04/11/2025

Dmytro Hushchin | Storyby

SQL for Data Insights

Dmytro Hushchin

2025 – Analytics Team Lead @Storyby

2024 – Master’s student in BFE @KSE

2023 – Product Analyst @Storyby

2022 – Genesis SE / IT School Alumni

Dmytro Hushchin

🥋 5 years of judo – back to it now

📚 Anime & manga fan

✏️ Sketching in my free time

Dmytro Hushchin

- 🥋 5 years of judo – back to it now
- 📚 Anime & manga fan
- ✏️ Sketching in my free time
- 🗿 A $5,000 lesson in SQL...

STORYBY

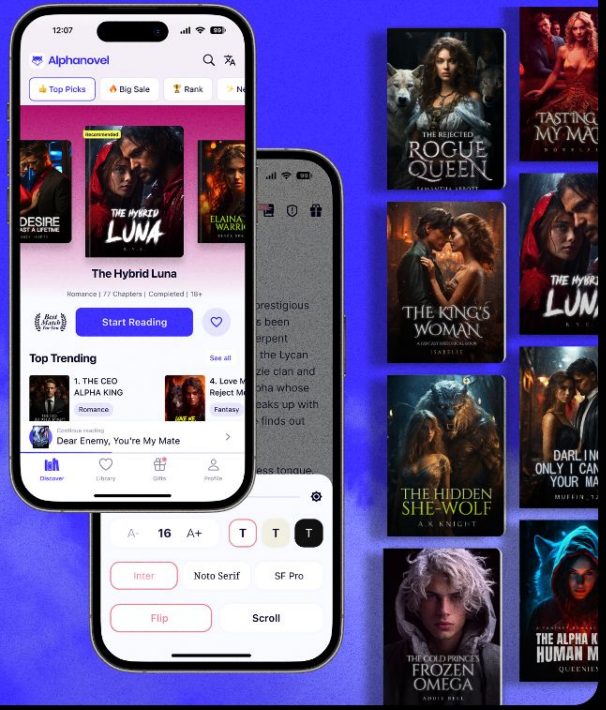
Discovering rising stars

Storyby is a tech company focused on discovering rising star authors and offering their stories to read, watch, and experience through engaging content.

Alphanovel

AlphaNovel - Read novels

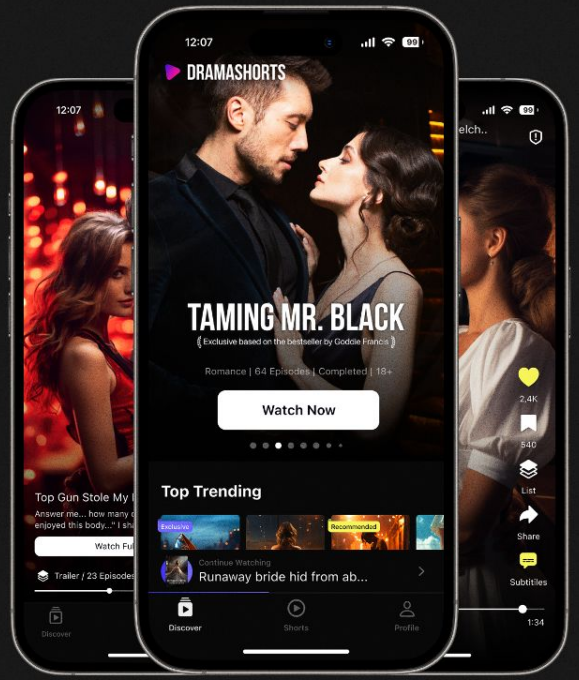
Explore captivating novels, enjoy suspenseful chapters, and level up with our gamification system. Plus, our bonus system adds extra fun – collect points and unlock rewards as you read.



DRAMASHORTS

DramaShort - Watch short movies

Immersive movie world of dynamic storytelling with film series unfolding in bite-sized episodes inspired by the best of AlphaNovel and literary masterpieces.



Agenda

Intro
Try to be short

Basic SQL
Commands, aggregation, joins

Advanced SQL
Window functions, optimisation

SQL in practice
Real business case

Q&A
It’s your time! 🎤

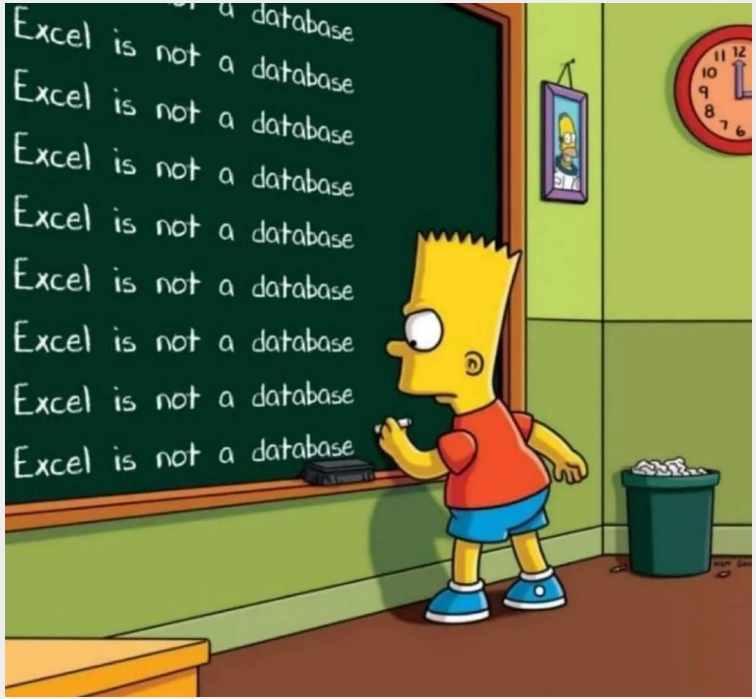
What is your experience level with SQL?

Which database do you work with most often?

Intro

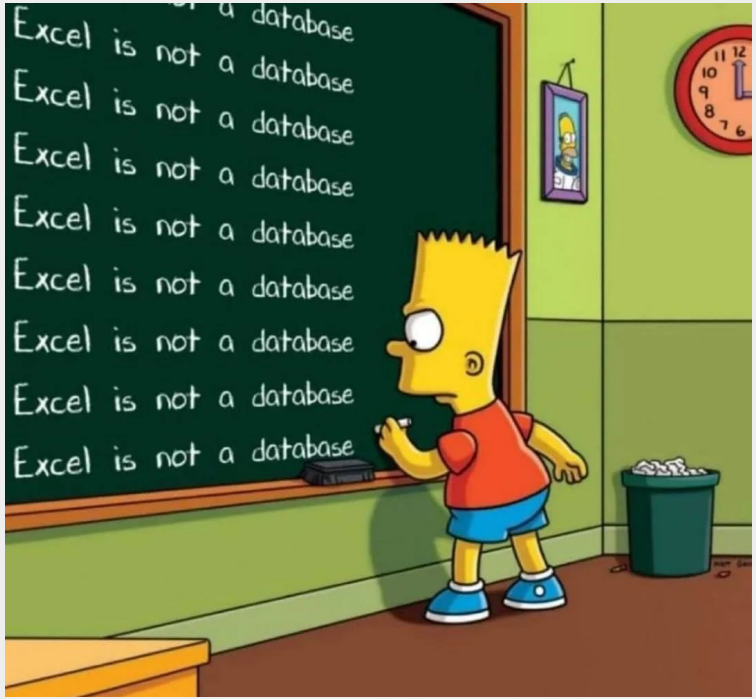
Analytics enables businesses to make data-driven decisions based on quantitative insights rather than intuition.

/Intro



Data is stored in tables

/Intro



Data is stored in tables

PostgreSQL

SQLite

SQL

MySQL

Google Big Query

/Intro

Structured Query Language (SQL) is a language used to query and manage data in databases

/BigQuery Setup

1. Go to Google Cloud Console

2. Create a new dataset

3. Create tables `users` and `logins`

4. You're ready!

Basic SQL

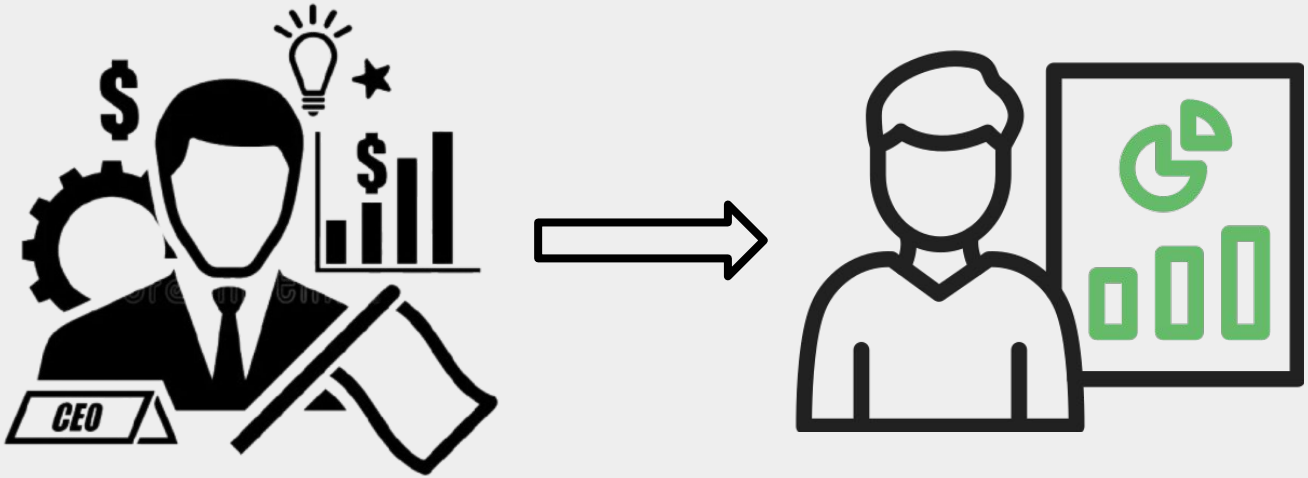
Basic SELECT

template

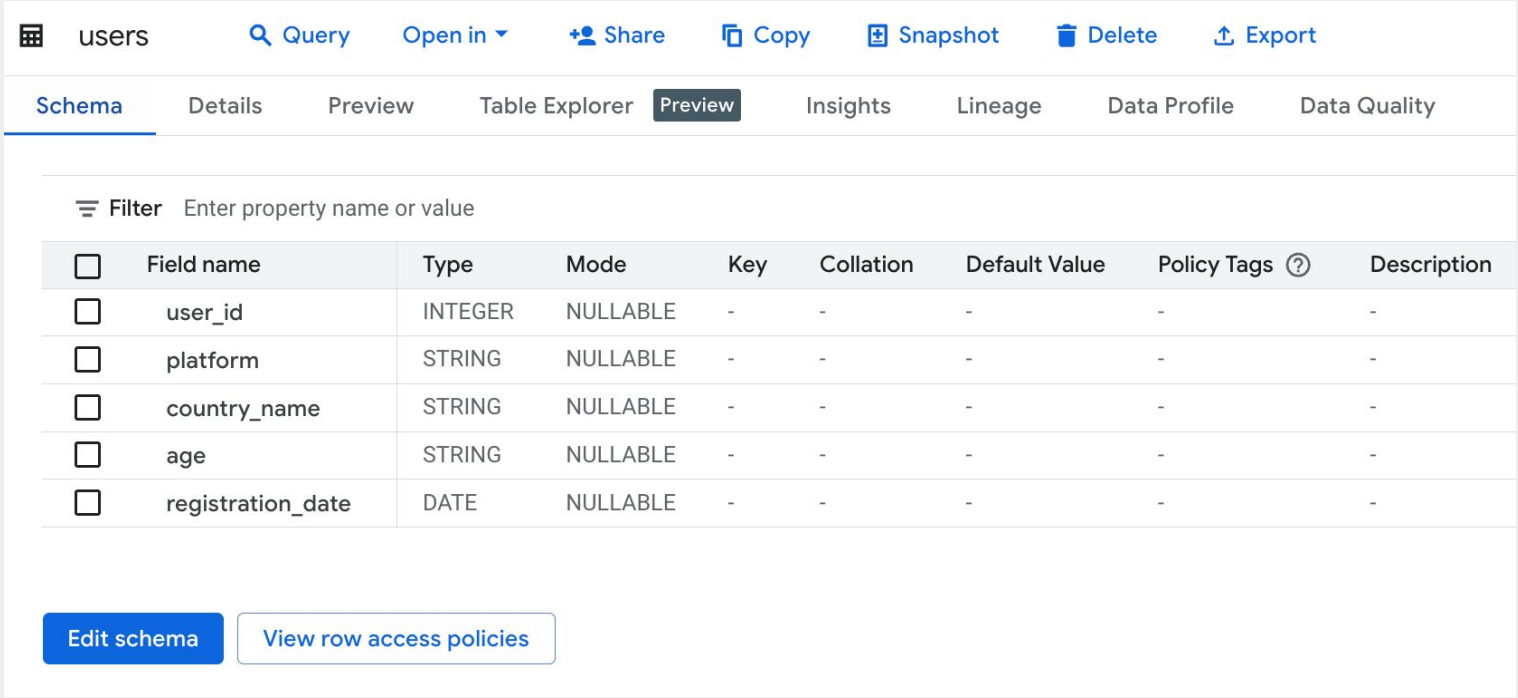
```
SELECT column
FROM table
WHERE condition
```

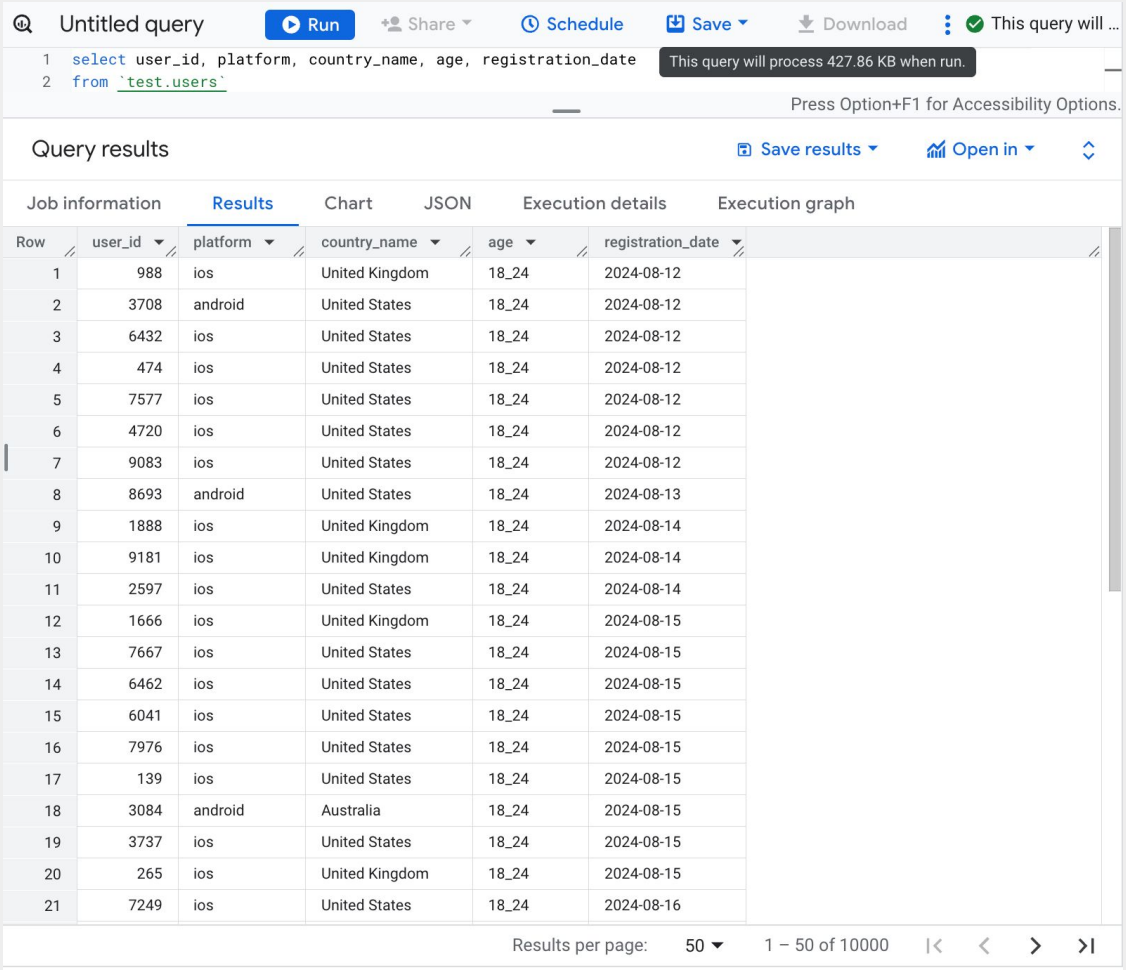
Case

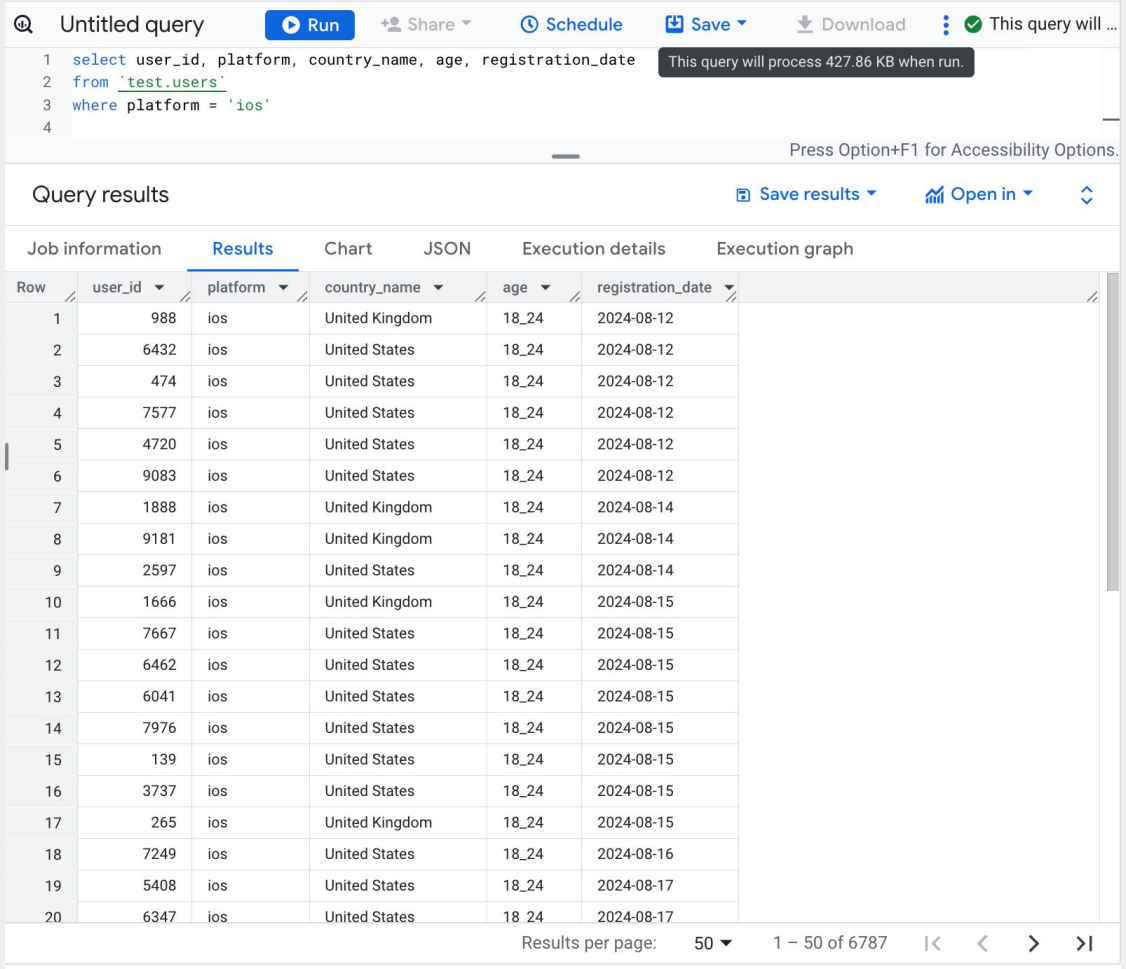
“What are the top 3 countries on iOS by number of users, from highest to lowest”



Schema


users
Query
Open in
Share
Copy
Snapshot
Delete
Export
Schema
Details
Preview
Table Explorer
Preview
Insights
Lineage
Data Profile
Data Quality
Filter
Enter property name or value
Field name
Type
Mode
Key
Collation
Default Value
Policy Tags
Description
user_id
INTEGER
NULLABLE
platform
STRING
NULLABLE
country_name
STRING
NULLABLE
age
STRING
NULLABLE
registration_date
DATE
NULLABLE
Edit schema
View row access policies


Untitled query
Run
Share
Schedule
Save
Download
This query will ...
1 select user_id, platform, country_name, age, registration_date
2 from `test.users`
This query will process 427.86 KB when run.
Press Option+F1 for Accessibility Options.
Query results
Save results
Open in
Job information
Results
Chart
JSON
Execution details
Execution graph
Row user_id platform country_name age registration_date
1 988 ios United Kingdom 18_24 2024-08-12
2 3708 android United States 18_24 2024-08-12
3 6432 ios United States 18_24 2024-08-12
4 474 ios United States 18_24 2024-08-12
5 7577 ios United States 18_24 2024-08-12
6 4720 ios United States 18_24 2024-08-12
7 9083 ios United States 18_24 2024-08-12
8 8693 android United States 18_24 2024-08-13
9 1888 ios United Kingdom 18_24 2024-08-14
10 9181 ios United Kingdom 18_24 2024-08-14
11 2597 ios United States 18_24 2024-08-14
12 1666 ios United Kingdom 18_24 2024-08-15
13 7667 ios United States 18_24 2024-08-15
14 6462 ios United States 18_24 2024-08-15
15 6041 ios United States 18_24 2024-08-15
16 7976 ios United States 18_24 2024-08-15
17 139 ios United States 18_24 2024-08-15
18 3084 android Australia 18_24 2024-08-15
19 3737 ios United States 18_24 2024-08-15
20 265 ios United Kingdom 18_24 2024-08-15
21 7249 ios United States 18_24 2024-08-16
Results per page: 50
1 – 50 of 10000


Untitled query
Run
Share
Schedule
Save
Download
This query will ...
1 select user_id, platform, country_name, age, registration_date
2 from `test.users`
3 where platform = 'ios'
4
This query will process 427.86 KB when run.
Press Option+F1 for Accessibility Options.
Query results
Save results
Open in
Job information
Results
Chart
JSON
Execution details
Execution graph
Row user_id platform country_name age registration_date
1 988 ios United Kingdom 18_24 2024-08-12
2 6432 ios United States 18_24 2024-08-12
3 474 ios United States 18_24 2024-08-12
4 7577 ios United States 18_24 2024-08-12
5 4720 ios United States 18_24 2024-08-12
6 9083 ios United States 18_24 2024-08-12
7 1888 ios United Kingdom 18_24 2024-08-14
8 9181 ios United Kingdom 18_24 2024-08-14
9 2597 ios United States 18_24 2024-08-14
10 1666 ios United Kingdom 18_24 2024-08-15
11 7667 ios United States 18_24 2024-08-15
12 6462 ios United States 18_24 2024-08-15
13 6041 ios United States 18_24 2024-08-15
14 7976 ios United States 18_24 2024-08-15
15 139 ios United States 18_24 2024-08-15
16 3737 ios United States 18_24 2024-08-15
17 265 ios United Kingdom 18_24 2024-08-15
18 7249 ios United States 18_24 2024-08-16
19 5408 ios United States 18_24 2024-08-17
20 6347 ios United States 18_24 2024-08-17
Results per page: 50
1 – 50 of 6787

Untitled query

Run

Share

```
1  select distinct country_name
2  from `test.users`
3  where platform = 'ios'
4
```

Query results

Job information | Results | Chart | JSON

Row	country_name
1	United Kingdom
2	United States
3	Australia
4	Canada
5	New Zealand

Basic SELECT

template

```
SELECT column
FROM table
WHERE condition
```

example

```
SELECT country_name
FROM `test.users`
WHERE platform = ‘ios’
```

```
1  select count(distinct user_id) as unique_users
2  from `test.users`
```

Query completed

Query results

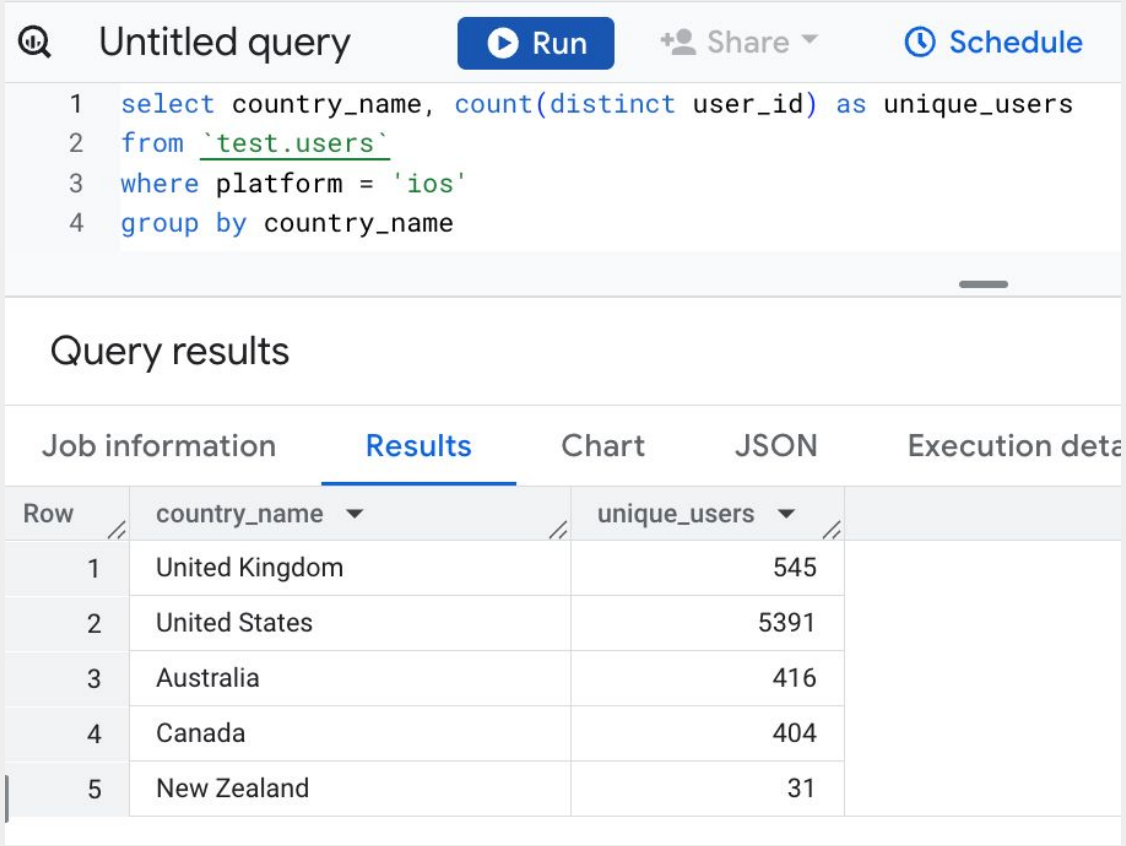
Job information | **Results** | Chart | JSON

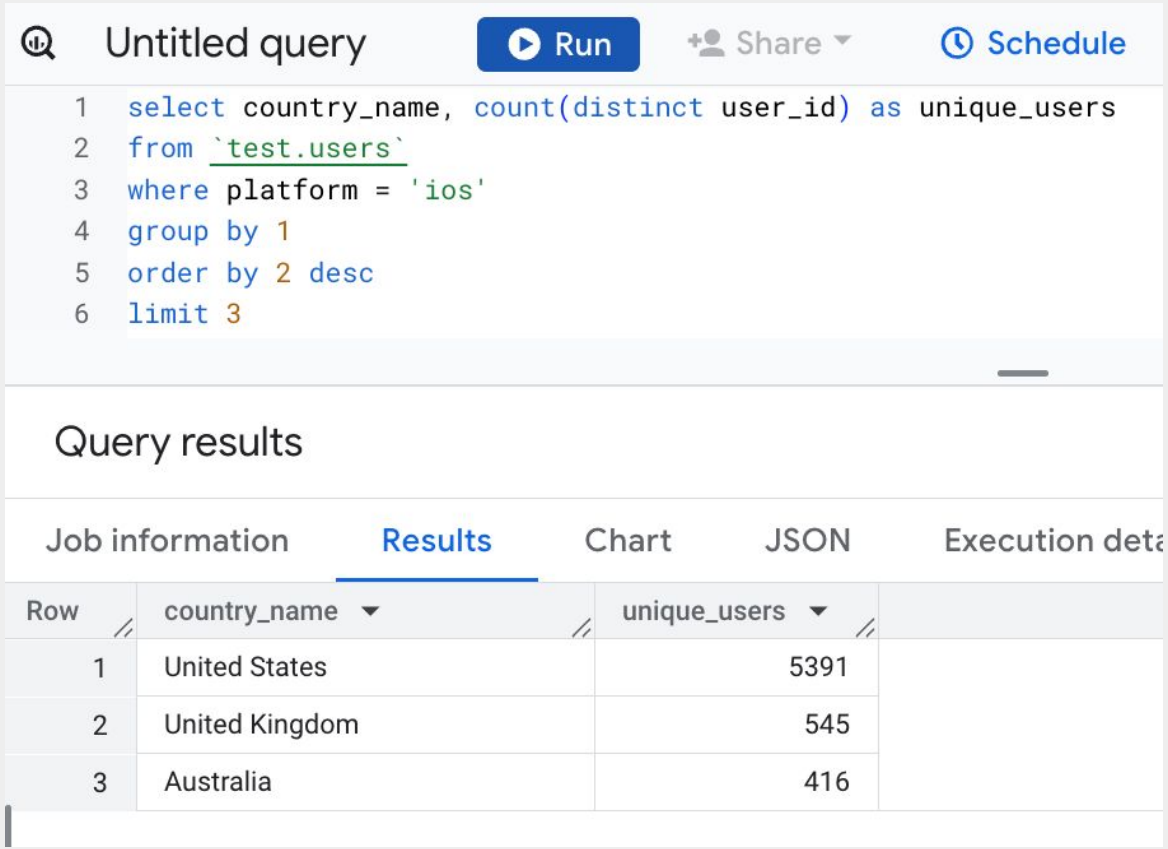
Row	unique_users
1	10000

Aggregation

template

```
SELECT column,
            COUNT(1) AS cnt
FROM table
WHERE condition
GROUP BY column
ORDER BY cnt DESC
LIMIT rows
```


Untitled query
Run
Share
Schedule
1 select country_name, count(distinct user_id) as unique_users
2 from `test.users`
3 where platform = 'ios'
4 group by country_name
Query results
Job information
Results
Chart
JSON
Execution deta
Row
country_name
unique_users
1 United Kingdom 545
2 United States 5391
3 Australia 416
4 Canada 404
5 New Zealand 31


Untitled query
Run
Share
Schedule
1 select country_name, count(distinct user_id) as unique_users
2 from `test.users`
3 where platform = 'ios'
4 group by 1
5 order by 2 desc
6 limit 3
Query results
Job information
Results
Chart
JSON
Execution deta
Row
country_name
unique_users
1 United States 5391
2 United Kingdom 545
3 Australia 416

Aggregation

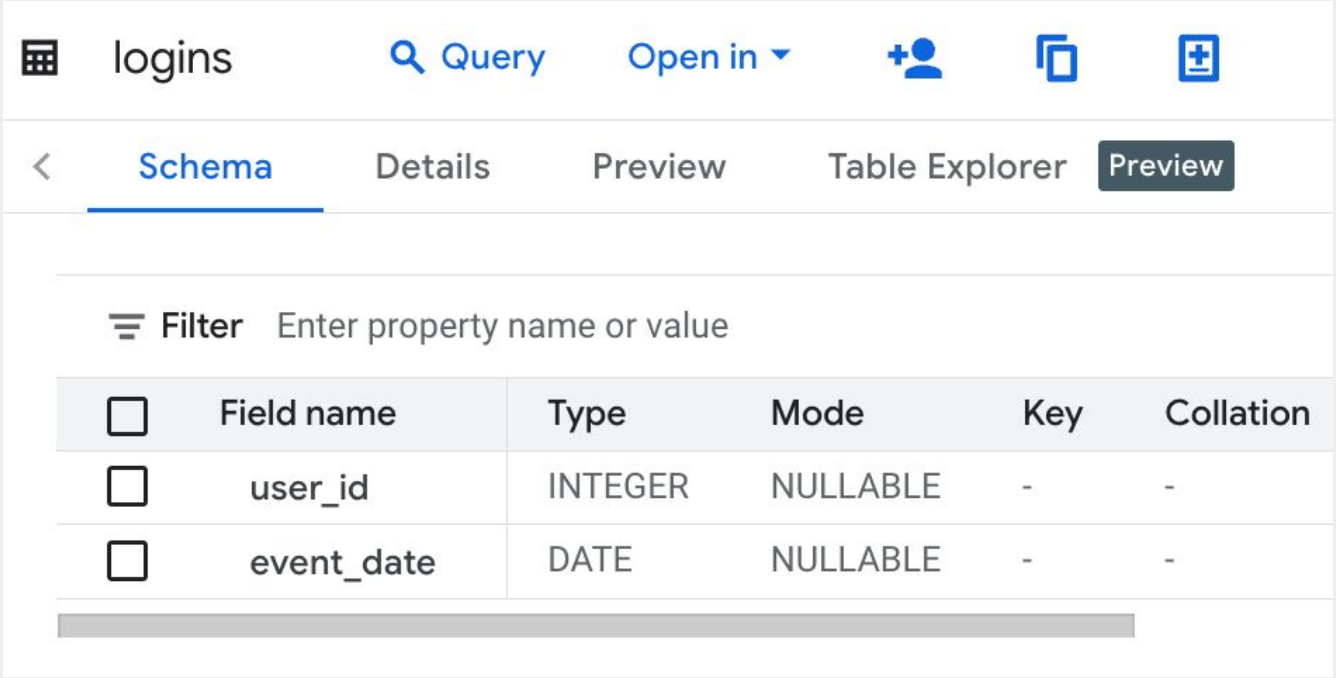
template

```
SELECT column,
            COUNT(1) AS cnt
FROM table
WHERE condition
GROUP BY column
ORDER BY cnt DESC
LIMIT rows
```

example

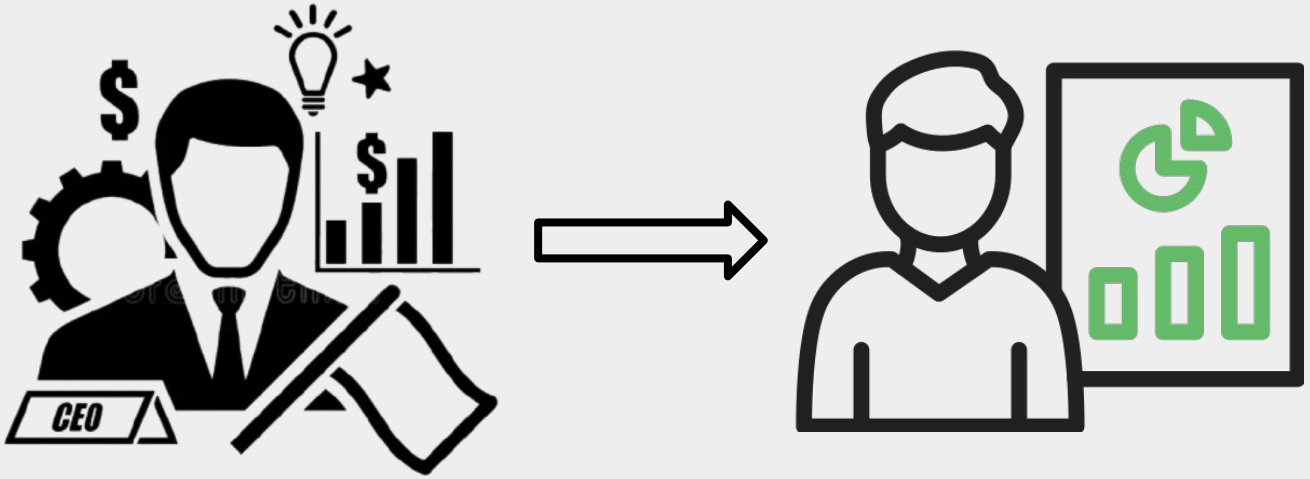
```
SELECT country_name,
            COUNT(user_id) AS users
FROM `test.users`
WHERE platform = ‘ios’
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5
```

Schema


logins
Query
Open in
Schema
Details
Preview
Table Explorer
Preview
Filter
Enter property name or value
Field name
Type
Mode
Key
Collation
user_id
INTEGER
NULLABLE
-
-
event_date
DATE
NULLABLE
-
-

Case

“What are the top 5 active days by number of users, from highest to lowest”



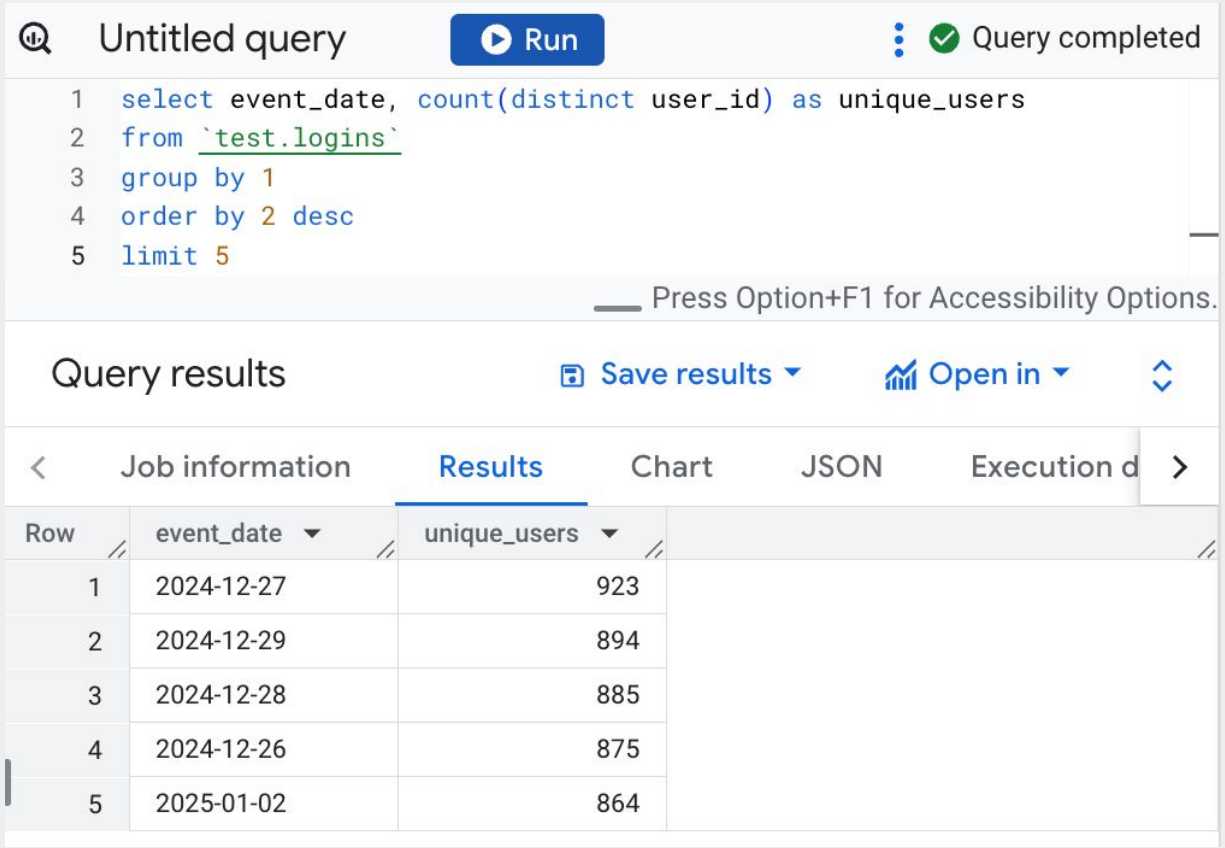

Untitled query
Run
Query completed
1 select event_date, count(distinct user_id) as unique_users
2 from `test.logins`
3 group by 1
4 order by 2 desc
5 limit 5
Press Option+F1 for Accessibility Options.
Query results
Save results
Open in
Job information
Results
Chart
JSON
Execution d
Row
event_date
unique_users
1 2024-12-27 923
2 2024-12-29 894
3 2024-12-28 885
4 2024-12-26 875
5 2025-01-02 864

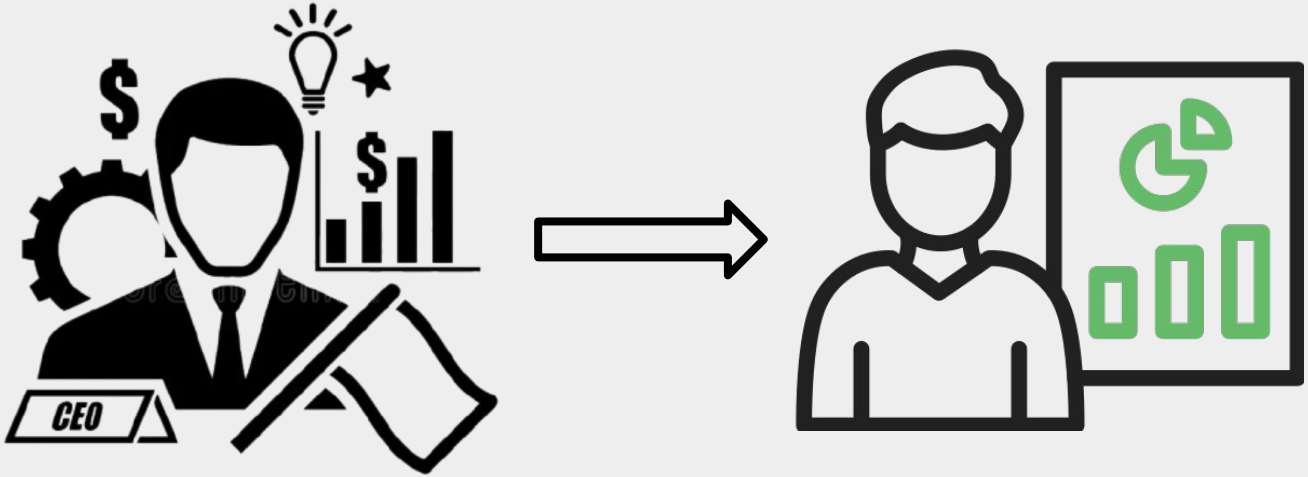
Table Joins

template

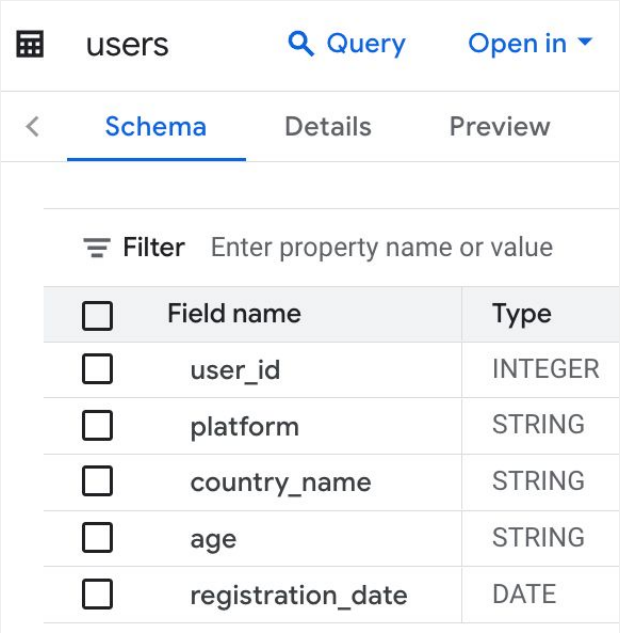
```
SELECT   a.column,
         b.column
FROM table_a AS a
JOIN table_b AS b ON a.key = b.key
WHERE condition;
```

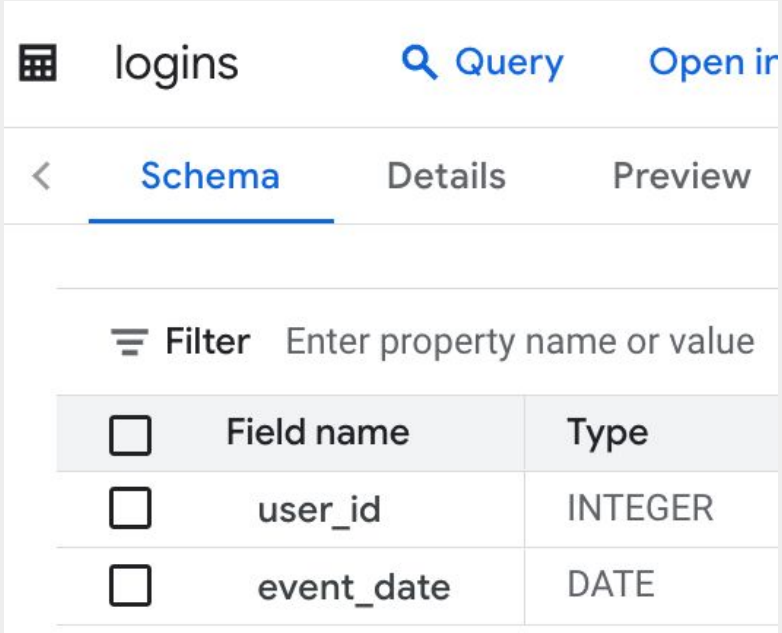
Case

“Calculate the average number of days US iOS user logs into the app by age number”



Schema


users
Query
Open in
Schema
Details
Preview
Filter
Enter property name or value
Field name
Type
user_id
INTEGER
platform
STRING
country_name
STRING
age
STRING
registration_date
DATE


logins
Query
Open ir
Schema
Details
Preview
Filter
Enter property name or value
Field name
Type
user_id
INTEGER
event_date
DATE

```sql
select *
from `test.logins`
where user_id = 10000
```

Query results

Job information | Results

Row	user_id	event_date
1	10000	2024-11-30
2	10000	2024-12-01
3	10000	2025-01-19
4	10000	2025-02-01
5	10000	2025-02-02

Untitled query | Run | Save

```sql
select *
from `test.users`
where user_id = 10000
```

Press Option+F1

Query results

Save results

Job information | Results | Chart | JSON | Execution details

Row	user_id	platform	country_name	age	registration_date
1	10000	ios	United States	25_34	2024-11-30

```sql
select *
from `test.logins` l
join `test.users` u on l.user_id = u.user_id
where l.user_id = 10000
```

Press Option+F1 for Accessibility Options.

Query results

Save results | Open in

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	user_id_1	platform	country_name	age	registration_date
1	10000	2024-11-30	10000	ios	United States	25_34	2024-11-30
2	10000	2024-12-01	10000	ios	United States	25_34	2024-11-30
3	10000	2025-01-19	10000	ios	United States	25_34	2024-11-30
4	10000	2025-02-01	10000	ios	United States	25_34	2024-11-30
5	10000	2025-02-02	10000	ios	United States	25_34	2024-11-30

Untitled query

Run

```
1  select *
2  from `test.logins`
3  where user_id = 10001
4  order by 2
```

Query results

Job information | Results | Char

Row	user_id	event_date
1	10001	2025-02-22
2	10001	2025-02-27
3	10001	2025-03-04
4	10001	2025-03-10

Untitled query

Run

Save

```
1  select *
2  from `test.users`
3  where user_id = 10001
```

Query results

Job information | Results | Chart | JSON

There is no data to display.

Untitled query

Run Save Download Share

```
1  select *
2  from `test.logins` l
3  join `test.users` u on l.user_id = u.user_id
4  where l.user_id in (10000, 10001)
5  order by l.user_id, event_date
```

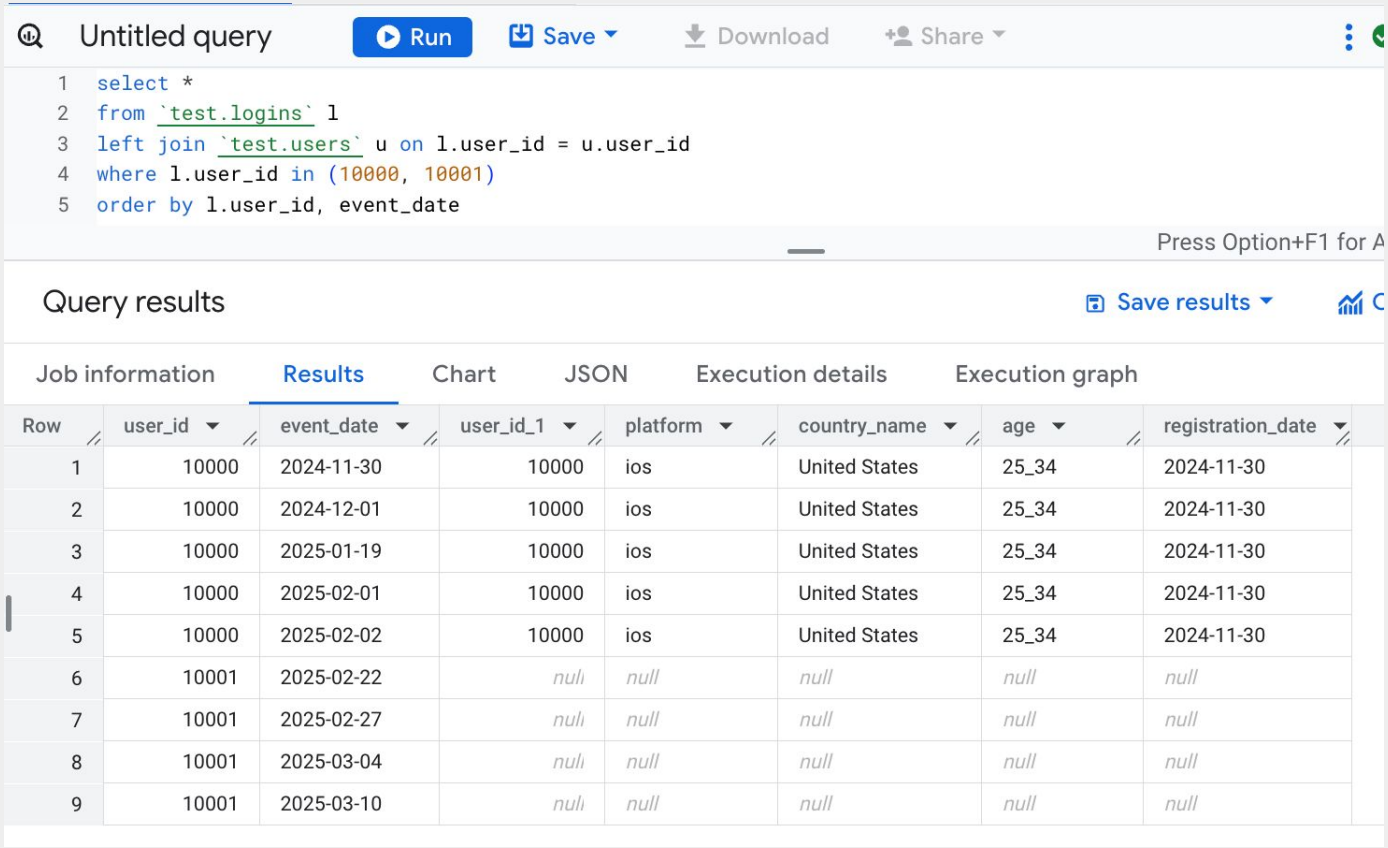
Press Option+F1 for

Query results

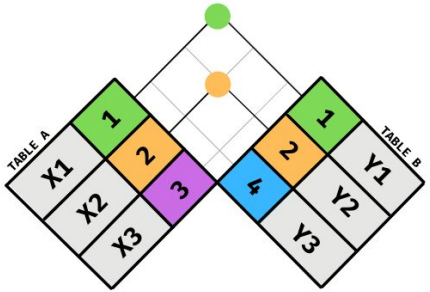
Save results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	user_id_1	platform	country_name	age	registration_date
1	10000	2024-11-30	10000	ios	United States	25_34	2024-11-30
2	10000	2024-12-01	10000	ios	United States	25_34	2024-11-30
3	10000	2025-01-19	10000	ios	United States	25_34	2024-11-30
4	10000	2025-02-01	10000	ios	United States	25_34	2024-11-30
5	10000	2025-02-02	10000	ios	United States	25_34	2024-11-30



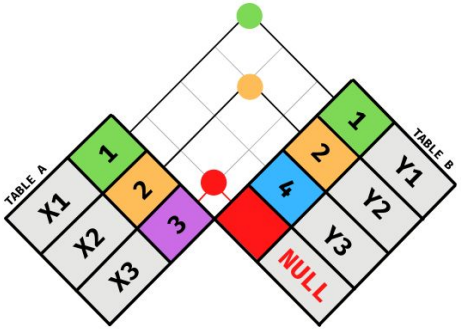
Row	user_id	event_date	user_id_1	platform	country_name	age	registration_date
1	10000	2024-11-30	10000	ios	United States	25_34	2024-11-30
2	10000	2024-12-01	10000	ios	United States	25_34	2024-11-30
3	10000	2025-01-19	10000	ios	United States	25_34	2024-11-30
4	10000	2025-02-01	10000	ios	United States	25_34	2024-11-30
5	10000	2025-02-02	10000	ios	United States	25_34	2024-11-30
6	10001	2025-02-22	null	null	null	null	null
7	10001	2025-02-27	null	null	null	null	null
8	10001	2025-03-04	null	null	null	null	null
9	10001	2025-03-10	null	null	null	null	null



INNER JOIN

```
SELECT
    <SELECT LIST>
FROM       TABLE_A A
INNER JOIN TABLE_B B
    ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2



LEFT JOIN

```
SELECT
    <SELECT LIST>
FROM      TABLE_A A
LEFT JOIN TABLE_B B
    ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
3	X3	NULL

Source

Table Joins

template

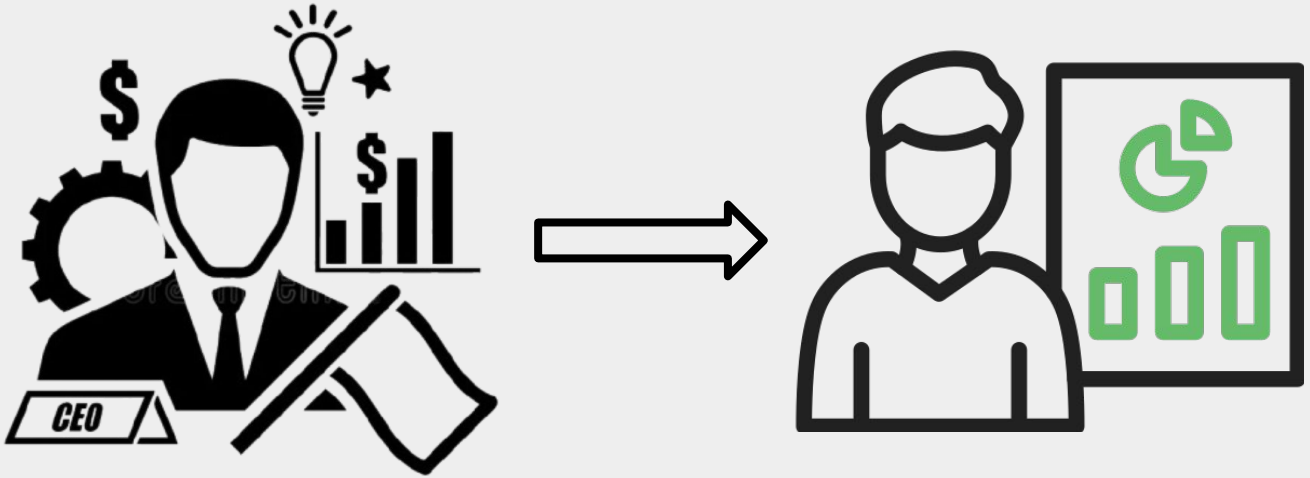
```
SELECT   a.column,
         b.column
FROM table_a AS a
JOIN table_b AS b ON a.key = b.key
WHERE condition;
```

example

```
SELECT   l.user_id,
         l.event_date,
         u.registration_date
FROM `test.logins` l
JOIN `test.users` u ON l.user_id = u.user_id
WHERE l.user_id IN (10000, 10001)
```

Case

“Calculate the average number of days US iOS user logs into the app by age number”



Untitled query

Run Save Download Share

```
1  select *
2  from `test.logins` l
3  join `test.users` u on l.user_id = u.user_id
4  where l.user_id in (10000, 10001)
5  order by l.user_id, event_date
```

Press Option+F1 for

Query results

Save results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	user_id_1	platform	country_name	age	registration_date
1	10000	2024-11-30	10000	ios	United States	25_34	2024-11-30
2	10000	2024-12-01	10000	ios	United States	25_34	2024-11-30
3	10000	2025-01-19	10000	ios	United States	25_34	2024-11-30
4	10000	2025-02-01	10000	ios	United States	25_34	2024-11-30
5	10000	2025-02-02	10000	ios	United States	25_34	2024-11-30

Untitled query

Run | Save | Download | Share

```sql
select  age,
        count(distinct l.user_id) as unique_users,
        count(distinct l.user_id||event_date) as active_days,
        round(count(distinct l.user_id||event_date) / count(distinct l.user_id), 2) as avg_days
from `test.logins` l
join `test.users` u on l.user_id = u.user_id
where platform = 'ios' and country_name = 'United States'
group by 1
order by 1

```

Press Option

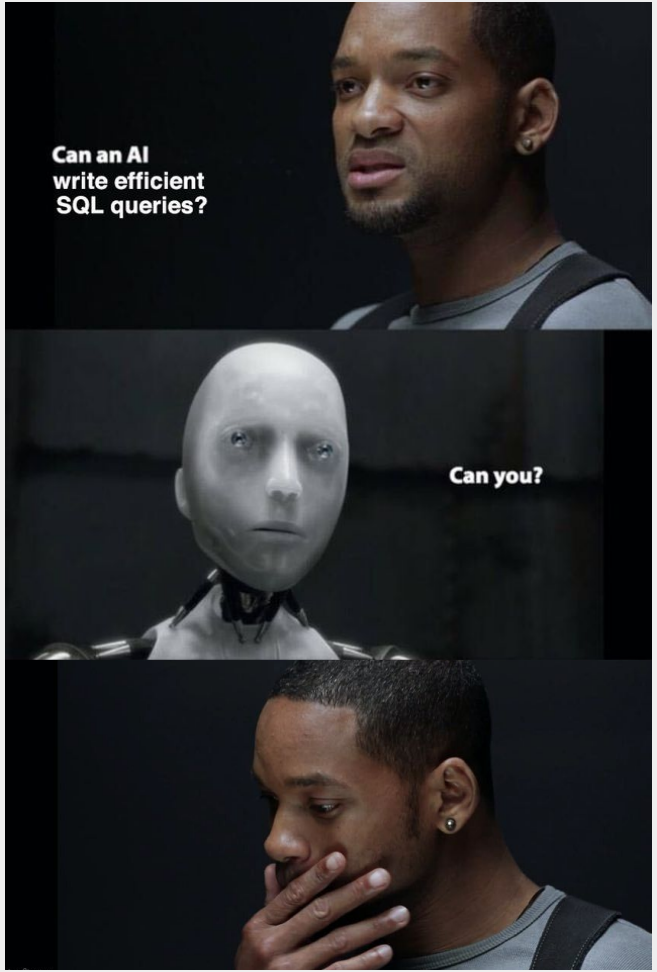
Query results

Save results

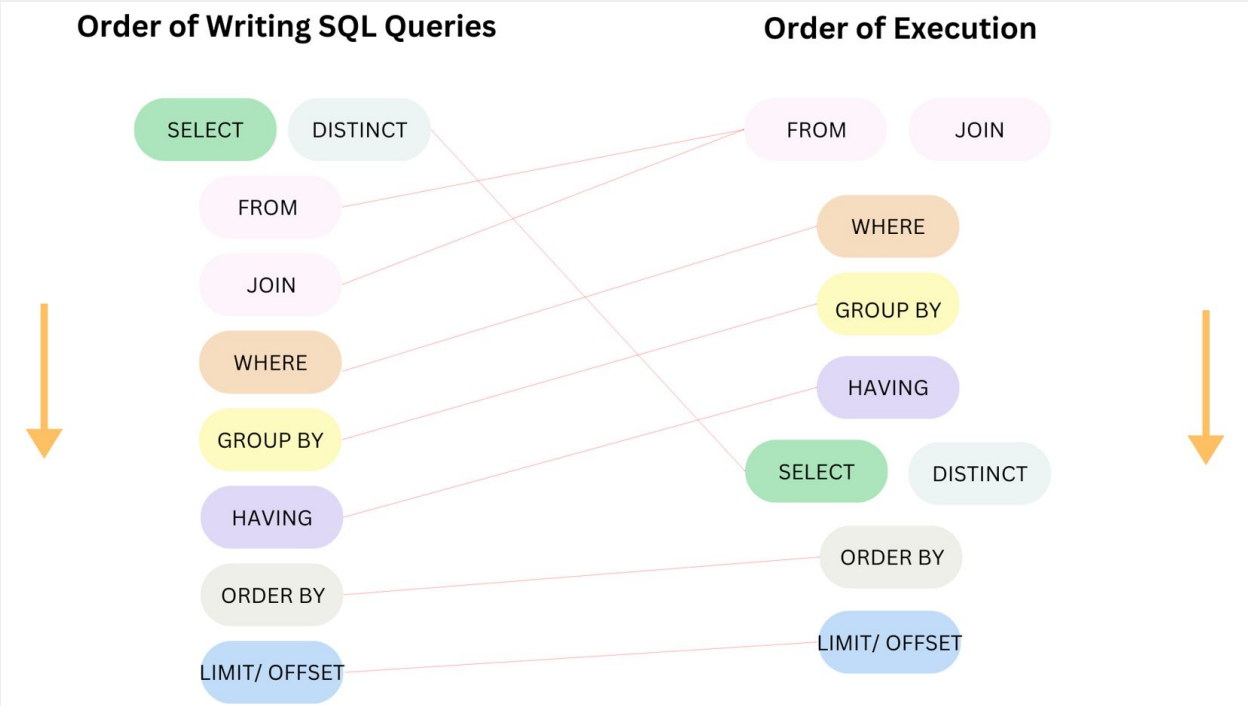
Job information | Results | Chart | JSON | Execution details | Execution graph

Row	age	unique_users	active_days	avg_days
1	18_24	613	4656	7.6
2	25_34	1325	10265	7.75
3	35_44	1698	16440	9.68
4	45_54	1035	12472	12.05
5	55_100	720	11042	15.34

Advanced SQL


Can an AI
write efficient
SQL queries?
Can you?

SQL Execution Order



Let’s practice!

Get a list of users with their name and email

Vote: Which one is better?

A

```sql
SELECT * FROM users;
```

B

```sql
SELECT id, name, email FROM users;
```

Let’s practice!

Get a list of users with their name and email

Vote: Which one is better?

A

```
SELECT * FROM users;
```

B

```
SELECT id, name, email FROM users;
```

Let’s practice!

Calculate total revenue from completed orders in the last 30 days.

Vote: Which one is better?

A

```sql
SELECT user_id, SUM(amount)
FROM orders
WHERE status = 'completed'
  AND created_at >= CURRENT_DATE - INTERVAL '30 DAY'
GROUP BY user_id;
```

B

```sql
SELECT user_id, SUM(amount)
FROM orders
GROUP BY user_id
HAVING MAX(status = 'completed') = TRUE
AND MAX(created_at >= CURRENT_DATE - INTERVAL '30 DAY') = TRUE;
```

Let’s practice!

Calculate total revenue from completed orders in the last 30 days.

Vote: Which one is better?

A

```sql
SELECT user_id, SUM(amount)
FROM orders
WHERE status = 'completed'
  AND created_at >= CURRENT_DATE - INTERVAL '30 DAY'
GROUP BY user_id;
```

B

```sql
SELECT user_id, SUM(amount)
FROM orders
GROUP BY user_id
HAVING MAX(status = 'completed') = TRUE
AND MAX(created_at >= CURRENT_DATE - INTERVAL '30 DAY') = TRUE;
```

Let’s practice!

Sum all orders from active users.

Vote: Which one is better?

A

```sql
SELECT SUM(o.amount)
FROM orders o
JOIN users u ON o.user_id = u.id
WHERE u.is_active = TRUE;
```

B

```sql
SELECT SUM(amount)
FROM orders
WHERE user_id IN (SELECT id FROM users
WHERE is_active = TRUE);
```

Let’s practice!

Sum all orders from active users.

Vote: Which one is better?

A

```sql
SELECT SUM(o.amount)
FROM orders o
JOIN users u ON o.user_id = u.id
WHERE u.is_active = TRUE;
```

B

```sql
SELECT SUM(amount)
FROM orders
WHERE user_id IN (SELECT id FROM users
WHERE is_active = TRUE);
```

Let’s practice!

Count unique users in the logs table.

Vote: Which one is better?

A

```sql
SELECT COUNT(DISTINCT user_id)
FROM logs;
```

B

```sql
SELECT user_id
FROM logs
GROUP BY 1;
```

Let’s practice!

Count unique users in the logs table.

Vote: Which one is better?

A

```sql
SELECT COUNT(DISTINCT user_id)
FROM logs;
```

B

```sql
SELECT user_id
FROM logs
GROUP BY 1;
```

Let’s practice!

Show all transactions for user.

Vote: Which one is better?

A

```sql
SELECT *
FROM payments
WHERE user_id = 'ABC'
ORDER BY created_at DESC
LIMIT 100;
```

B

```sql
SELECT *
FROM payments
WHERE user_id = 'ABC';
```

Let’s practice!

Show all transactions for user.

Vote: Which one is better?

A

```sql
SELECT *
FROM payments
WHERE user_id = 'ABC'
ORDER BY created_at DESC
LIMIT 100;
```

B

```sql
SELECT *
FROM payments
WHERE user_id = 'ABC';
```

Let’s practice!

Show total revenue per country for completed orders.

Vote: Which one is better?

A

```sql
SELECT c.country, SUM(o.amount)
FROM orders o
JOIN customers c ON o.customer_id = c.id
WHERE o.status = 'completed'
GROUP BY c.country;
```

B

```sql
WITH completed_orders AS (
  SELECT customer_id, amount
  FROM orders WHERE status = 'completed'
)

SELECT c.country, SUM(o.amount)
FROM completed_orders o
JOIN customers c ON o.customer_id = c.id
GROUP BY c.country;
```

Let’s practice!

Show total revenue per country for completed orders.

Vote: Which one is better?

A

```sql
SELECT c.country, SUM(o.amount)
FROM orders o
JOIN customers c ON o.customer_id = c.id
WHERE o.status = 'completed'
GROUP BY c.country;
```

B

```sql
WITH completed_orders AS (
  SELECT customer_id, amount
  FROM orders WHERE status = 'completed'
)

SELECT c.country, SUM(o.amount)
FROM completed_orders o
JOIN customers c ON o.customer_id = c.id
GROUP BY c.country;
```

Optimisation Hints

Avoid SELECT *

Select only needed columns

Count Clean

Use DISTINCT; it’s fine

Filter Early

Reduce data before grouping

Sort Wisely

Don’t sort without a reason

Join Efficiently

Prefer JOIN over subqueries

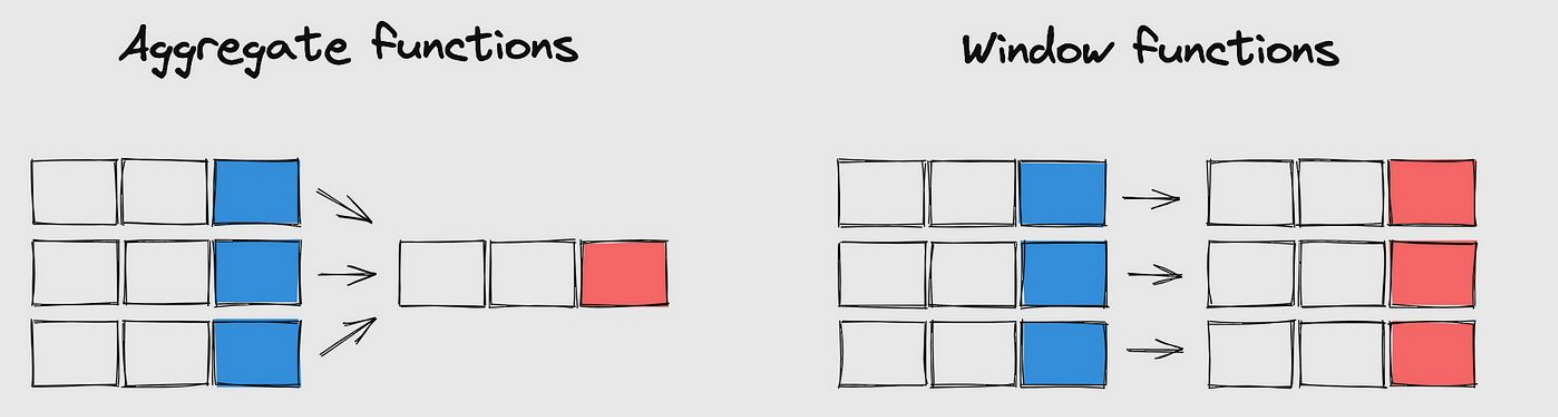
Simplify Logic

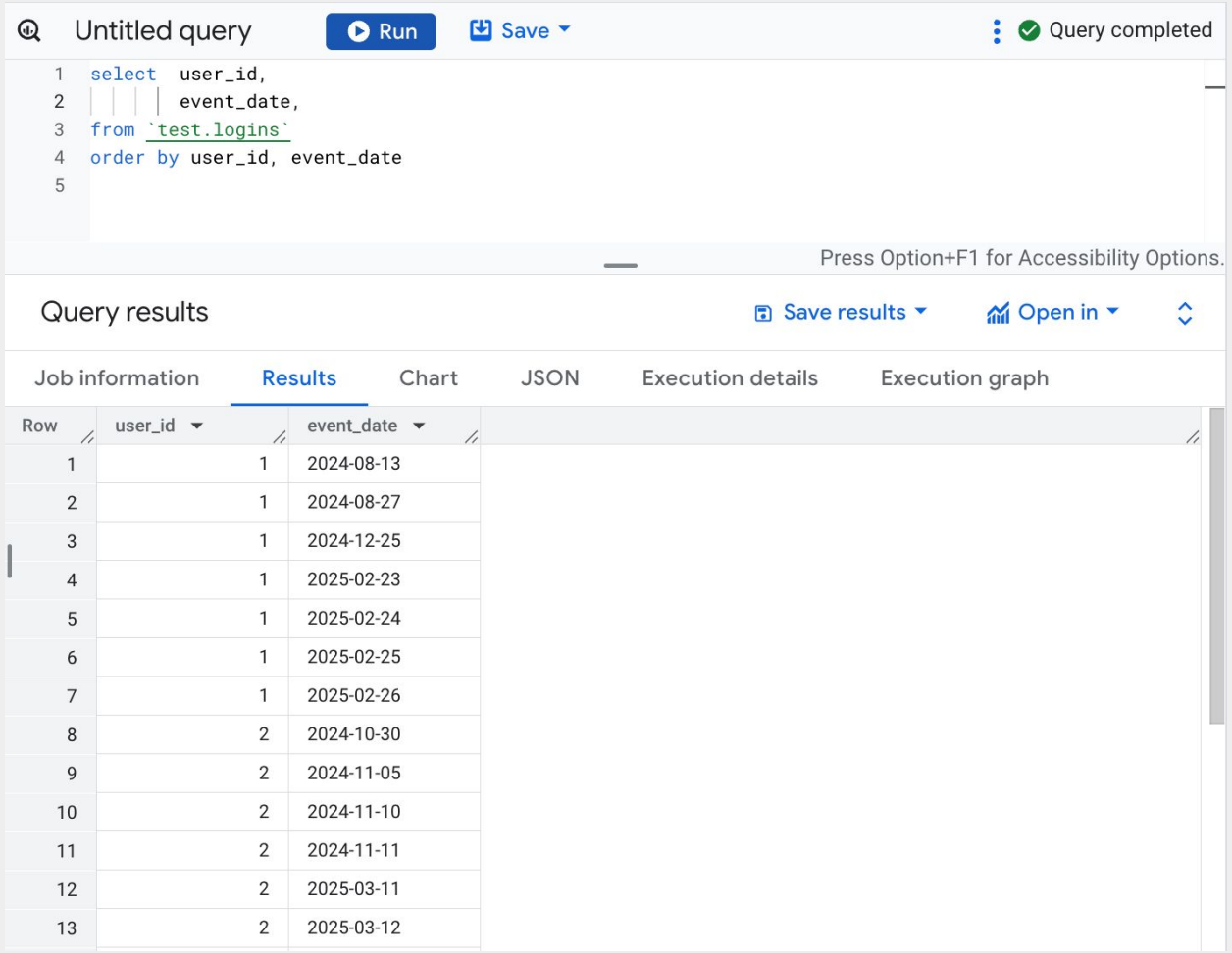
Use CTEs for clarity

Query Optimisation

Window functions

Window functions calculate values using data from other rows, without changing the number of rows in the result.




Untitled query
Run
Save
Query completed
1 select user_id,
2 event_date,
3 from `test.logins`
4 order by user_id, event_date
5
Press Option+F1 for Accessibility Options.
Query results
Save results
Open in
Job information
Results
Chart
JSON
Execution details
Execution graph
Row
user_id
event_date
1 1 2024-08-13
2 1 2024-08-27
3 1 2024-12-25
4 1 2025-02-23
5 1 2025-02-24
6 1 2025-02-25
7 1 2025-02-26
8 2 2024-10-30
9 2 2024-11-05
10 2 2024-11-10
11 2 2024-11-11
12 2 2025-03-11
13 2 2025-03-12

Untitled query Run Save Query completed

```
1  select  user_id,
2          event_date,
3          count(distinct event_date) over (partition by user_id) as total_days,
4  from `test.logins`
5  order by user_id, event_date
6
```

Press Option+F1 for Accessibility Options.

Query results

Save results Open in

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	total_days
1	1	2024-08-13	7
2	1	2024-08-27	7
3	1	2024-12-25	7
4	1	2025-02-23	7
5	1	2025-02-24	7
6	1	2025-02-25	7
7	1	2025-02-26	7
8	2	2024-10-30	6
9	2	2024-11-05	6
10	2	2024-11-10	6
11	2	2024-11-11	6
12	2	2025-03-11	6
13	2	2025-03-12	6

Untitled query

Run | Save | This query will process 1.67 MB when run.

```sql
select  user_id,
        event_date,
        count(distinct event_date) over (partition by user_id) as total_days,
        count(event_date) over (partition by user_id order by event_date) as current_total_days,
from `test.logins`
order by user_id, event_date

```

Press Option+F1 for Accessibility Options.

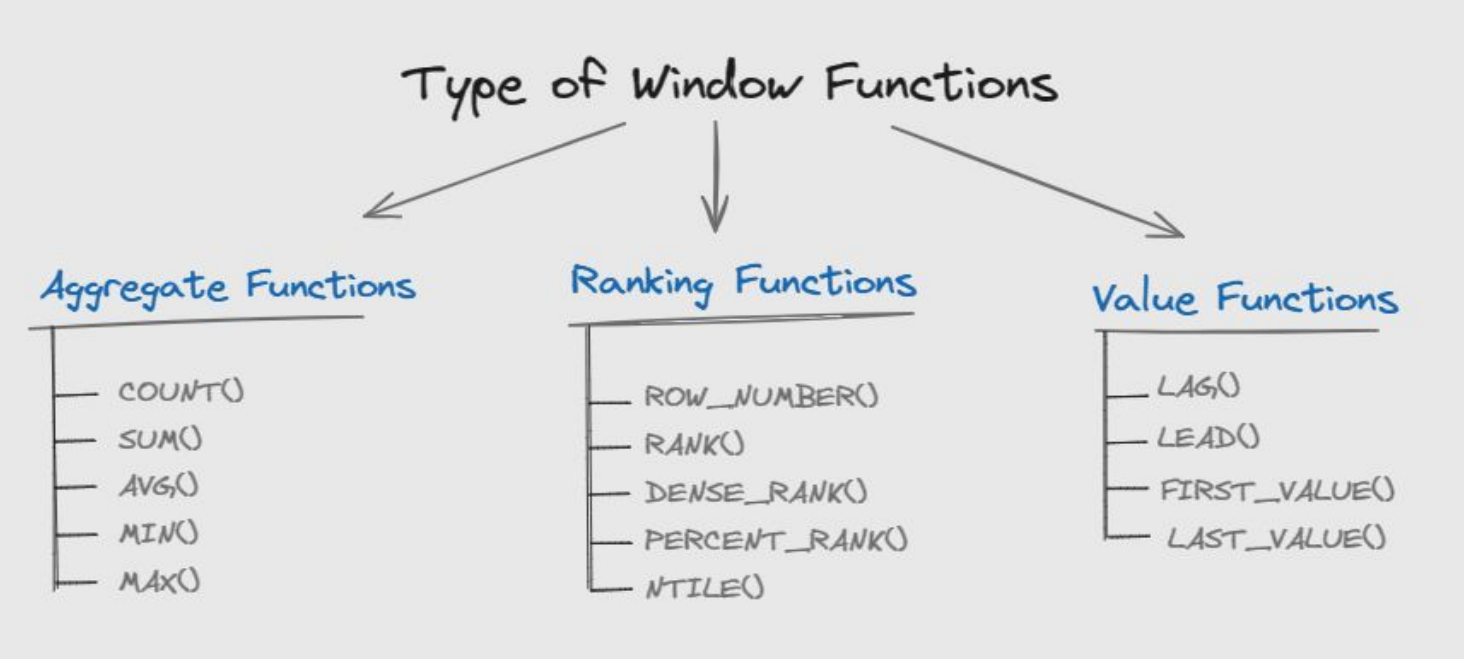
Query results

Save results | Open in

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	total_days	current_total_days
1	1	2024-08-13	7	1
2	1	2024-08-27	7	2
3	1	2024-12-25	7	3
4	1	2025-02-23	7	4
5	1	2025-02-24	7	5
6	1	2025-02-25	7	6
7	1	2025-02-26	7	7
8	2	2024-10-30	6	1
9	2	2024-11-05	6	2
10	2	2024-11-10	6	3
11	2	2024-11-11	6	4
12	2	2025-03-11	6	5
13	2	2025-03-12	6	6

Window functions



Untitled query

Run Save Query completed

```
1  select  user_id,
2          event_date,
3          count(distinct event_date) over (partition by user_id) as total_days,
4          count(event_date) over (partition by user_id order by event_date) as current_total_days,
5          row_number() over (partition by user_id order by event_date) as current_total_days_rn,
6  from `test.logins`
7  order by user_id, event_date
```

Press Option+F1 for Accessibility Options.

Query results

Save results Open in

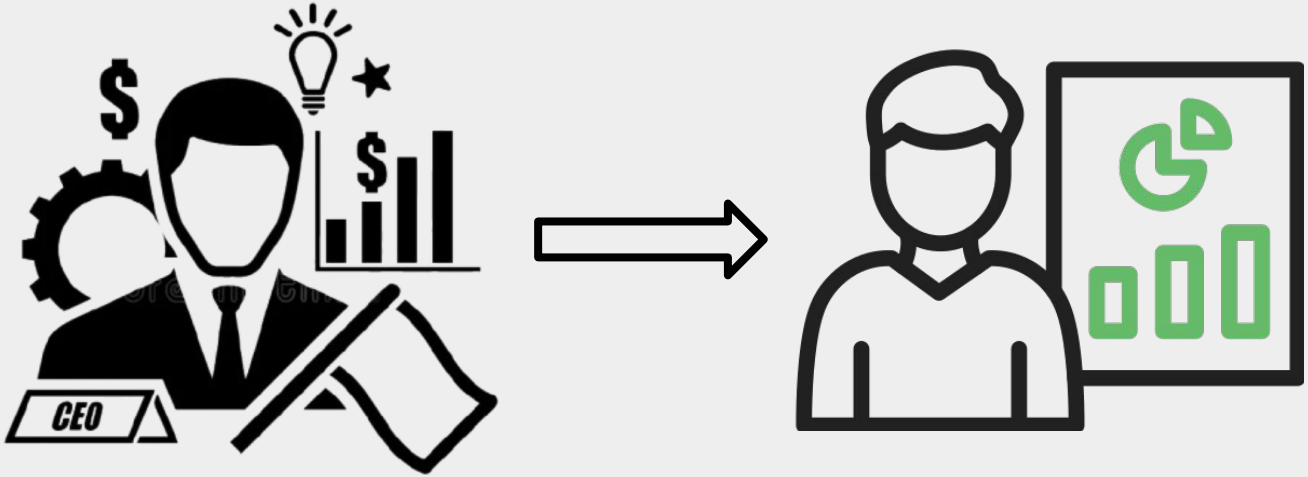
Job information | Results | Chart | JSON | Execution details | Execution graph

Row	user_id	event_date	total_days	current_total_days	current_total_days_rn
1	1	2024-08-13	7	1	1
2	1	2024-08-27	7	2	2
3	1	2024-12-25	7	3	3
4	1	2025-02-23	7	4	4
5	1	2025-02-24	7	5	5
6	1	2025-02-25	7	6	6
7	1	2025-02-26	7	7	7
8	2	2024-10-30	6	1	1
9	2	2024-11-05	6	2	2
10	2	2024-11-10	6	3	3
11	2	2024-11-11	6	4	4
12	2	2025-03-11	6	5	5
13	2	2025-03-12	6	6	6

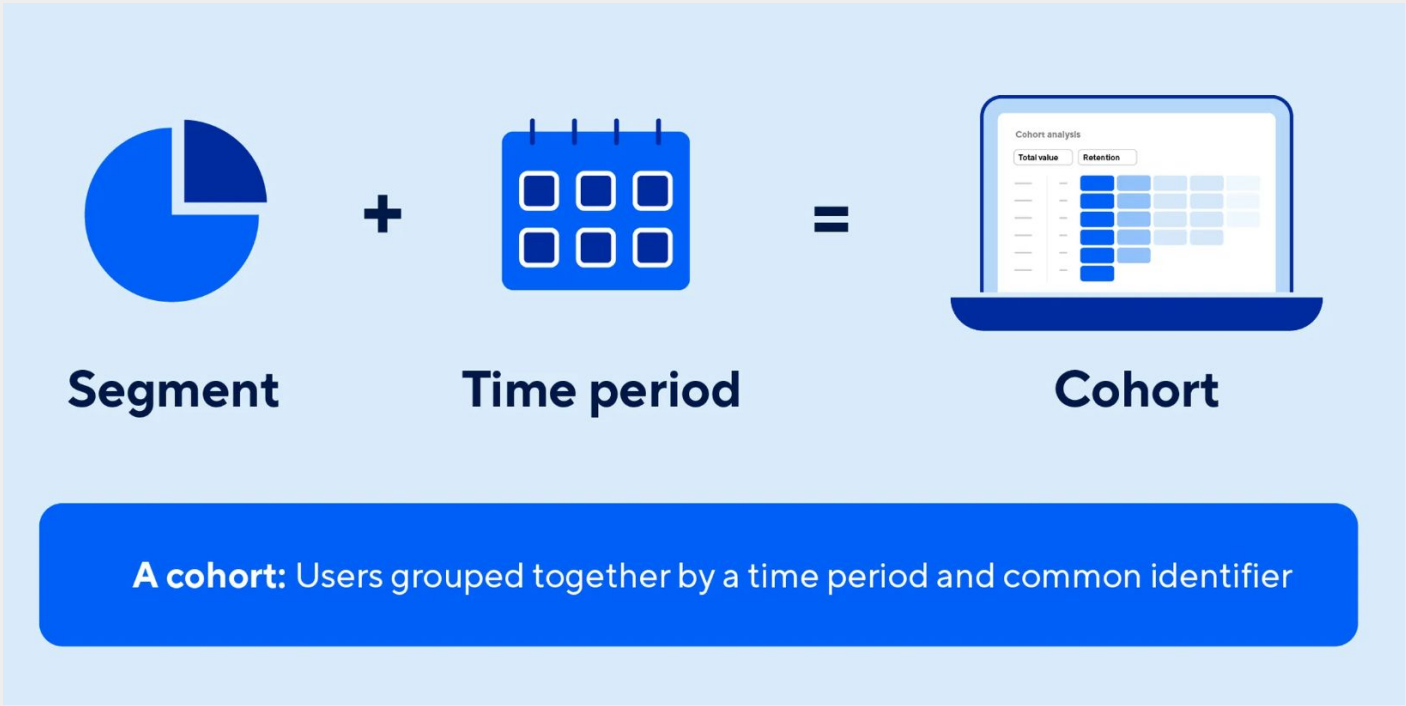
SQL in practice

Case

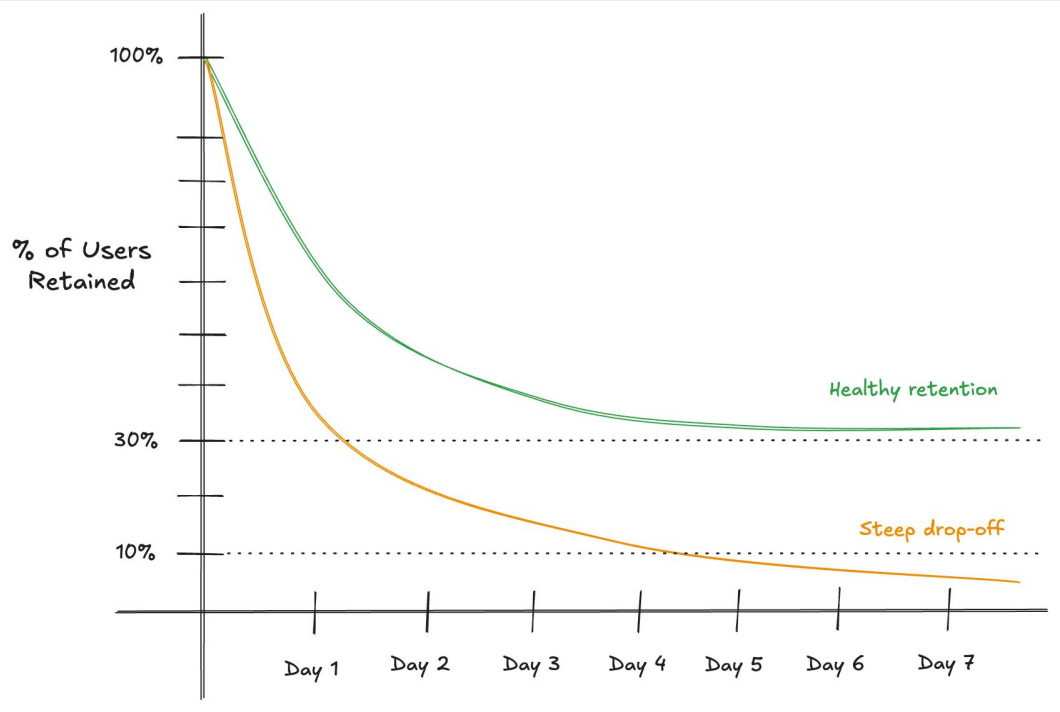
“The release was on January 10, 2025. We need to check how it affected retention on each platform”



Cohort

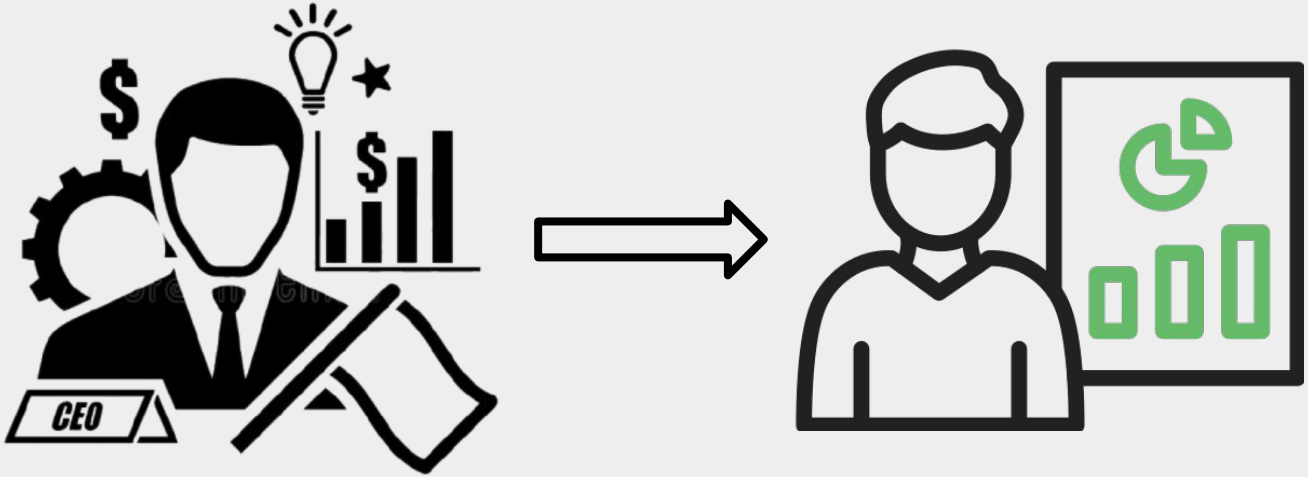

Cohort analysis
Total value
Retention
Segment
Time period
Cohort
A cohort: Users grouped together by a time period and common identifier

Retention


100%
% of Users Retained
30%
10%
Healthy retention
Steep drop-off
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7

Case

“The release was on January 10, 2025. We need to check how it affected retention on each platform”



```sql
with

users_data as (
        select user_id, registration_date, platform
        from `test.users`
        where registration_date between '2025-01-01' and '2025-01-31'
),

```

```sql
with

users_data as (
        select user_id, registration_date, platform
        from `test.users`
        where registration_date between '2025-01-01' and '2025-01-31'
),

logins as (
        select  l.*, u.* except (user_id)
        from `test.logins` l
        join users_data u using (user_id)
        where event_date between '2025-01-01' and '2025-02-01'
),
```

Row	user_id	event_date	registration_date	platform
1	5	2025-01-12	2025-01-12	ios
2	5	2025-01-14	2025-01-12	ios
3	8	2025-01-22	2025-01-22	ios
4	8	2025-01-23	2025-01-22	ios
5	8	2025-01-24	2025-01-22	ios
6	8	2025-01-25	2025-01-22	ios
7	8	2025-01-27	2025-01-22	ios
8	8	2025-01-28	2025-01-22	ios
9	8	2025-02-01	2025-01-22	ios
10	11	2025-01-08	2025-01-08	ios
11	11	2025-01-15	2025-01-08	ios
12	21	2025-01-20	2025-01-20	ios
13	21	2025-01-21	2025-01-20	ios
14	21	2025-01-23	2025-01-20	ios
15	21	2025-01-24	2025-01-20	ios
16	21	2025-01-26	2025-01-20	ios
17	21	2025-01-27	2025-01-20	ios

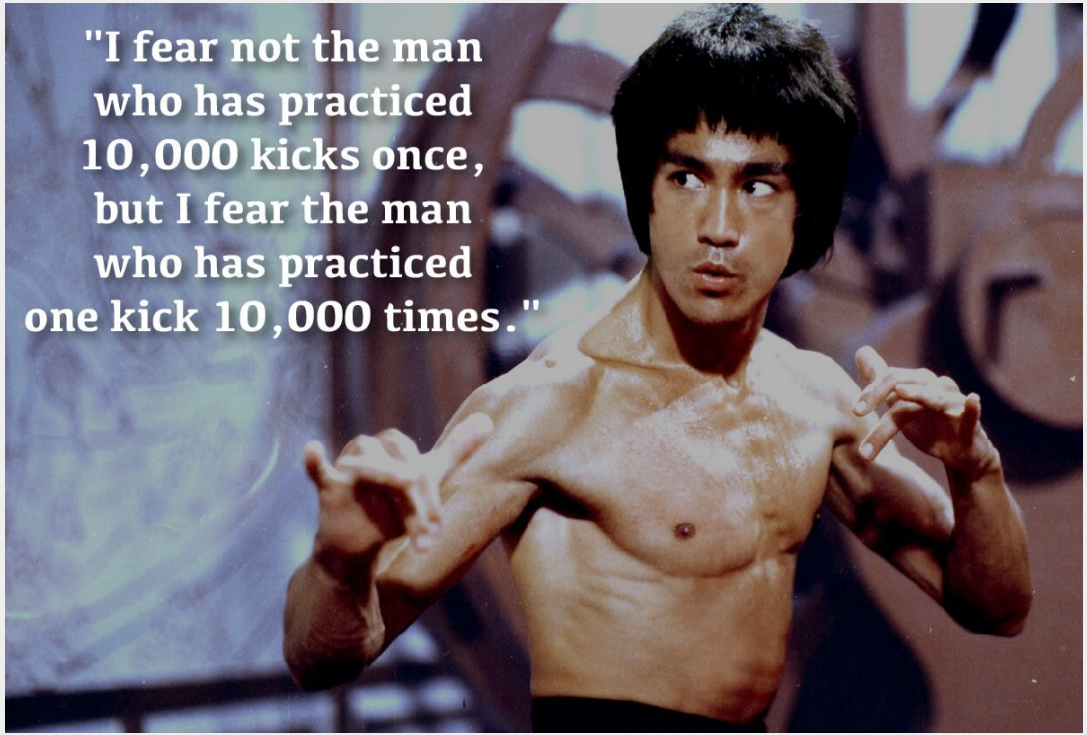
```sql
with

users_data as (
        select user_id, registration_date, platform
        from `test.users`
        where registration_date between '2025-01-01' and '2025-01-31'
),

logins as (
        select  l.*, u.* except (user_id)
        from `test.logins` l
        join users_data u using (user_id)
        where event_date between '2025-01-01' and '2025-02-01'
),

retention_over_day as (
        select  registration_date,
                count(distinct case when platform = 'ios' and date_diff(event_date, registration_date, day) = 1 then user_id else null end)
                / count(distinct case when platform = 'ios' then user_id else null end) retention_lt1_ios,
                count(distinct case when platform = 'android' and date_diff(event_date, registration_date, day) = 1 then user_id else null end)
                / count(distinct case when platform = 'android' then user_id else null end) retention_lt1_android
        from logins
        group by 1
)
```

Row	registration_date	retention_lt1_ios	retention_lt1_android
1	2025-01-01	0.35	0.5
2	2025-01-02	0.521739130434…	0.458333333333333…
3	2025-01-03	0.350877192982…	0.322580645161290…
4	2025-01-04	0.307692307692…	0.363636363636363…
5	2025-01-05	0.395833333333…	0.333333333333333…
6	2025-01-06	0.309523809523…	0.555555555555555…
7	2025-01-07	0.361702127659…	0.538461538461538…
8	2025-01-08	0.5	0.5
9	2025-01-09	0.276595744680…	0.277777777777777…
10	2025-01-10	0.441860465116…	0.8
11	2025-01-11	0.303030303030…	0.333333333333333…
12	2025-01-12	0.333333333333…	0.5
13	2025-01-13	0.230769230769…	0.769230769230769…
14	2025-01-14	0.416666666666…	0.538461538461538…
15	2025-01-15	0.625	0.666666666666666…
16	2025-01-16	0.289473684210…	0.555555555555555…
17	2025-01-17	0.333333333333…	0.090909090909090…
18	2025-01-18	0.391304347826…	0.529411764705882…
19	2025-01-19	0.488888888888…	0.55
20	2025-01-20	0.367346938775…	0.538461538461538…

```sql
with

users_data as (
        select user_id, registration_date, platform
        from `test.users`
        where registration_date between '2025-01-01' and '2025-01-31'
),

logins as (
        select  l.*, u.* except (user_id)
        from `test.logins` l
        join users_data u using (user_id)
        where event_date between '2025-01-01' and '2025-02-01'
),

retention_over_day as (
        select  registration_date,
                count(distinct case when platform = 'ios' and date_diff(event_date, registration_date, day) = 1 then user_id else null end)
                / count(distinct case when platform = 'ios' then user_id else null end) retention_lt1_ios,
                count(distinct case when platform = 'android' and date_diff(event_date, registration_date, day) = 1 then user_id else null end)
                / count(distinct case when platform = 'android' then user_id else null end) retention_lt1_android
        from logins
        group by 1
)

select  registration_date,
        round(avg(retention_lt1_ios) over (order by registration_date rows between 2 preceding and current row), 2) AS ret_lt1_ios_ma_3d,
        round(avg(retention_lt1_android) over (order by registration_date rows between 2 preceding and current row), 2) AS ret_lt1_android_ma_3d,
from retention_over_day
order by registration_date
```

Row	registration_date	ret_lt1_ios_ma_2d	ret_lt1_android_ma_2d
1	2025-01-01	0.35	0.5
2	2025-01-02	0.44	0.48
3	2025-01-03	0.44	0.39
4	2025-01-04	0.33	0.34
5	2025-01-05	0.35	0.35
6	2025-01-06	0.35	0.44
7	2025-01-07	0.34	0.55
8	2025-01-08	0.43	0.52
9	2025-01-09	0.39	0.39
10	2025-01-10	0.36	0.54
11	2025-01-11	0.37	0.57
12	2025-01-12	0.32	0.42
13	2025-01-13	0.28	0.63
14	2025-01-14	0.32	0.65
15	2025-01-15	0.52	0.6
16	2025-01-16	0.46	0.61
17	2025-01-17	0.31	0.32
18	2025-01-18	0.36	0.31
19	2025-01-19	0.44	0.54
20	2025-01-20	0.43	0.54
21	2025-01-21	0.41	0.6

More Practice



More Practice

Mode SQL Tutorial – interactive beginner-friendly course

Window Functions – hands-on guide with exercises

LeetCode SQL Tasks – challenges to level up your SQL

BigQuery Public Datasets – open datasets for practice

Thanks for your attention

Contacts

Feedback


Q&A