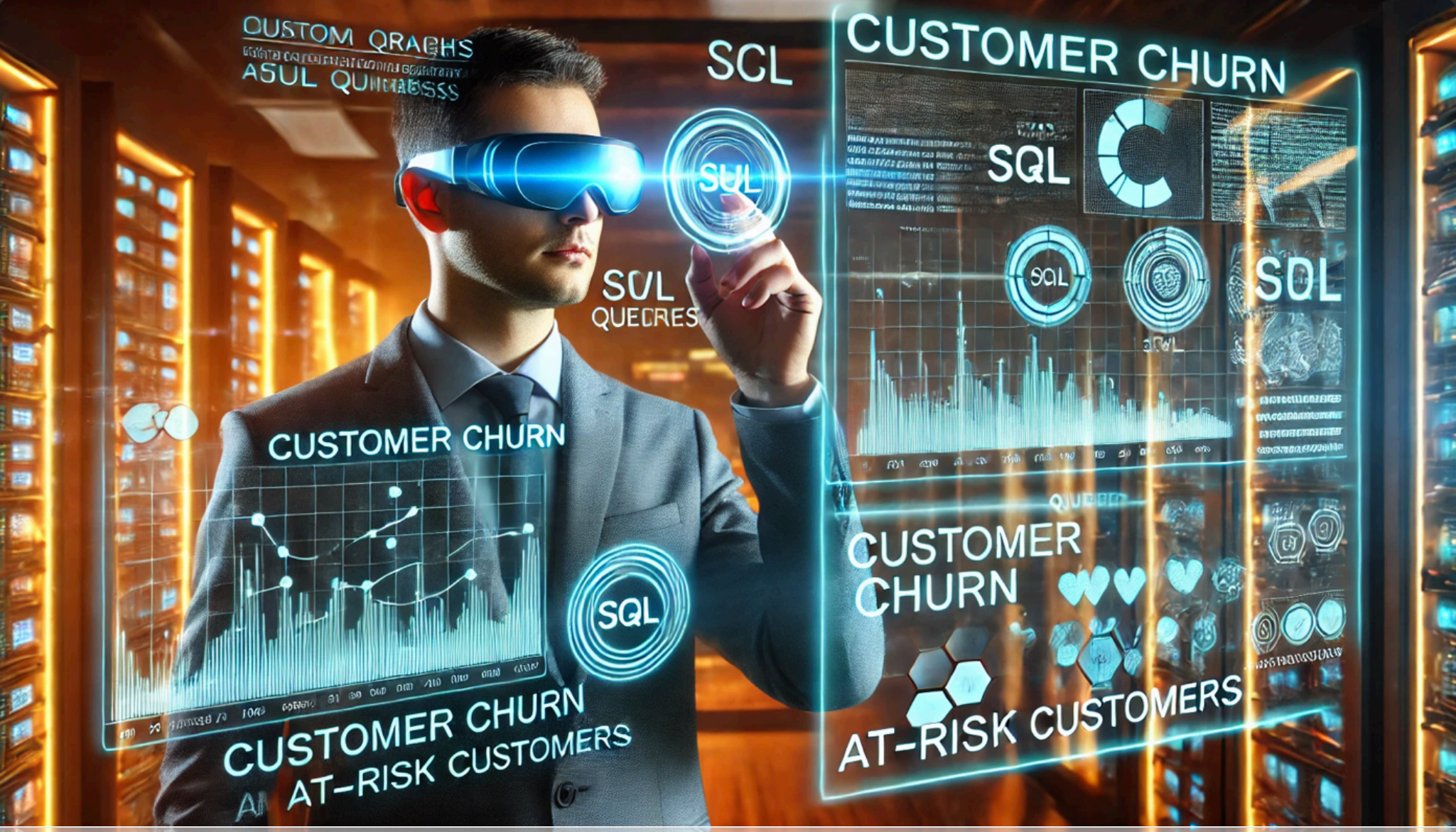
Ahmed Abdulwahid

Churn Analysis with SQL 🤫

Predict & Prevent Customer Loss! ☠️

#DataGenius



Losing customers is a nightmare for any business. 💀 *Every lost user = lost revenue* 💸*. But what if you could predict who's about to churn and stop it before it happens? That's the power of SQL in churn analysis!*

🧐 What Is Customer Churn?

📌 **Customer churn (or attrition) happens when a customer stops using a service within a given time.**

💡 *Examples of Churn Across Industries:*

- 🎬 *Streaming Services (Netflix, Spotify) → A user cancels their subscription.*
- 🛒 *E-commerce (Amazon, Shopify) → A customer hasn't purchased in 6+ months.*
- 💻 *SaaS Products (HubSpot, Zoom) → A company stops renewing its plan.*
- 📞 *Telecom (Verizon, Vodafone) → A customer switches to another provider.*

⚠️ Why does churn matter?

- 👉 *Acquiring a new customer costs 5x more than retaining an existing one.* 😱
- 👉 *A 5% increase in retention can boost profits by 25–95%!* 📈
- 👉 *Churn analysis helps businesses take action before customers leave!* 🚀

Let’s deep dive into how to detect, analyze, and prevent churn using SQL with practical examples and insights! 🧐✨

🏗️ SQL Dataset Setup: The Foundation of Churn Analysis

Let’s assume we have a customers table and a transactions table:

```sql
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100),
    signup_date DATE,
    last_login DATE,
    subscription_status VARCHAR(10) -- 'active' or 'churned'
);

CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY,
    customer_id INT,
    purchase_date DATE,
    amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

💡 Key Tables:

- 🔶 **customers**: Stores signup info, last login, and status.
- 🔶 **transactions**: Tracks all purchases.

🔍 Identifying Churned Customers with SQL

One simple way to define churn is customers who haven't made a purchase in X days. Let's check for customers who haven't bought anything in the last 90 days:

```sql
SELECT c.customer_id, c.name, MAX(t.purchase_date) AS last_purchase_date
FROM customers c
LEFT JOIN transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id, c.name
HAVING MAX(t.purchase_date) < CURRENT_DATE - INTERVAL '90' DAY;
```

💡 What this does:

🔹 *Finds the last purchase date for each customer.*

🔹 *Checks if it's been more than 90 days since their last transaction.*

🔹*Identifies customers at high risk of churn!* 🚨

📊 Churn Rate Calculation 🔍

How many customers are churning each month?

```sql
SELECT
    EXTRACT(MONTH FROM signup_date) AS signup_month,
    COUNT(*) AS total_customers,
    SUM(CASE WHEN subscription_status = 'churned'
             THEN 1 ELSE 0 END) AS churned_customers,
    ROUND((SUM(CASE WHEN subscription_status = 'churned'
                    THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) AS churn_rate
FROM customers
GROUP BY signup_month
ORDER BY signup_month;
```

💡 What this does:

- *Groups customers by the month they signed up.* 📆
- *Counts how many have churned.*
- *Calculates the churn rate (%) per month.* 📉

💡 Churn Rate Formula:

$$\text{Churn Rate} = \left(\frac{\text{Churned Customers}}{\text{Total Customers}}\right) \times 100$$

🔥 Spotting At-Risk Customers Before They Leave 🚨

We can also detect early signs of churn by looking at inactive users:

```sql
SELECT customer_id, name, last_login
FROM customers
WHERE last_login < CURRENT_DATE - INTERVAL '30' DAY
AND subscription_status = 'active';
```

💡 Key Insight:

🚨 *Customers who haven't logged in for 30+ days may be considering leaving. Reach out to them!* 📩

🎯 Churn Prediction Using SQL

Let's get fancy! 🤩 We'll create a churn risk score based on:

⚠️ No purchases in the last 90 days 🛑

⚠️ No login activity in 30+ days ⏳

⚠️ Low total spending 💰

```sql
SELECT
    c.customer_id,
    c.name,
    MAX(t.purchase_date) AS last_purchase,
    c.last_login,
    COUNT(t.transaction_id) AS total_purchases,
    SUM(t.amount) AS total_spent,
    CASE
        WHEN MAX(t.purchase_date) < CURRENT_DATE - INTERVAL '90' DAY
        THEN 'HIGH RISK 🚨'
        WHEN c.last_login < CURRENT_DATE - INTERVAL '30' DAY
        THEN 'MEDIUM RISK ⚠️'
        ELSE 'LOW RISK ✅'
    END AS churn_risk
FROM customers c
LEFT JOIN transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id, c.name, c.last_login;
```

💡 What this does:

🔹 *Assigns a churn risk level to each customer.*

🔹 *Helps businesses prioritize retention efforts.*

🚀 High-risk customers should get re-engagement emails ASAP! 📩💥

📣 How to Reduce Churn (Data-Driven Strategies)

Now that we've identified churn, how do we reduce it?

- 🔥 *Personalized Retention Emails → Offer discounts, special offers, or content based on purchase history.*
- 📣 *Customer Segmentation → Identify at-risk customers and target them with retention campaigns.*
- 📊 *Loyalty Programs → Reward frequent customers to keep them engaged.*
- 🛠️ *Improve User Experience → Identify friction points in your service and fix them!*

SQL can help with all of this by analyzing customer behavior patterns! 💡

🎯 Final Thoughts: Why SQL Is a Churn-Fighting Superpower

SQL is an essential skill for churn analysis. By using SQL, businesses can:

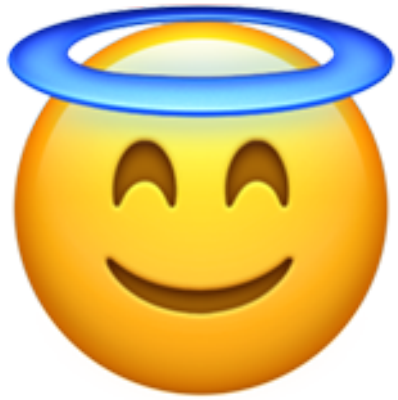
✅ *Identify churn patterns* 📊

✅ *Segment customers based on risk* 🎯

✅ *Take action before losing revenue* 💰

📌 **Next Steps? Implement churn analysis with SQL in your business and watch your customer retention soar!** 🚀

Repost it



Thank you