

Registrations and active users

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

User-centric KPIs

KPIs

- Registrations
- Active users
- Growth
- Retention

Benefits

- Measure performance well in B2Cs
- Used by investors to assess pre-revenue and -profit startups

Registrations - overview

- **Registration:** When a user first signs up for an account on Delivr through its app
- **Registrations KPI:** Counts registrations over time, usually per month
 - Good at measuring a company's success in attracting new users
- For Delivr, a user's registration date is the date of that user's first order

Registrations - setup

Query

```
SELECT
  user_id,
  MIN(order_date) AS reg_date
FROM orders
GROUP BY user_id
ORDER BY user_id
LIMIT 3;
```

Result

user_id	reg_date
-----	-----
0	2018-06-01
1	2018-06-01
2	2018-06-01

Registrations - query

```
WITH reg_dates AS (  
  SELECT  
    user_id,  
    MIN(order_date) AS reg_date  
  FROM orders  
  GROUP BY user_id)  
  
SELECT  
  DATE_TRUNC('month', reg_date) :: DATE AS delivr_month,  
  COUNT(DISTINCT user_id) AS regs  
FROM reg_dates  
GROUP BY delivr_month  
ORDER BY delivr_month ASC  
LIMIT 3;
```

Registrations - result

Result

delivr_month	regs
-----	----
2018-06-01	123
2018-07-01	140
2018-08-01	157

Active users - overview

- **Active users KPI:** Counts the active users of a company's app over a time period
 - by day (daily active users, or DAU)
 - by month (monthly active users, or MAU)
- Stickiness (DAU / MAU), measures how often users engage with an app on average
 - **Example:** If Deliver's stickiness is $\text{DAU} / \text{MAU} = 0.3$ (30%), users use Deliver for $30\% \times 30$ days = 9 days each month on average

Active users - query

```
SELECT
  DATE_TRUNC('month', order_date) :: DATE AS delivr_month,
  COUNT(DISTINCT user_id) AS mau
FROM orders
GROUP BY delivr_month
ORDER BY delivr_month ASC
LIMIT 3;
```

delivr_month	mau
-----	---
2018-06-01	123
2018-07-01	226
2018-08-01	337

Registrations and active users

ANALYZING BUSINESS DATA IN SQL

Window functions

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

Window functions - overview

- **Window functions:** Perform an operation across a set of rows related to the current row
- **Examples**
 - Calculate a running total
 - Fetch the value of a previous or following row

Running total

Running total: A cumulative sum of a variable's previous values

Example

```
x    x_rt
---  ----
1    1
2    3
3    6
4   11
5   16
```

Registrations running total - query

```
WITH reg_dates AS (  
  SELECT  
    user_id,  
    MIN(order_date) AS reg_date  
  FROM orders  
  GROUP BY user_id),  
registrations AS (  
  SELECT  
    DATE_TRUNC('month', reg_date) :: DATE AS delivr_month,  
    COUNT(DISTINCT user_id) AS regs  
  FROM reg_dates  
  GROUP BY delivr_month)  
  
SELECT  
  delivr_month,  
  regs,  
  SUM(regs) OVER (ORDER BY delivr_month ASC) AS regs_rt  
FROM registrations  
ORDER BY delivr_month ASC LIMIT 3;
```

Registrations running total - result

delivr_month	regs	regs_rt
-----	----	-----
2018-06-01	123	123
2018-07-01	140	263
2018-08-01	157	420

Lagged MAU - query

```
WITH maus AS (  
  SELECT  
    DATE_TRUNC('month', order_date) :: DATE AS deliver_month,  
    COUNT(DISTINCT user_id) AS mau  
  FROM orders  
  GROUP BY deliver_month)  
  
SELECT  
  deliver_month,  
  mau,  
  COALESCE(  
    LAG(mau) OVER (ORDER BY deliver_month ASC),  
    1) AS last_mau  
FROM maus  
ORDER BY deliver_month ASC  
LIMIT 3;
```

Lagged MAU - result

delivr_month	mau	last_mau
-----	---	-----
2018-06-01	123	1
2018-07-01	226	123
2018-08-01	337	226

Window functions

ANALYZING BUSINESS DATA IN SQL

Growth rate

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

Deltas - query

```
WITH maus AS (  
  SELECT  
    DATE_TRUNC('month', order_date) :: DATE AS deliver_month,  
    COUNT(DISTINCT user_id) AS mau  
  FROM orders  
  GROUP BY deliver_month),  
maus_lag AS (  
  SELECT  
    deliver_month,  
    mau,  
    COALESCE(  
      LAG(mau) OVER (ORDER BY deliver_month ASC),  
      1) AS last_mau  
  FROM maus)
```

Deltas - result

Query

```
WITH maus AS (...),
     maus_lag AS (...)

SELECT
  deliver_month,
  mau,
  mau - last_mau AS mau_delta
FROM maus_lag
ORDER BY deliver_month
LIMIT 3;
```

Result

deliver_month	mau	mau_delta
-----	---	-----
2018-06-01	123	1
2018-07-01	226	103
2018-08-01	337	111

Deltas - pitfalls

- Raw, absolute number
- Only shows one of three things about a variable
 - Decreasing if $\delta < 0$
 - Stable if $\delta = 0$
 - Increasing if $\delta > 0$

Growth rate - overview

- **Growth rate:** A percentage that show the change in a variable over time relative to that variable's initial value
- **Formula:** $\frac{\text{Current value} - \text{Previous value}}{\text{Previous value}}$
- **Example:** $\frac{67-50}{50} = 0.34 = 34\%$

Growth rate - query

Query

```
WITH maus AS (...),
     maus_lag AS (...)

SELECT
  deliver_month,
  mau,
  ROUND(
    (mau - last_mau) :: NUMERIC / last_mau,
    2) AS growth
FROM maus_lag
ORDER BY deliver_month
LIMIT 3;
```

Result

deliver_month	mau	growth
-----	---	-----
2018-06-01	123	122.00
2018-07-01	226	0.84
2018-08-01	337	0.49

Growth

ANALYZING BUSINESS DATA IN SQL

Retention

ANALYZING BUSINESS DATA IN SQL

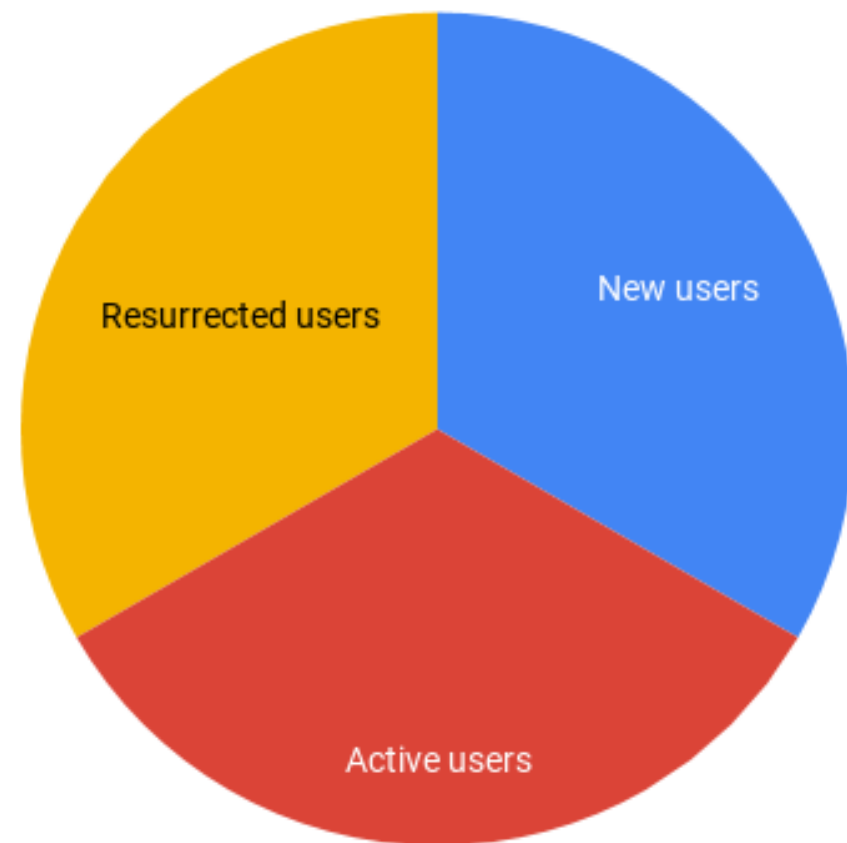


Michel Semaan
Data Scientist

MAU - pitfalls

- Doesn't show the breakdown of active users by tenure
- Doesn't distinguish between different patterns of user activity
 - **Case 1:** 100 users register every month, and are active for one month only
 - **Case 2:** Only 100 users register in the first month, and no one ever registers after, but these 100 users are active every single month
 - **Both cases' MAUs will be 100!**

MAU - breakdown



Breakdown

- New users joined this month
- Retained users were active in the previous month, and stayed active this month
- Resurrected users weren't active in the previous month, but returned to activity this month

Retention rate - overview

- **Retention rate:** A percentage measuring how many users who were active in a previous month are still active in the current month
- **Formula:** $\frac{U_c}{U_p}$, where U_c is the count of distinct users who were active in both the current and previous months, and U_p is the count of distinct users who were active in the previous period
- **Example:** $\frac{80}{100} = 0.8 = 80\%$

Retention rate - query

```
WITH user_activity AS (  
  SELECT DISTINCT  
    DATE_TRUNC('month', order_date) :: DATE AS delivr_month,  
    user_id  
  FROM orders)  
  
SELECT  
  previous.delivr_month,  
  ROUND(  
    COUNT(DISTINCT current.user_id) :: NUMERIC /  
    GREATEST(COUNT(DISTINCT previous.user_id), 1),  
    2) AS retention  
FROM user_activity AS previous  
LEFT JOIN user_activity AS current  
ON previous.user_id = current.user_id  
AND previous.delivr_month = (current.delivr_month - INTERVAL '1 month')  
GROUP BY previous.delivr_month  
ORDER BY previous.delivr_month ASC  
LIMIT 3;
```

Retention rate - result

delivr_month	retention
2018-06-01	0.70
2018-07-01	0.70
2018-08-01	0.76

Retention

ANALYZING BUSINESS DATA IN SQL