみましょう。

```
CREATE OR REPLACE VIEW analytics.daily_sales_by_weather_v
COMMENT = 'Daily Weather Metrics and Orders Data'
AS
WITH daily_orders_aggregated AS (
    SELECT DATE(o.order_ts) AS order_date, o.primary_city, o.country,
           o.menu_item_name, SUM(o.price) AS total_sales
    FROM harmonized.orders_v o
```

```

GROUP BY ALL
)
SELECT
    dw.date_valid_std AS date, dw.city_name, dw.country_desc,
    ZEROIFNULL(doa.total_sales) AS daily_sales, doa.menu_item_name,
    ROUND(dw.avg_temperature_air_2m_f, 2) AS avg_temp_fahrenheit,
    ROUND(dw.tot_precipitation_in, 2) AS avg_precipitation_inches,
    ROUND(dw.tot_snowdepth_in, 2) AS avg_snowdepth_inches,
    dw.max_wind_speed_100m_mph AS max_wind_speed_mph
FROM harmonized.daily_weather_v dw
LEFT JOIN daily_orders_aggregated doa
    ON dw.date_valid_std = doa.order_date
    AND dw.city_name = doa.primary_city
    AND dw.country_desc = doa.country
ORDER BY date ASC;

```

ステップ 5 - ビジネス上の質問への回答

アナリストは、「シアトル市場での売上高に、多量の降水量がどのような影響を与えていますか？」などの複雑なビジネス上の質問に答えることができるようになりました。

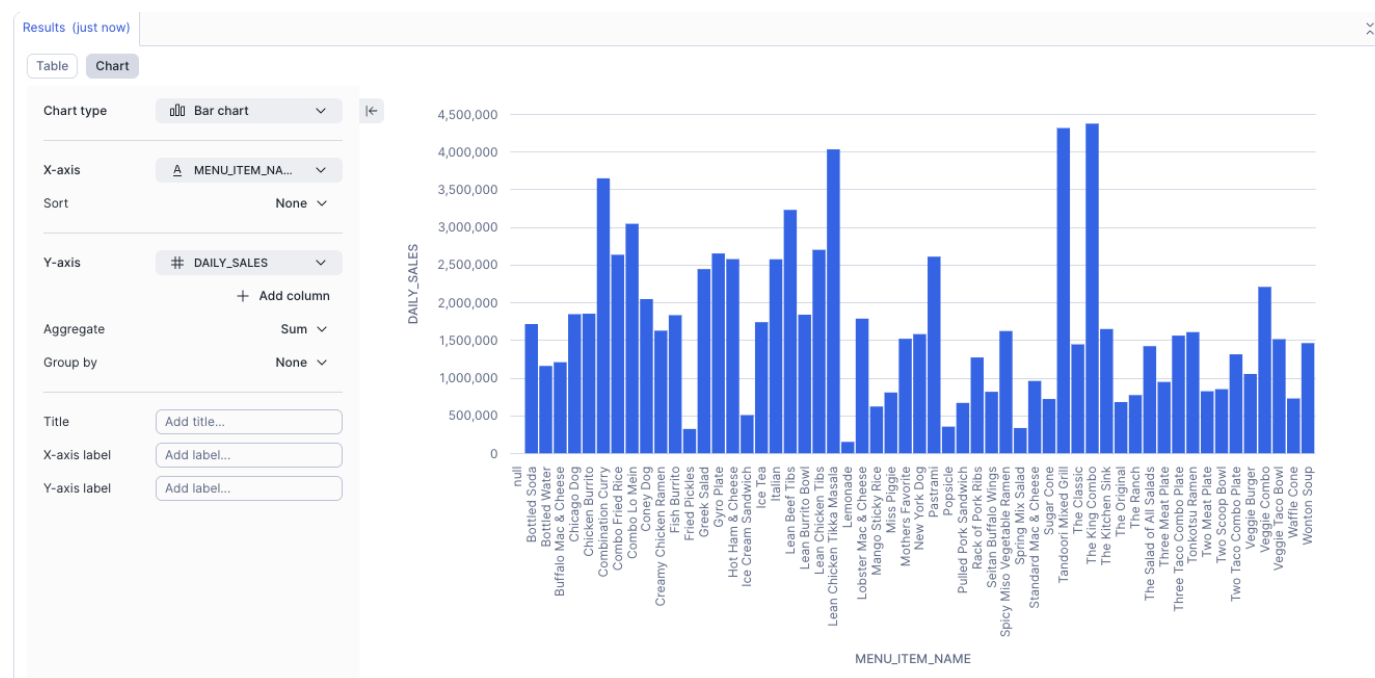
```

SELECT * EXCLUDE (city_name, country_desc, avg_snowdepth_inches, max_wind_speed_mph)
FROM analytics.daily_sales_by_weather_v
WHERE
    country_desc = 'United States'
    AND city_name = 'Seattle'
    AND avg_precipitation_inches >= 1.0
ORDER BY date ASC;

```

結果を Snowsight で棒グラフとして再度視覚化してみましょう。

1. 上記のクエリを実行します。
2. **Results** ペインで、**Chart** をクリックします。
3. **Chart Type** を **Bar** に設定します。
4. **X-Axis** を **MENU_ITEM_NAME** に設定します。
5. **Y-Axis** を **DAILY_SALES** に設定します。



Point-of-Interest データの探索 (Explore Point-of-Interest Data)

概要

アナリストは、フードトラックの特定の場所についてさらに詳しく知りたいと考えています。Snowflake マーケットプレイスの別のプロバイダーである Safegraph から Point-of-Interest (POI) データを取得して、分析をさらに強化できます。

ステップ 1 - Safegraph POI データの取得

以前と同じ手順に従って、マーケットプレイスから Safegraph データを取得します。

1. **ACCOUNTADMIN** ロールを使用していることを確認します。
2. **Data Products » Marketplace** に移動します。
3. 検索バーに **safegraph frostbyte** と入力します。
4. **Safegraph: frostbyte** リスティングを選択し、**Get** をクリックします。
5. オプションをクリックして展開し、**Database name** を **ZTS_SAFEGRAPH** に設定します。
6. **PUBLIC** ロールにアクセス権を付与します。
7. **Get** をクリックします。

ステップ 2 - POI ビューの作成

内部の **location** データと Safegraph POI データを結合するビューを作成しましょう。

```
CREATE OR REPLACE VIEW harmonized.tastybytes_poi_v
AS
SELECT
    l.location_id, sg.postal_code, sg.country, sg.city, sg.iso_country_code,
    sg.location_name, sg.top_category, sg.category_tags,
    sg.includes_parking_lot, sg.open_hours
FROM raw_pos.location l
JOIN zts_safegraph.public.frostbyte_tb_safegraph_s sg
    ON l.location_id = sg.location_id
    AND l.iso_country_code = sg.iso_country_code;
```

ステップ 3 - POI データと気象データの結合

これで、内部データ、気象データ、POI データの 3 つのデータセットすべてを組み合わせることができます。2022 年の米国で最も風が強かったトラックの場所トップ 3 を見つけましょう。

```
SELECT TOP 3
    p.location_id, p.city, p.postal_code,
    AVG(hd.max_wind_speed_100m_mph) AS average_wind_speed
FROM harmonized.tastybytes_poi_v AS p
JOIN zts_weathersource.onpoint_id.history_day AS hd
    ON p.postal_code = hd.postal_code
WHERE
    p.country = 'United States'
    AND YEAR(hd.date_valid_std) = 2022
GROUP BY p.location_id, p.city, p.postal_code
ORDER BY average_wind_speed DESC;
```

ステップ 4 - ブランドの天候に対する回復力の分析

最後に、ブランドの回復力を判断するために、より複雑な分析を行いましょう。共通テーブル式 (CTE) を使用して、まず最も風が強い場所を見つけ、次にそれらの場所での各トラックブランドの「穏やかな」日と「風の強い」日の売上を比較します。これは、回復力の低いブランドに対して「強風の日」プロモーションを提供するなど、運用上の意思決定に役立ちます。

```
WITH TopWindiestLocations AS (
    SELECT TOP 3
        p.location_id
    FROM harmonized.tastybytes_poi_v AS p
    JOIN zts_weathersource.onpoint_id.history_day AS hd ON p.postal_code = hd.postal_code
    WHERE p.country = 'United States' AND YEAR(hd.date_valid_std) = 2022
    GROUP BY p.location_id, p.city, p.postal_code
    ORDER BY AVG(hd.max_wind_speed_100m_mph) DESC
)
SELECT
    o.truck_brand_name,
    ROUND(AVG(CASE WHEN hd.max_wind_speed_100m_mph <= 20 THEN o.order_total END), 2) AS
    avg_sales_calm_days,
```



```
ZEROIFNULL(ROUND(AVG(CASE WHEN hd.max_wind_speed_100m_mph > 20 THEN o.order_total
END), 2)) AS avg_sales_windy_days
FROM analytics.orders_v AS o
JOIN zts_weathersource.onpoint_id.history_day AS hd
  ON o.primary_city = hd.city_name AND DATE(o.order_ts) = hd.date_valid_std
WHERE o.location_id IN (SELECT location_id FROM TopWindiestLocations)
GROUP BY o.truck_brand_name
ORDER BY o.truck_brand_name;
```

Snowflake in Streamlit の紹介 (Introduction to Streamlit in Snowflake)



Streamlit

Streamlit は、機械学習やデータサイエンス向けの Web アプリケーションを簡単に作成して共有するために設計されたオープンソースの Python ライブラリです。データ駆動型アプリの迅速な開発と展開を可能にします。

Streamlit in Snowflake により、開発者は Snowflake 内で直接アプリケーションを安全に構築、展開、共有できます。この統合により、データやアプリケーションコードを外部システムに移動する必要なく、Snowflake に保存されたデータを処理および利用するアプリを構築できます。

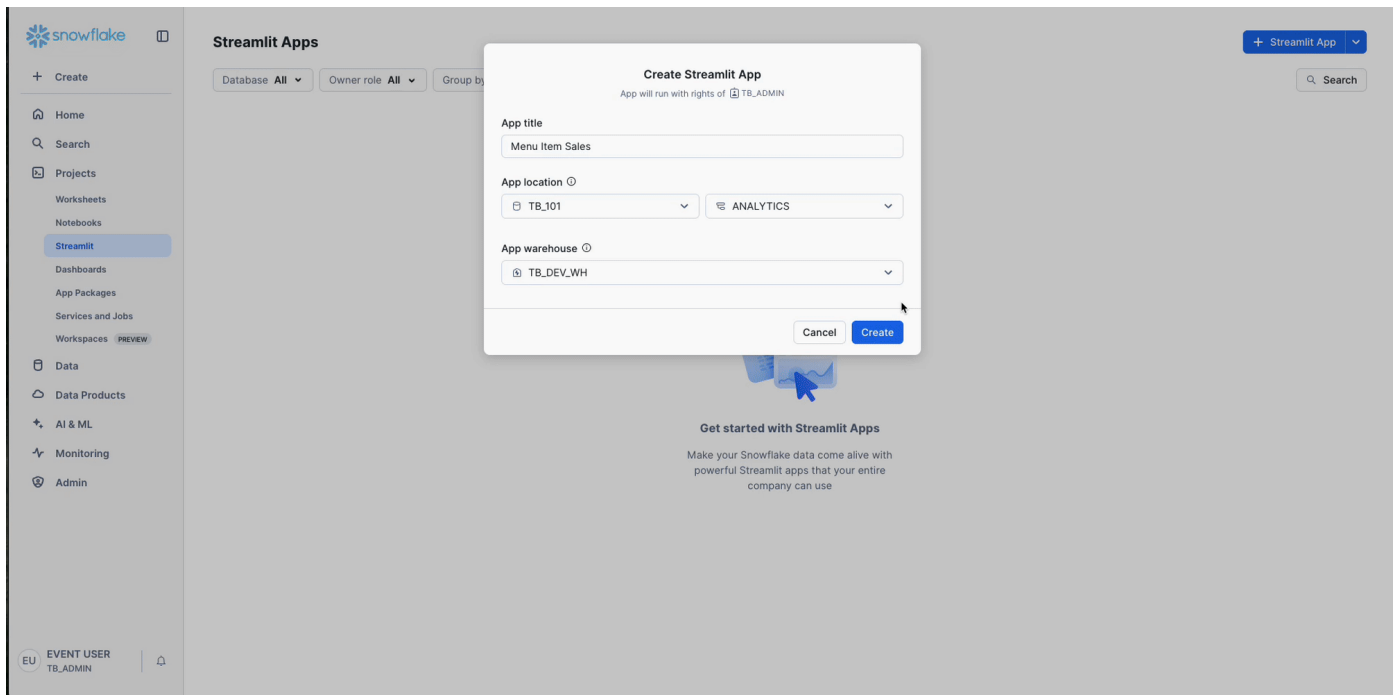
ステップ 1 - Streamlit アプリの作成

2022 年 2 月の日本の各メニュー項目の売上データを表示し、グラフ化する最初の Streamlit アプリを作成しましょう。

1. まず、**Projects** » **Streamlit** に移動し、右上の青い「+ Streamlit App」ボタンをクリックして新しいアプリを作成します。
2. 「Create Streamlit App」ポップアップに次の値を入力します:
 - App title: Menu Item Sales
 - App location:
 - Database: tb_101
 - Schema: Analytics
 - App warehouse: tb_dev_wh
3. 「Create」をクリックします。
アプリが最初に読み込まれると、右側のペインにサンプルアプリが表示され、左側のエディタペインにアプリのコードが表示されます。

4. すべてのコードを選択して削除します。

5. 次に、この [コード](#) をコピーして空白のエディタウィンドウに貼り付け、右上の「Run」をクリックします。



結論とリソース (Conclusion and Resources)

概要

おめでとうございます！Tasty Bytes - Zero to Snowflake の全行程を無事に完了しました。

ウェアハウスの構築と構成、データのクローン作成と変換、タイムトラベルによる削除されたテーブルの復元、半構造化データの自動データパイプラインの構築を行いました。また、単純な AISQL 関数で分析を生成し、Snowflake Copilot でワークフローを加速することで、AI を使用したインサイトを引き出しました。さらに、ロールとポリシーを使用した堅牢なガバナンスフレームワークを実装し、Snowflake マーケットプレイスからのライブデータセットで独自のデータをシームレスに強化しました。

このクイックスタートを再実行したい場合は、SQL ファイルの下部にある完全な **RESET** スクリプトを実行してください。

学習した内容

- **ウェアハウジングとパフォーマンス:** 仮想ウェアハウスを作成、管理、スケーリングし、Snowflake の結果キャッシュを活用する方法。
- **データ変換:** 安全な開発のためにゼロコピークローンを使用し、データを変換し、タイムトラベルと **UNDROP** を使用してエラーから即座に回復する方法。
- **データパイプライン:** 外部ステージからデータを取り込み、半構造化 **VARIANT** データを処理し、ダイナミックテーブルを使用して自動 ELT パイプラインを構築する方法。
- **Snowflake Cortex AI:** Snowflake Cortex AI を活用して顧客分析プラットフォームを構築する方法。
- **データガバナンス:** ロールベースのアクセス制御、自動 PII 分類、タグベースのデータマスキング、および行アクセスポリシーを使用してセキュリティフレームワークを実装する方法。

- **データコラボレーション:** Snowflake マーケットプレイスからライブのサードパーティデータセットを発見して取得し、独自のデータとシームレスに結合して新しいインサイトを生成する方法。

リソース

- [仮想ウェアハウスと設定](#)
- [リソースモニター](#)
- [予算](#)
- [ユニバーサル検索](#)
- [外部ステージからの取り込み](#)
- [半構造化データ](#)
- [ダイナミックテーブル](#)
- [ロールとアクセス制御](#)
- [タグベースの分類](#)
- [マスキングポリシーによる列レベルのセキュリティ](#)
- [行アクセスポリシーによる行レベルのセキュリティ](#)
- [データメトリック関数](#)
- [Trust Center](#)
- [データ共有](#)
- [Snowflake Cortex Playground](#)
- [Snowflake Cortex の AI SQL 関数](#)
- [Snowflake Cortex Search 概要](#)
- [Snowflake Cortex Analyst](#)