Case Study: Data Analysis on Subscription-based SaaS Company

Introduction

In this case study, I explore the complexities of analyzing subscription data for a simulated SaaS company. The project is designed to resemble real-world business scenarios, providing a platform to showcase my SQL skills and my ability to manage and analyze databases. The goal is to extract insights that can influence strategic business decisions.

Project Overview

In this project, I've constructed a simulated business environment by developing a series of interconnected tables, including Users, UserSubscriptions, and SubscriptionPlans. This relational database setup was meticulously designed to mirror real-world SaaS operations, facilitating an in-depth exploration of subscription dynamics. The arrangement of these tables allowed me to perform comprehensive analyses on subscription behaviors, such as acquisition trends and churn rates, providing a solid foundation for understanding key business metrics.

SQL Implementation and Data Engineering

Creating a Relational Database

The step involved establishing a robust relational database to model the relationships essential to a subscription-based business accurately:

- **Users Table**: Captures demographic and login information.
- **UserSubscriptions Table**: Links users to their respective subscription plans with start and end date details.
- **SubscriptionPlans Table**: Contains details about each plan, including cost and plan features, which are pivotal for revenue analysis.

These tables were meticulously designed with foreign keys and constraints to maintain data integrity and relational logic.

Users Table:

```
CREATE TABLE Users (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    DateOfBirth DATE,
    Country VARCHAR(50),
    RegistrationDate DATE
);
```

SubscriptionPlans Table:

```
CREATE TABLE SubscriptionPlans (
    PlanID INT AUTO_INCREMENT PRIMARY KEY,
    PlanName VARCHAR(100),
    Price DECIMAL(10, 2)
);
```

UserSubscription Table:

```
CREATE TABLE UserSubscriptions (
   SubscriptionID INT AUTO_INCREMENT PRIMARY KEY,
   UserID INT,
   PlanID INT,
   StartDate DATE,
   EndDate DATE NULLABLE,
   FOREIGN KEY (UserID) REFERENCES Users(UserID),
   FOREIGN KEY (PlanID) REFERENCES SubscriptionPlans(PlanID)
);
```

UserActivity Table:

```
CREATE TABLE UserSubscriptions (
   SubscriptionID INT AUTO_INCREMENT PRIMARY KEY,
   UserID INT,
   PlanID INT,
   StartDate DATE,
   EndDate DATE NULLABLE,
   FOREIGN KEY (UserID) REFERENCES Users(UserID),
   FOREIGN KEY (PlanID) REFERENCES SubscriptionPlans(PlanID)
);
```

Data Import and Data Integrity

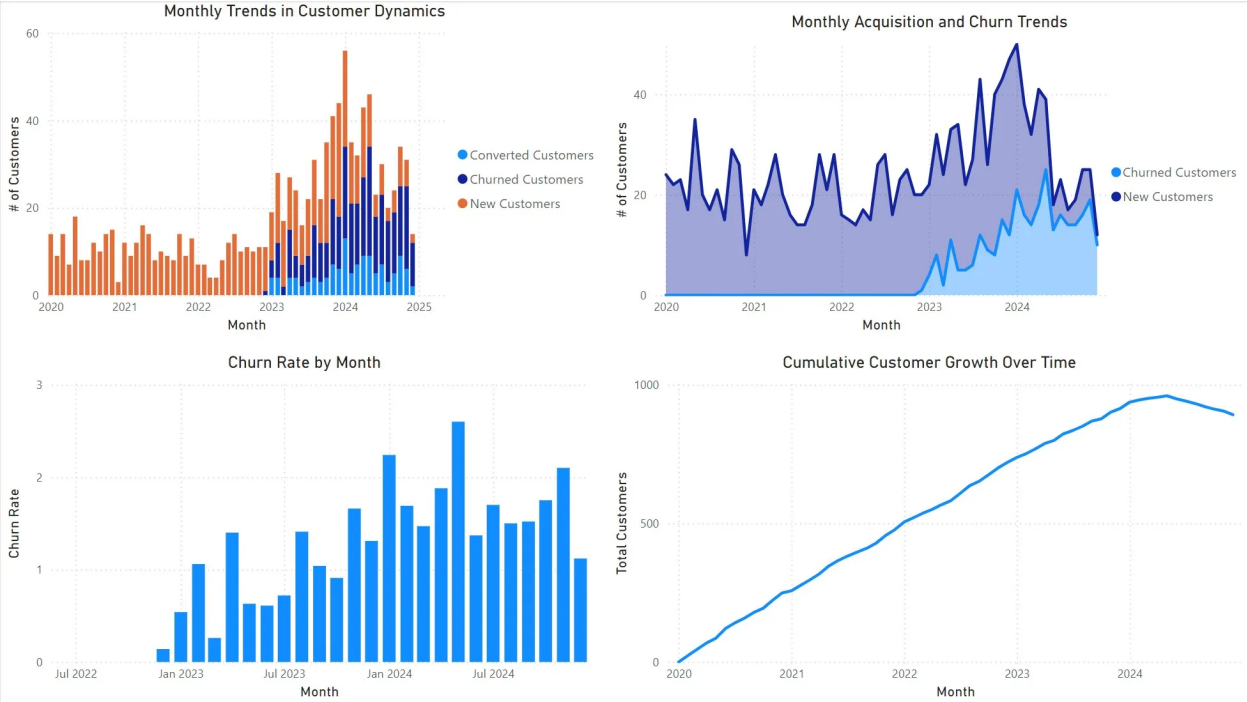
The main challenge with this project was creating a realistic simulation of data from a subscription-based SaaS environment. I used Mockaroo, a tool for generating realistic data sets, to face this potential challenge. It allowed me to tailor the data specifically for my project's needs, down to the SQL format for seamless integration.

Once data was imported, I performed integrity checks to confirm the data's validity. From checking relationships between tables to verifying key assignments, the database was populated precisely.

Analytical Insights and Business Impact

Monthly Churn Rates

In a subscription-based business model, understanding customer churn is crucial. This metric is pivotal in assessing the stickiness of the service and the effectiveness of retention strategies. My analysis involves tracking the number of customers who leave each month and comparing it with new sign-ups to understand overall business health and customer lifecycle trends.



This comprehensive dashboard visualizes critical metrics of customer dynamics, including churn rates, new sign-ups, and cumulative growth

This dashboard highlights key trends in customer dynamics. Starting in January 2023, we observe a noticeable increase in churned users, which suggests challenges in retaining customers. Despite a strong influx of new customers throughout 2023, the rising churn rate negatively impacts overall customer growth by early 2024. The curve showing cumulative customer growth begins to flatten, indicating that customer losses are starting to offset new sign-ups.

Code for Churn Rate:

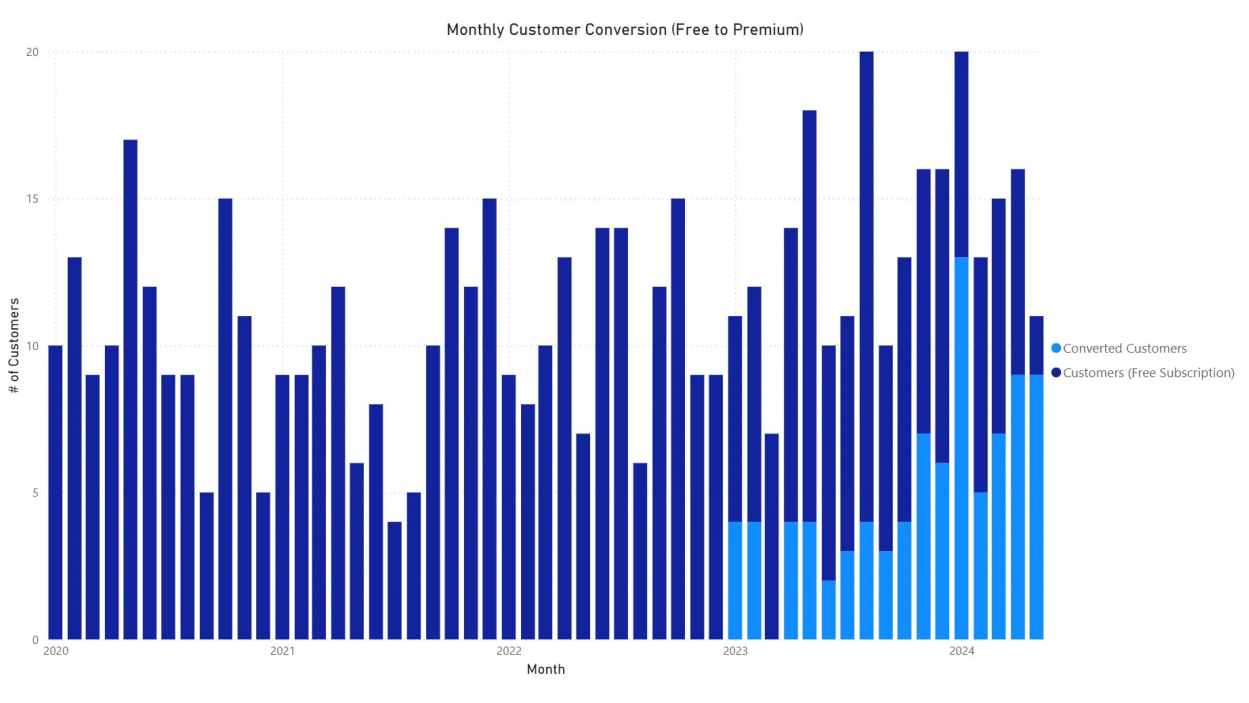
The SQL code below effectively gathers data to compute the churn rates by creating a comprehensive time series of all months in which either a new subscription starts or an existing one ends. It helps us track when customers drop off and how many new ones start each month.

```
WITH MonthSeries AS (
   SELECT DISTINCT DATE_FORMAT(StartDate, '%Y-%m') AS Month
   FROM UserSubscriptions
   UNION
   SELECT DISTINCT DATE_FORMAT(EndDate, '%Y-%m')
   FROM UserSubscriptions
   WHERE EndDate IS NOT NULL
),
Churned AS (
   SELECT
      DATE_FORMAT(EndDate, '%Y-%m') AS Month,
      COUNT(DISTINCT UserID) AS ChurnedCustomers
   FROM
```

```
    UserSubscriptions
  WHERE
    EndDate IS NOT NULL
  GROUP BY
    Month
),
NewCustomers AS (
  SELECT
    DATE_FORMAT(StartDate, '%Y-%m') AS Month,
    COUNT(DISTINCT UserID) AS NewCustomers
  FROM
    UserSubscriptions
  GROUP BY
    Month
)
SELECT
  m.Month,
  COALESCE(c.ChurnedCustomers, 0) AS ChurnedCustomers,
  COALESCE(n.NewCustomers, 0) AS NewCustomers
FROM
  MonthSeries m
  LEFT JOIN Churned c ON m.Month = c.Month
  LEFT JOIN NewCustomers n ON m.Month = n.Month
ORDER BY
  m.Month;
```

Monthly Conversion Rates

The examination of subscription data enabled the identification of monthly conversion trends, highlighting the rate at which customers transitioned from basic to premium plans. This analysis is crucial for understanding the effectiveness of different subscription models over time. By tracking these trends, the company can strategically enhance marketing efforts and tailor promotions to optimize conversion rates during periods historically showing higher potential for upgrades.



In this visual, we can see an upward trend in conversions toward the end of 2023.

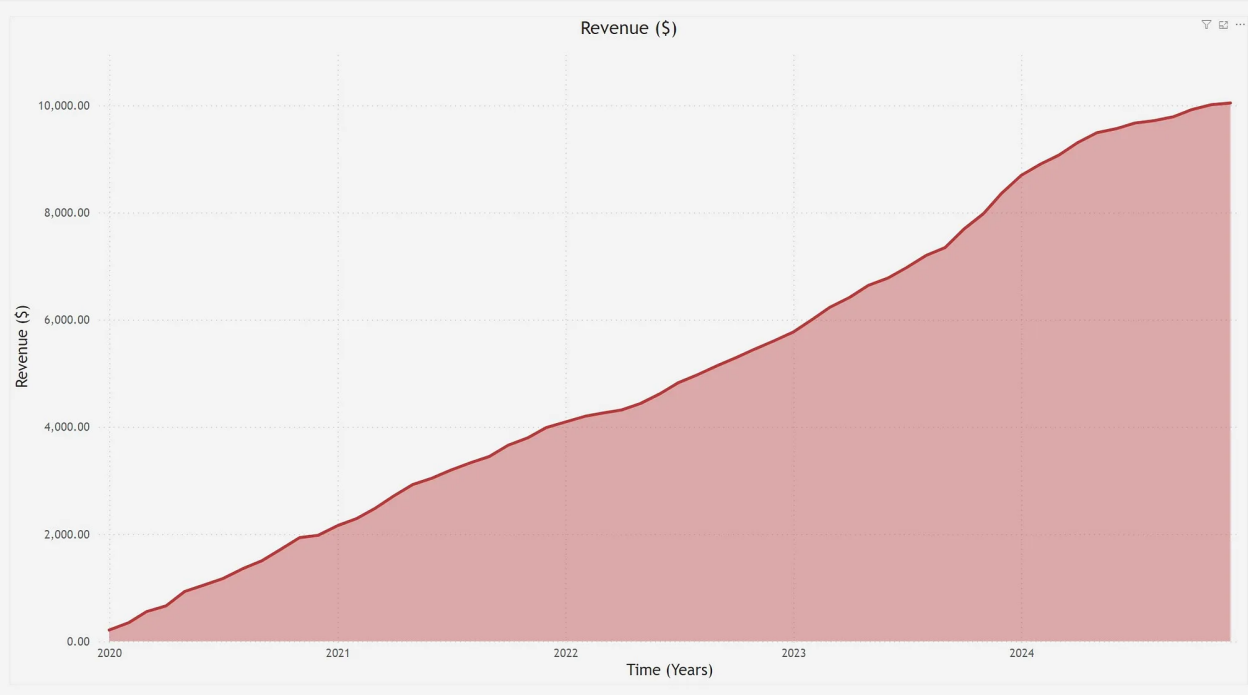
Code for Monthly Conversion Rates:

```
WITH MonthlyConversions AS (
    SELECT
        DATE_FORMAT(us1.EndDate, '%Y-%m') AS ConversionMonth,
        COUNT(DISTINCT us1.UserID) AS TotalConversions
    FROM
        UserSubscriptions us1
    JOIN
        UserSubscriptions us2 ON us1.UserID = us2.UserID AND us1.SubscriptionID <> us2.SubscriptionID
    WHERE
        us1.PlanID = 1 AND us2.PlanID = 2 -- Assuming 1 is the basic plan and 2 is the premium plan
        AND us1.EndDate IS NOT NULL
        AND us1.EndDate = us2.StartDate
    GROUP BY
        ConversionMonth
),
Months AS (
    SELECT DISTINCT DATE_FORMAT(StartDate, '%Y-%m') AS Month
    FROM UserSubscriptions
    UNION
    SELECT DISTINCT DATE_FORMAT(EndDate, '%Y-%m')
    FROM UserSubscriptions
    WHERE EndDate IS NOT NULL
)
SELECT
    m.Month,
    COALESCE(mc.TotalConversions, 0) AS TotalConversions
```

```
FROM
   Months m
   LEFT JOIN MonthlyConversions mc ON m.Month = mc.ConversionMonth
ORDER BY
   m.Month;
```

Revenue:

A key metric of a subscription-based SaaS company's performance is its ability to generate consistent and growing revenue. This aspect of the analysis focuses on the monthly revenue trends, which are crucial for understanding the financial trends of the company.



While there are fluctuations, there is an observable upward trend in revenue through mid-2023, indicating that the business was experiencing growth during this period. However, the sharp declines and peaks post-2023 suggest that the company might be facing increased volatility in its revenue streams, which could be due to various external or internal factors.

Code for Monthly Revenue:

```
WITH MonthlySignups AS (
   SELECT
      DATE_FORMAT(us.StartDate, '%Y-%m') AS YearMonth,
      COUNT(DISTINCT us.UserID) AS NewPremiumSignups
   FROM
      UserSubscriptions us
   WHERE
      us.PlanID = 2 AND us.StartDate IS NOT NULL
```

```
    GROUP BY
        YearMonth
),
CumulativeRevenue AS (
    SELECT
        m.YearMonth,
        ROUND(SUM(ms.NewPremiumSignups) OVER (ORDER BY m.YearMonth) * 14.99, 2) AS CumulativeRevenue
    FROM (
        SELECT DISTINCT
            DATE_FORMAT(StartDate, '%Y-%m') AS YearMonth
        FROM
            UserSubscriptions
        WHERE
            StartDate IS NOT NULL
        UNION DISTINCT
        SELECT DISTINCT
            DATE_FORMAT(EndDate, '%Y-%m')
        FROM
            UserSubscriptions
        WHERE
            EndDate IS NOT NULL
    ) m
    LEFT JOIN MonthlySignups ms ON m.YearMonth = ms.YearMonth
)
SELECT
    cr.YearMonth,
    cr.CumulativeRevenue
FROM
    CumulativeRevenue cr
ORDER BY
    cr.YearMonth;
```

Conclusion

Throughout this project, I have demonstrated a comprehensive approach to setting up and analyzing a relational database that reflects a real-world business environment. I crafted a detailed schema in MySQL to manage subscription data and employed SQL for querying and data manipulation, ensuring accuracy and integrity. Visualizations were created in Power BI to effectively communicate trends in customer behavior, conversion rates, and revenue fluctuations. This project showcased my capabilities in using data tools and SQL to derive actionable business insights, reinforcing my skills as a data analyst.