Web performance

This role focuses on the website's technical performance and user experience. They monitor traffic, sessions, conversion funnels, and A/B test results to optimize the site.

Category 1: Traffic & User Activity:

Goal

To understand **where users come from**, **how they interact**, and **when the traffic peaks** i.e., core **web performance metrics**.

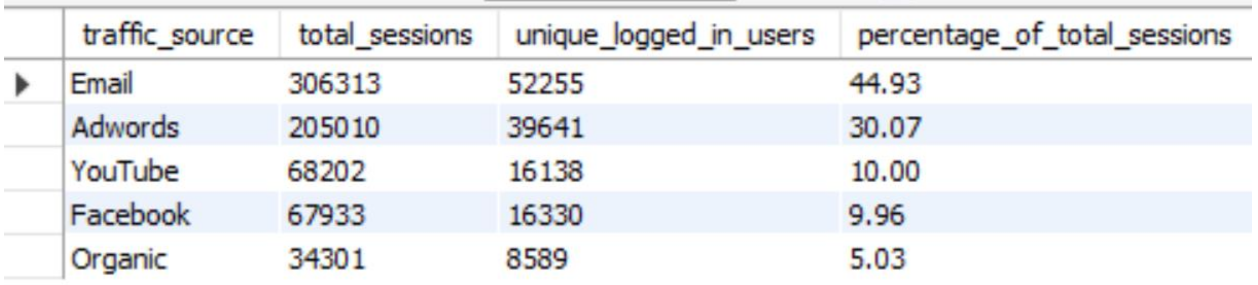
View 1: traffic_source_summary

Description

This view aggregates the number of events per traffic source and calculates the percentage contribution of each source to total traffic.
It helps you identify which channels (e.g., Google, Facebook, Direct) are driving the most visits.

```
USE looker_ecommerce;

CREATE VIEW traffic_source_summary AS
SELECT
    traffic_source,

    COUNT(DISTINCT session_id) AS total_sessions,

    COUNT(DISTINCT
        CASE
            WHEN user_id IS NOT NULL AND user_id <> 0 THEN user_id
        END
    ) AS unique_logged_in_users,

    ROUND(
        COUNT(DISTINCT session_id) * 100.0 /
        (SELECT COUNT(DISTINCT session_id) FROM events),
        2
    ) AS percentage_of_total_sessions
```

```
FROM events
GROUP BY traffic_source
ORDER BY total_sessions DESC;
```



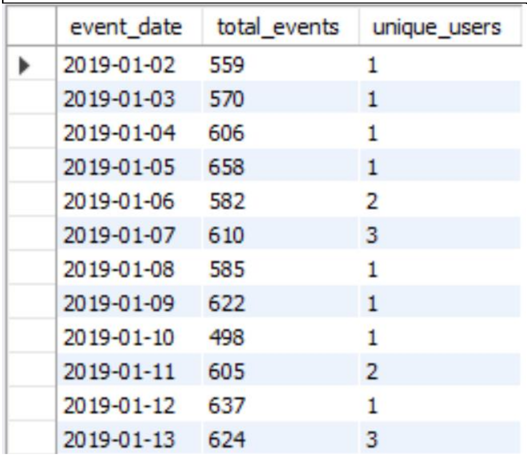
	traffic_source	total_sessions	unique_logged_in_users	percentage_of_total_sessions
▶	Email	306313	52255	44.93
	Adwords	205010	39641	30.07
	YouTube	68202	16138	10.00
	Facebook	67933	16330	9.96
	Organic	34301	8589	5.03

View 2: daily_traffic_overview

Description

Shows total events and unique users **per day**.
This helps visualize **traffic trends** over time for example, which days have the highest engagement.

```
CREATE VIEW daily_traffic_overview AS
SELECT
    DATE(created_at) AS event_date,
    COUNT(*) AS total_events,
    COUNT(DISTINCT user_id) AS unique_users
FROM events
GROUP BY DATE(created_at)
ORDER BY event_date;
```



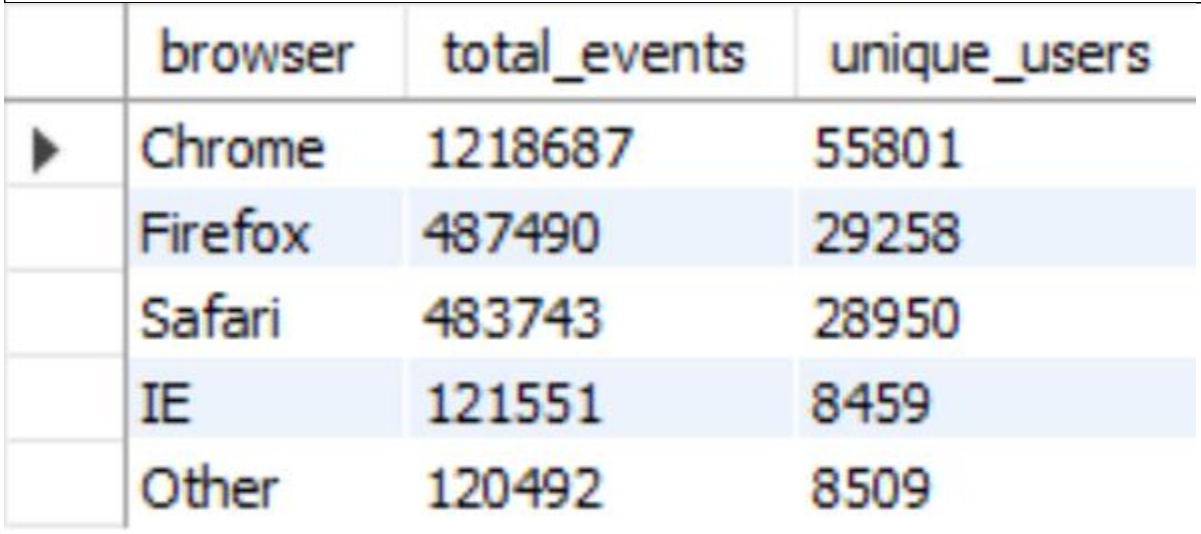
	event_date	total_events	unique_users
▶	2019-01-02	559	1
	2019-01-03	570	1
	2019-01-04	606	1
	2019-01-05	658	1
	2019-01-06	582	2
	2019-01-07	610	3
	2019-01-08	585	1
	2019-01-09	622	1
	2019-01-10	498	1
	2019-01-11	605	2
	2019-01-12	637	1
	2019-01-13	624	3

View 3: browser_usage_stats

Description

Displays the number of users and events per browser.
It helps understand which browsers dominate your user base useful for **frontend optimization**.

```
CREATE VIEW browser_usage_stats AS
SELECT
    browser,
    COUNT(*) AS total_events,
    COUNT(DISTINCT user_id) AS unique_users
FROM events
GROUP BY browser
ORDER BY total_events DESC;
```



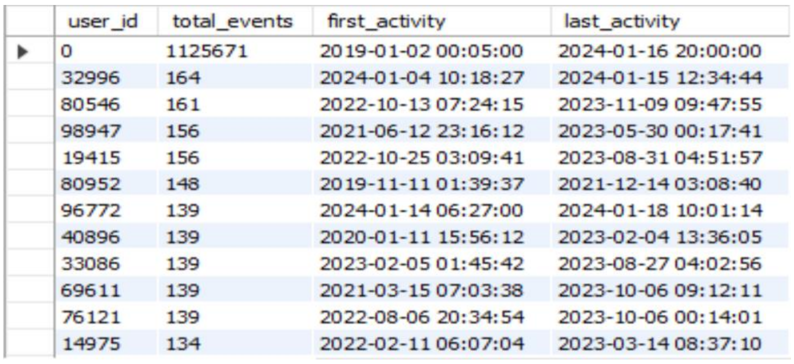
browser	total_events	unique_users
Chrome	1218687	55801
Firefox	487490	29258
Safari	483743	28950
IE	121551	8459
Other	120492	8509

View 4: user_event_frequency

Description

Shows the number of events each user performed.
Useful to measure **engagement per user** (active vs passive visitors).

```
CREATE VIEW user_event_frequency AS
SELECT
    user_id,
    COUNT(*) AS total_events,
    MIN(created_at) AS first_activity,
    MAX(created_at) AS last_activity
FROM events
GROUP BY user_id
ORDER BY total_events DESC;
```



user_id	total_events	first_activity	last_activity
0	1125671	2019-01-02 00:05:00	2024-01-16 20:00:00
32996	164	2024-01-04 10:18:27	2024-01-15 12:34:44
80546	161	2022-10-13 07:24:15	2023-11-09 09:47:55
98947	156	2021-06-12 23:16:12	2023-05-30 00:17:41
19415	156	2022-10-25 03:09:41	2023-08-31 04:51:57
80952	148	2019-11-11 01:39:37	2021-12-14 03:08:40
96772	139	2024-01-14 06:27:00	2024-01-18 10:01:14
40896	139	2020-01-11 15:56:12	2023-02-04 13:36:05
33086	139	2023-02-05 01:45:42	2023-08-27 04:02:56
69611	139	2021-03-15 07:03:38	2023-10-06 09:12:11
76121	139	2022-08-06 20:34:54	2023-10-06 00:14:01
14975	134	2022-02-11 06:07:04	2023-03-14 08:37:10

Category 2: Engagement & Conversion Metrics

Goal

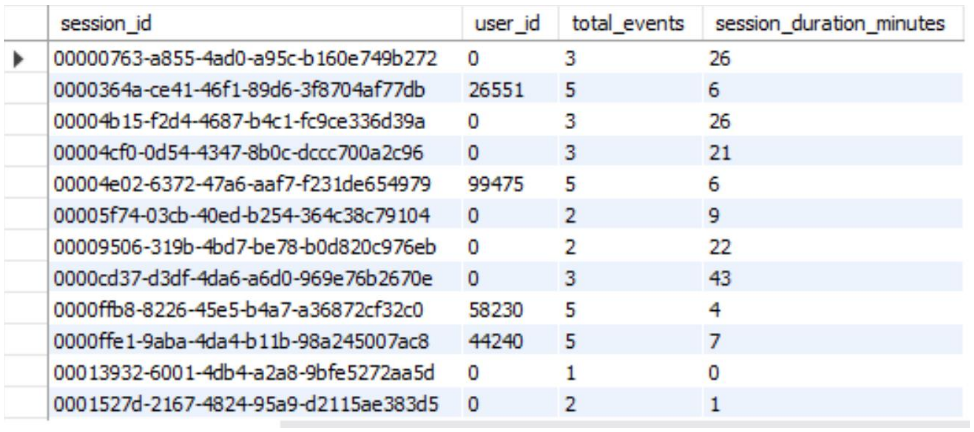
To measure **how deeply users interact** with your website and how often their visits lead to conversions (e.g., purchases, sign-ups, etc.).
These views help you analyze **user engagement**, **bounce rate**, and **conversion funnels**.

View 5: session_engagement

Description

This view summarizes user sessions, counting the number of events per session and calculating the average duration.
It's useful for identifying how long users stay active and how engaged they are within a single visit.

```
CREATE VIEW session_engagement AS
SELECT
    session_id,
    user_id,
    COUNT(*) AS total_events,
    TIMESTAMPDIFF(MINUTE, MIN(created_at), MAX(created_at)) AS
session_duration_minutes
FROM events
GROUP BY session_id, user_id;
```



session_id	user_id	total_events	session_duration_minutes
00000763-a855-4ad0-a95c-b160e749b272	0	3	26
0000364a-ce41-46f1-89d6-3f8704af77db	26551	5	6
00004b15-f2d4-4687-b4c1-fc9ce336d39a	0	3	26
00004cf0-0d54-4347-8b0c-dccc700a2c96	0	3	21
00004e02-6372-47a6-aaf7-f231de654979	99475	5	6
00005f74-03cb-40ed-b254-364c38c79104	0	2	9
00009506-319b-4bd7-be78-b0d820c976eb	0	2	22
0000cd37-d3df-4da6-a6d0-969e76b2670e	0	3	43
0000ffb8-8226-45e5-b4a7-a36872cf32c0	58230	5	4
0000ffe1-9aba-4da4-b11b-98a245007ac8	44240	5	7
00013932-6001-4db4-a2a8-9bfe5272aa5d	0	1	0
0001527d-2167-4824-95a9-d2115ae383d5	0	2	1

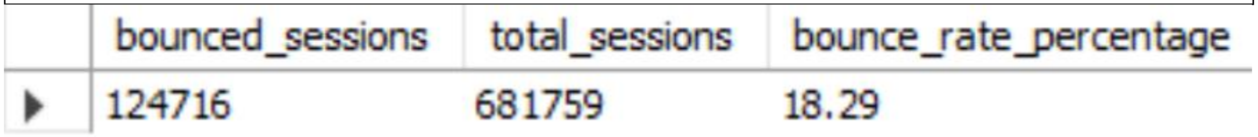
View 6:

bounce_rate_analysis

Description

Identifies users who **left after only one event** (bounce sessions).
Helps in diagnosing landing page or UX issues.

```
CREATE VIEW bounce_rate_analysis AS
SELECT
    COUNT(DISTINCT CASE WHEN event_count = 1 THEN session_id END) AS
bounced_sessions,
    COUNT(DISTINCT session_id) AS total_sessions,
    ROUND(
        COUNT(DISTINCT CASE WHEN event_count = 1 THEN session_id END)
        / COUNT(DISTINCT session_id) * 100, 2
    ) AS bounce_rate_percentage
FROM (
    SELECT session_id, COUNT(*) AS event_count
    FROM events
    GROUP BY session_id
) AS session_counts;
```



bounced_sessions	total_sessions	bounce_rate_percentage
124716	681759	18.29

Category 3: Page & Event Performance

Goal

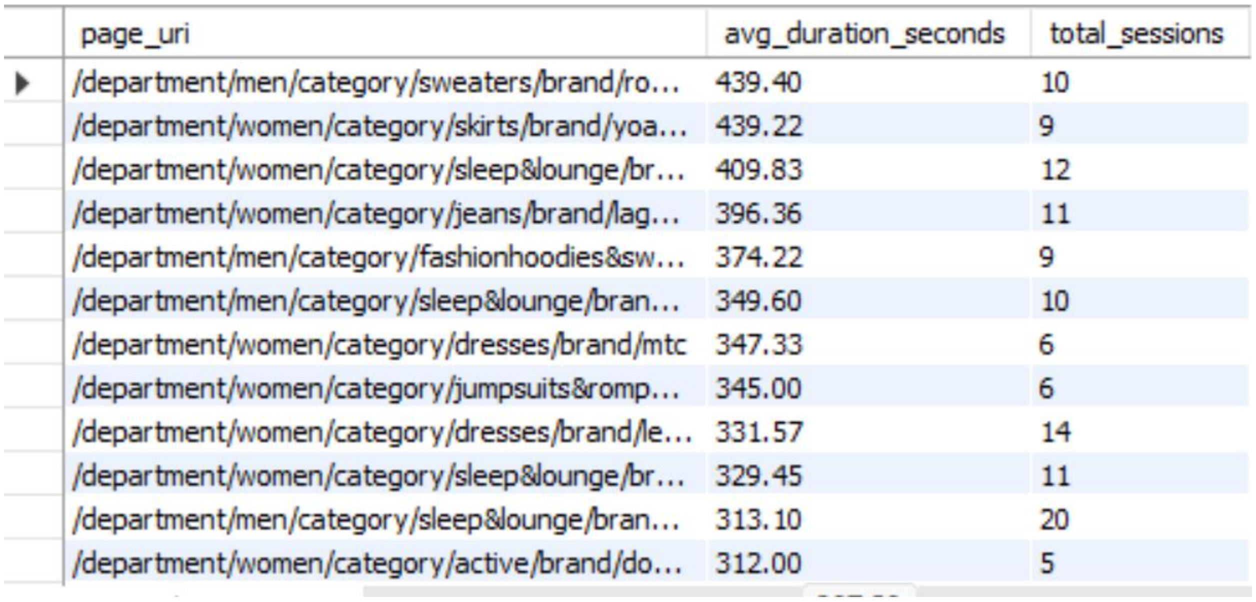
To measure **which pages and actions perform best** and detect any **bottlenecks** that slow down the user experience.
This is vital for both **UX optimization** and **technical SEO**.

View : avg_page_load_time

Description

Calculates the **average page load time** for each page (based on a hypothetical `page_load_time` column in events or performance logs).
Useful for identifying slow pages that degrade the user experience.

```sql
CREATE VIEW avg_page_duration AS
SELECT
    uri AS page_uri,
    ROUND(AVG(session_duration_seconds), 2) AS avg_duration_seconds,
    COUNT(*) AS total_sessions
FROM (
    SELECT
        session_id,
        uri,
        TIMESTAMPDIFF(SECOND, MIN(created_at), MAX(created_at)) AS
session_duration_seconds
    FROM events
    WHERE uri IS NOT NULL AND uri <> ''
    GROUP BY session_id, uri
) AS durations
GROUP BY uri
ORDER BY avg_duration_seconds DESC
LIMIT 50;
```



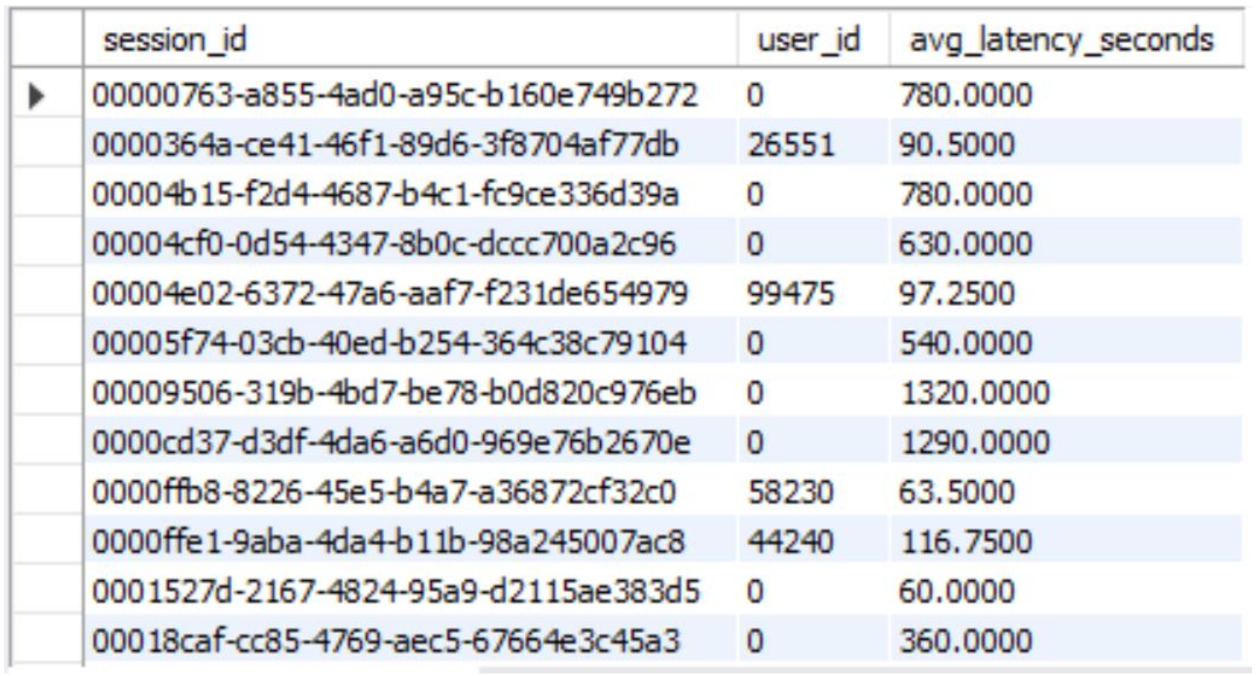
page_uri	avg_duration_seconds	total_sessions
/department/men/category/sweaters/brand/ro...	439.40	10
/department/women/category/skirts/brand/yoa...	439.22	9
/department/women/category/sleep&lounge/br...	409.83	12
/department/women/category/jeans/brand/lag...	396.36	11
/department/men/category/fashionhoodies&sw...	374.22	9
/department/men/category/sleep&lounge/bran...	349.60	10
/department/women/category/dresses/brand/mtc	347.33	6
/department/women/category/jumpsuits&romp...	345.00	6
/department/women/category/dresses/brand/le...	331.57	14
/department/women/category/sleep&lounge/br...	329.45	11
/department/men/category/sleep&lounge/bran...	313.10	20
/department/women/category/active/brand/do...	312.00	5

View : event_latency_analysis

Description

Measures the **time delay between consecutive events** for each session indicating performance issues or user hesitation points.

```
CREATE VIEW event_latency_analysis AS
SELECT
    session_id,
    user_id,
    AVG(TIMESTAMPDIFF(SECOND, prev_event_time, current_event_time))
AS avg_latency_seconds
FROM (
    SELECT
        session_id,
        user_id,
        created_at AS current_event_time,
        LAG(created_at) OVER (PARTITION BY session_id ORDER BY
created_at) AS prev_event_time
    FROM events
) AS t
WHERE prev_event_time IS NOT NULL
GROUP BY session_id, user_id;
```



session_id	user_id	avg_latency_seconds
00000763-a855-4ad0-a95c-b160e749b272	0	780.0000
0000364a-ce41-46f1-89d6-3f8704af77db	26551	90.5000
00004b15-f2d4-4687-b4c1-fc9ce336d39a	0	780.0000
00004cf0-0d54-4347-8b0c-dccc700a2c96	0	630.0000
00004e02-6372-47a6-aaf7-f231de654979	99475	97.2500
00005f74-03cb-40ed-b254-364c38c79104	0	540.0000
00009506-319b-4bd7-be78-b0d820c976eb	0	1320.0000
0000cd37-d3df-4da6-a6d0-969e76b2670e	0	1290.0000
0000ffb8-8226-45e5-b4a7-a36872cf32c0	58230	63.5000
0000ffe1-9aba-4da4-b11b-98a245007ac8	44240	116.7500
0001527d-2167-4824-95a9-d2115ae383d5	0	60.0000
00018caf-cc85-4769-aec5-67664e3c45a3	0	360.0000

View : error_event_summary

Description

Tracks how many **error-related events** occur (like `404`, `timeout`, or JS errors), and which pages are most affected.
Helps debug **frontend or backend reliability issues**.

Category 4: Retention & Returning Users

View 1 — returning_users_summary

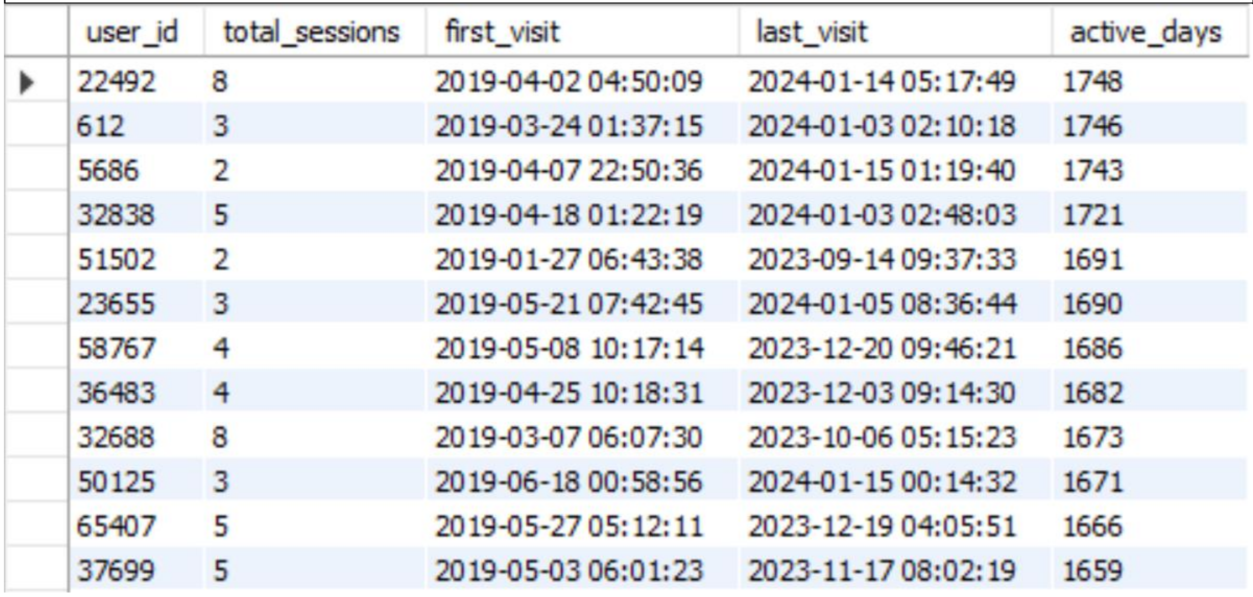
Explanation (English):
This view identifies users who returned to the website more than once.

- It **excludes guest users** (`user_id = 0`) because they are not uniquely trackable.
- For each registered user, we count the number of **distinct sessions** to measure engagement.
- `MIN(created_at)` shows the first visit and `MAX(created_at)` the last visit.
- `TIMESTAMPDIFF` calculates how many days passed between the first and last visit an indicator of user activity over time.

- Only users with more than one session are included (`HAVING total_sessions > 1`).

This helps track long-term engagement and retention patterns.

```
CREATE VIEW returning_users_summary AS
SELECT
    user_id,
    COUNT(DISTINCT session_id) AS total_sessions,
    MIN(created_at) AS first_visit,
    MAX(created_at) AS last_visit,
    TIMESTAMPDIFF(DAY, MIN(created_at), MAX(created_at)) AS
active_days
FROM events
WHERE user_id <> 0
GROUP BY user_id
HAVING total_sessions > 1
ORDER BY active_days DESC;
```



user_id	total_sessions	first_visit	last_visit	active_days
22492	8	2019-04-02 04:50:09	2024-01-14 05:17:49	1748
612	3	2019-03-24 01:37:15	2024-01-03 02:10:18	1746
5686	2	2019-04-07 22:50:36	2024-01-15 01:19:40	1743
32838	5	2019-04-18 01:22:19	2024-01-03 02:48:03	1721
51502	2	2019-01-27 06:43:38	2023-09-14 09:37:33	1691
23655	3	2019-05-21 07:42:45	2024-01-05 08:36:44	1690
58767	4	2019-05-08 10:17:14	2023-12-20 09:46:21	1686
36483	4	2019-04-25 10:18:31	2023-12-03 09:14:30	1682
32688	8	2019-03-07 06:07:30	2023-10-06 05:15:23	1673
50125	3	2019-06-18 00:58:56	2024-01-15 00:14:32	1671
65407	5	2019-05-27 05:12:11	2023-12-19 04:05:51	1666
37699	5	2019-05-03 06:01:23	2023-11-17 08:02:19	1659

View 2 — user_retention_rate

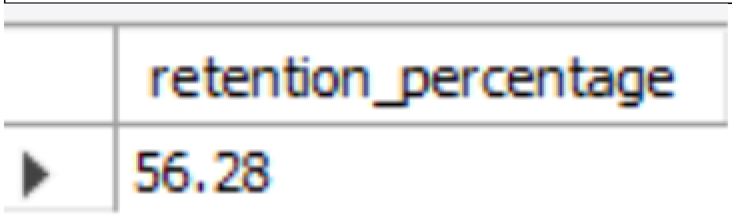
Explanation :
This view calculates the **overall retention rate** the percentage of users who returned to the website after their first visit.

- We first group by `user_id` and count how many unique sessions each user had.

- Then, we count how many users had more than one session (`total_sessions > 1`).
- Finally, we divide that number by the total number of unique users and multiply by 100 to get a percentage.
- The result is rounded to two decimal places for clarity.

This view gives a **single KPI metric** showing how engaging the website is for registered users.

```
CREATE VIEW user_retention_rate AS
SELECT
    ROUND(
        (COUNT(DISTINCT CASE WHEN total_sessions > 1 THEN user_id
END)
        / COUNT(DISTINCT user_id)) * 100, 2
    ) AS retention_percentage
FROM (
    SELECT
        user_id,
        COUNT(DISTINCT session_id) AS total_sessions
    FROM events
    WHERE user_id <> 0
    GROUP BY user_id
) AS user_sessions;
```



	retention_percentage
▶	56.28

COO

The COO oversees daily operations, logistics, and processes. They ensure that orders, deliveries, and customer service run smoothly and efficiently.

Category 1: Customer Activity Overview

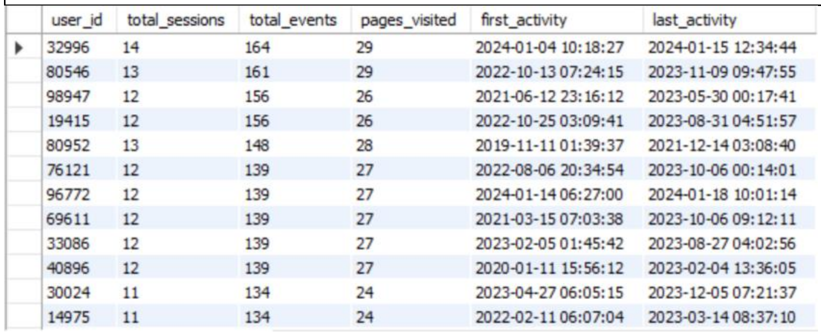
View 1 — customer_activity_summary

Explanation :
This view provides a general overview of each user's activity on the website.

- It includes **only registered users** (`user_id <> 0`).
- For each user, it counts total sessions, events, and unique pages visited.
- The earliest and latest activity timestamps show how long the user has been active.
- Sorting by `total_events` highlights the most engaged users.

This is ideal for identifying **high-value or highly active customers**.

```sql
CREATE VIEW customer_activity_summary AS
SELECT
    user_id,
    COUNT(DISTINCT session_id) AS total_sessions,
    COUNT(*) AS total_events,
    COUNT(DISTINCT uri) AS pages_visited,
    MIN(created_at) AS first_activity,
    MAX(created_at) AS last_activity
FROM events
WHERE user_id <> 0
GROUP BY user_id
ORDER BY total_events DESC;
```



user_id	total_sessions	total_events	pages_visited	first_activity	last_activity
32996	14	164	29	2024-01-04 10:18:27	2024-01-15 12:34:44
80546	13	161	29	2022-10-13 07:24:15	2023-11-09 09:47:55
98947	12	156	26	2021-06-12 23:16:12	2023-05-30 00:17:41
19415	12	156	26	2022-10-25 03:09:41	2023-08-31 04:51:57
80952	13	148	28	2019-11-11 01:39:37	2021-12-14 03:08:40
76121	12	139	27	2022-08-06 20:34:54	2023-10-06 00:14:01
96772	12	139	27	2024-01-14 06:27:00	2024-01-18 10:01:14
69611	12	139	27	2021-03-15 07:03:38	2023-10-06 09:12:11
33086	12	139	27	2023-02-05 01:45:42	2023-08-27 04:02:56
40896	12	139	27	2020-01-11 15:56:12	2023-02-04 13:36:05
30024	11	134	24	2023-04-27 06:05:15	2023-12-05 07:21:37
14975	11	134	24	2022-02-11 06:07:04	2023-03-14 08:37:10

Category 2: Customer Engagement & Conversion

View 1 — conversion_funnel_summary

Explanation :

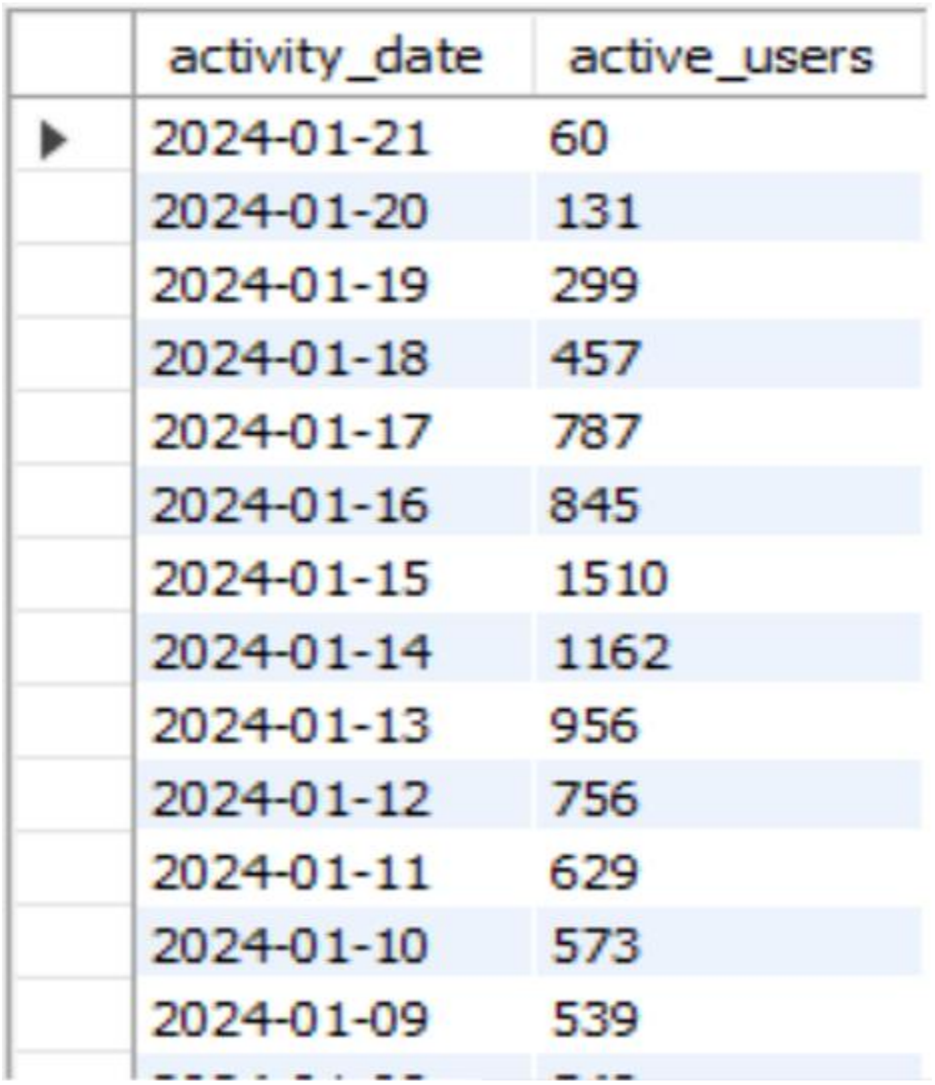
This view shows how many **unique users** were active per day.

- Uses `DATE(created_at)` to group events by day.
- Excludes anonymous visitors (`user_id = 0`).
- Works fast because it only groups by date — no joins or nested conditions.

Useful for:

- Tracking user retention.
- Measuring marketing campaign impact.
- Feeding Power BI trend charts easily.

```sql
CREATE OR REPLACE VIEW daily_active_users AS
SELECT
    DATE(created_at) AS activity_date,
    COUNT(DISTINCT user_id) AS active_users
FROM events
WHERE user_id <> 0
GROUP BY DATE(created_at)
ORDER BY activity_date DESC;
```



activity_date	active_users
2024-01-21	60
2024-01-20	131
2024-01-19	299
2024-01-18	457
2024-01-17	787
2024-01-16	845
2024-01-15	1510
2024-01-14	1162
2024-01-13	956
2024-01-12	756
2024-01-11	629
2024-01-10	573
2024-01-09	539

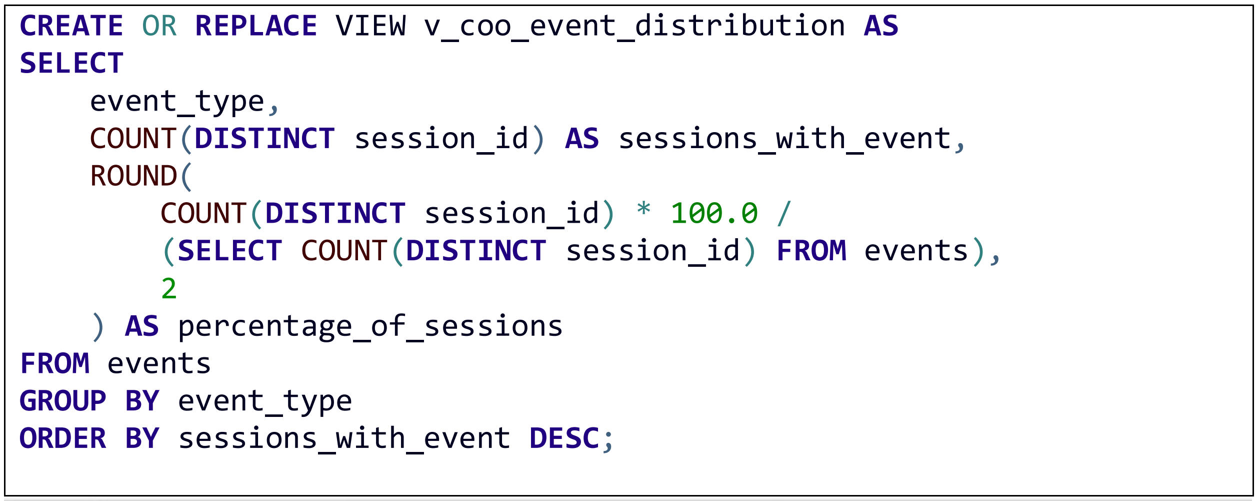
Category 3: Operations & Efficiency

View 1: Event Type Distribution

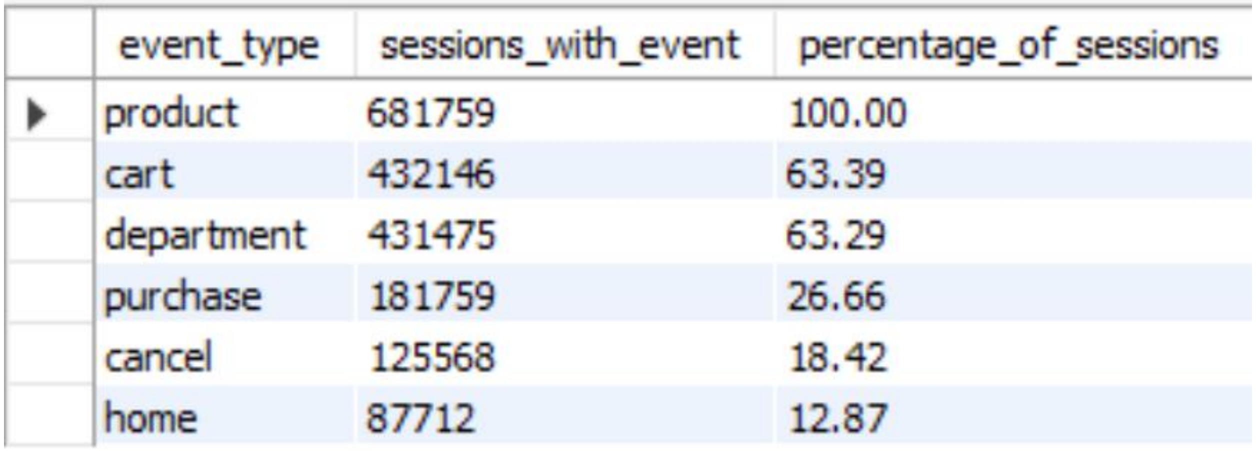
Explanation :

This view shows what types of events (like view, add_to_cart, purchase, etc.) happen most often and their percentage of total activity.

It helps the COO understand how the system is being used and identify any imbalance (e.g., too many errors, too few purchases).



```
CREATE OR REPLACE VIEW v_coo_event_distribution AS
SELECT
    event_type,
    COUNT(DISTINCT session_id) AS sessions_with_event,
    ROUND(
        COUNT(DISTINCT session_id) * 100.0 /
        (SELECT COUNT(DISTINCT session_id) FROM events),
        2
    ) AS percentage_of_sessions
FROM events
GROUP BY event_type
ORDER BY sessions_with_event DESC;
```



event_type	sessions_with_event	percentage_of_sessions
product	681759	100.00
cart	432146	63.39
department	431475	63.29
purchase	181759	26.66
cancel	125568	18.42
home	87712	12.87