Autodoc Data Analyst Assignment

Candidate: Giovanni Faria Giraldeli
Reference: November 2024

Table of Contents:

Context

The product owner wants to change the marketing strategy based on customers' behavior.

1. We need to build a custom purchase funnel based on the tracking data, the funnel will show the customer journey from the moment of visiting the website, and viewing a product or listing, to the moment of purchase. The main marketing funnel steps you can add or define yourself. Your main goal is to find details on every stage of the marketing funnel.

2. The achieved result must be fixed in metrics that, in your view, describe the user behavior on the site.

3. What additional data/information would you need to know the effectiveness of the current strategy?

4. The resulting analytical solution should show how the customer journey varies depending on what page_type was visited first in the session.

Deliverables:

• The achieved result must be presented in form of a report/dashboard with any suitable BI tool.

• Calculations are done in SQL or Python.

• The results of the analysis and the logic of the calculations should be presented to the product team informatively and to be understandable to the entire team, which includes technical and non-technical specialists.

Dataset description:

Filed	Description
event_date	The time the event occurred
session	Session ID
user	User ID
page_type	Type of the page
event_type	Type of the event
product	Product ID

Summary

Before creating the dashboard, an exploratory data analysis was executed to understand the data behavior.

Afterwards, there was a metric planning routine to define how the data should be modeled to supply the requested dashboard.

The following metrics were defined: Events Conversion, Visits Conversion, Unique Users Conversion and Bounce Rate.

Additionally, it was suggested the ingestion of the following data: product details, marketing channels details, user demographic details, device details and platform details.

Materializing the data model, a dashboard was created to track the site conversion, mapping the funnel performance key points.

The last part of this document addresses the SQL tasks requested in the briefing.

Exploratory Data Analysis

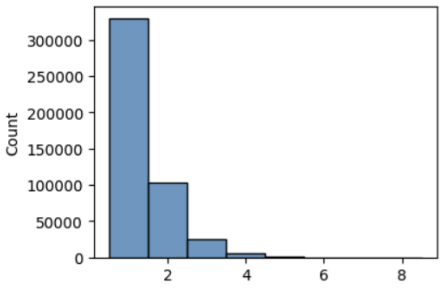
After analyzing the data profile, we identified that there are no NULLs in the dataset provided, meaning that the data is 100% complete.

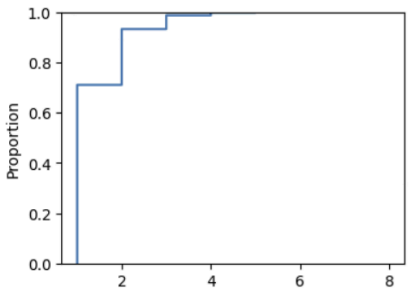
With this in mind, we can state that handling NULLs is unnecessary and we can start drilling down the data to dissect patterns and extract insights. Firstly we'll understand the data distribution and afterwards start designing correlations.

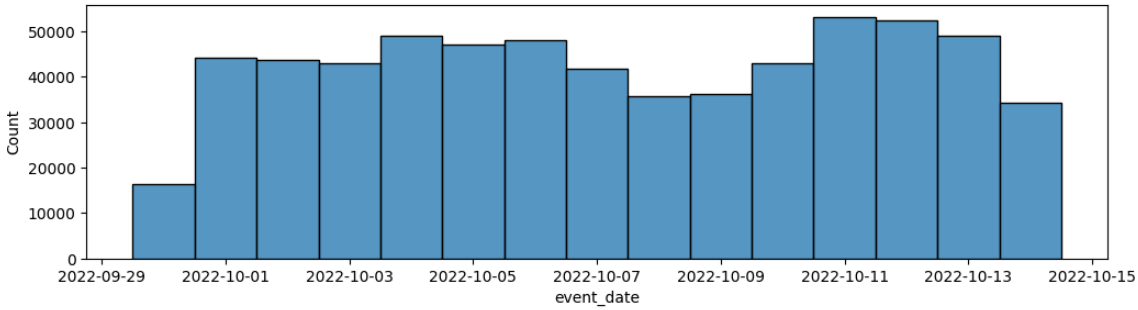
Features distributions

- event_date
    - Most of the records are unique (52%), but some coincide, meaning that some records were triggered concurrently.
    - It has data from 2022-09-30 to 2022-10-13.

	count
1	330033
2	103238
3	25438
4	4880
5	825
6	106
7	18
8	1

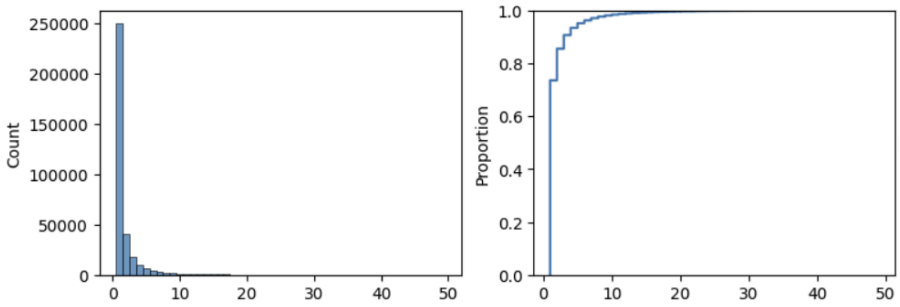






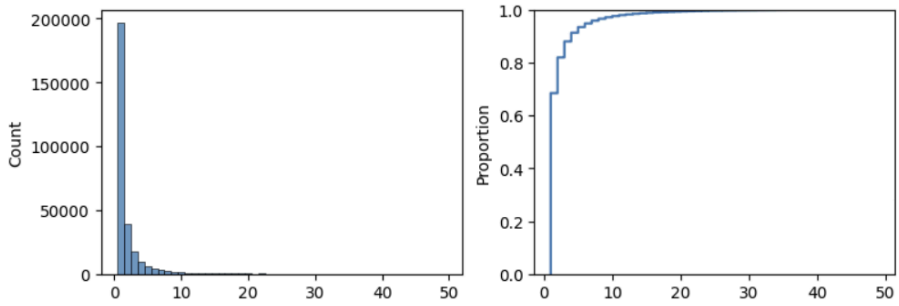
- session
    - Has a long-tail behavior, with most of the sessions containing only a single event, but with one reaching 1,264 events.
    - Sessions with 105 events occurred three times, while sessions with higher events count occurred only once each.
    - Sessions with 46 events occurred fifteen times, while sessions with higher events count occurred less than ten times each.

count	340443.000000
mean	1.871791
std	3.692538
min	1.000000
50%	1.000000
75%	2.000000
90%	3.000000
95%	5.000000
99%	14.000000
99.5%	19.000000
max	1264.000000



- user
    - Has a distribution similar to the one observed in the variable “session”.
    - Users with 163 events occurred two times, while users with higher events count occurred only once each.
    - Users with 60 events occurred twelve times, while users with higher events count occurred less than ten times each.

count	288088.000000
mean	2.211956
std	6.307799
min	1.000000
50%	1.000000
75%	2.000000
90%	4.000000
95%	7.000000
99%	17.000000
99.5%	24.000000
max	1266.000000



- page_type
    - Only has 4 distinct records: [product_page, listing_page, search_listing_page, order_page]
    - The order page is the least visited, which was expected since usually this is the last funnel step.

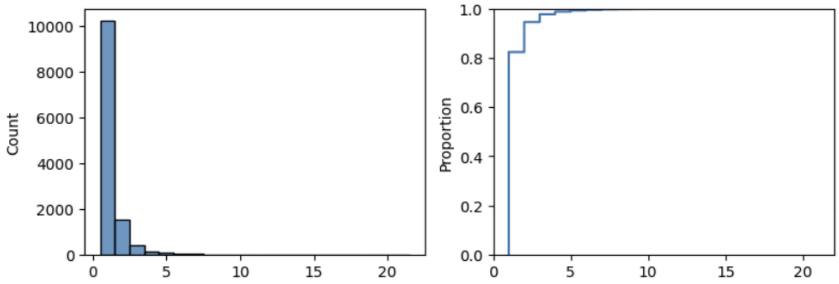
page_type	count
product_page	282950
listing_page	231789
search_listing_page	113758
order_page	8741

- event_type
    - Only has 3 distinct records: [page_view, add_to_cart, order]
    - The order page is the least here, but called the attention that add_to_cart is roughly the double of it. This could indicate that the users had the intention to buy, but didn't finish the purchase, or that a user tends to buy multiple products in a single purchase.
        - Assumption inspected further in this same document.

event_type	count
page_view	612498
add_to_cart	15999
order	8741

- product
    - Most of the values were flagged as "0" (97% of the total), likely because there was no product associated with the related event.
    - The sum of the records containing a product different than "0" results in 15,999, matching the exact count of "add_to_cart" events from the event_type table. This can mean that we only have the product associated with the add_to_cart probably.
        - Assumption confirmed further in this same document.
    - Most of the products (64% of the non-zeros) only have a single event, but with one reaching 21 events.

1	10248
2	1519
3	393
4	135
5	57
6	34
7	23
8	10
9	6
10	3
11	4
12	3
13	2
15	1
18	1
20	1
21	1



Assumptions validation

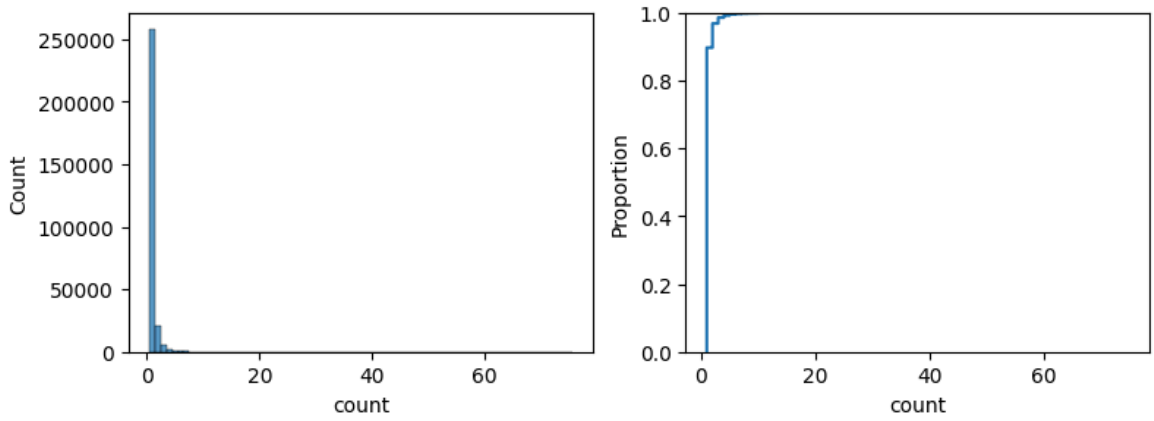
Before starting deeper analysis, we still need to understand how the variables are related to each other and check data quality with the expected behavior.

- **Sessions should have a single user:** it's expected that each session is related to a unique user, but the opposite could exist.

  Analyzing this pair of variables we could confirm this expected behavior.

- **Users can have multiple sessions:** it's expected some users to have multiple sessions, indicating that they accessed the platform multiple times.

  Most of the users visited only once, while one of them had 75 sessions in this 15 days period.



- **Product ID only exists when adding to cart:** in the distribution, we made an assumption based on the count of the "product" that only the "event_type" = "add_to_cart" has a product ID associated.

  Here we could confirm the assumption, making explicit that we don't have the product ID associated with the page views, nor directly with the order.

has_product_id	event_type	count
0	page_view	612498
1	add_to_cart	15999
0	order	8741

- **Users buy multiple products in a single order:** we identified in the distributions that the volume of orders is roughly half if compared with the volume of products added to cart. We need to confirm how much of this decrease is due to users dropping the funnel.

Inspecting the session profile, we could see that the proportion of drop is based on the sessions that have orders is 28.4%, indicating that some users bought multiple products in a single order. The conversion based on events was 45.4% (8,741 orders divided by 15,999 add_to_cart).

	has_add_to_cart	has_order	count	total_sessions_count	add_to_cart_count	add_order_count	cart_to_order_conversion
1	0	1	3,388	14,055	10,667	7,637	0.716
2	1	0	6,418	14,055	10,667	7,637	0.716
3	1	1	4,249	14,055	10,667	7,637	0.716

Dashboard

Now that we understand the data better and how it behaves, we are ready to create a data model to supply our sales funnel dashboard.

Metrics Definition

Before creating the dashboard, we need to define which metrics will be used to track de funnel performance. This step will impact on how the data will be modeled to supply the BI tool.

Supporting dimensions definitions:

- **Customer:** an user that has ordered a product on the platform.
- **Landing Page:** first page visited by a user in a session.
- **Bounce:** session with only one event, and this event being page_view.

Metrics definitions:

- **Events Conversion:** number of orders divided by the total number of events.
- **Visits Conversion:** number of orders divided by the number of sessions.
- **Unique Users Conversion:** number of unique customers divided by the number of unique users.
- **Bounce Rate:** number of bounces divided by the number of page views.

Data Acquisition Suggestions

To drill segment even further the numbers, we could benefit having:

- **Product details:** categories and prices could eventually surface conversion patterns.
- **Marketing channel details:** we could calculate the performance for each source if we had described the channel and the money invested in each paid traffic partner.
- **User demographic details:** information about clients' region can support segmentation and optimize marketing campaigns.
- **Device and platform details:** it is relatively common to have different conversions for the app if compared with the website, similarly for smartphones compared with desktops.

Analytical Data Model

The main request is a daily report about the funnel conversion, so I'll aggregate the data to deliver this specific need in order to save compute and optimize query consumption by the BI tool.

Created the analytical data model and named it as sales_funnel_fact, the size of the table reduced significantly going from 637,238 rows to 343 rows. This reduction will mainly affect performance of retrieving the data in the dashboard and making it respond faster.

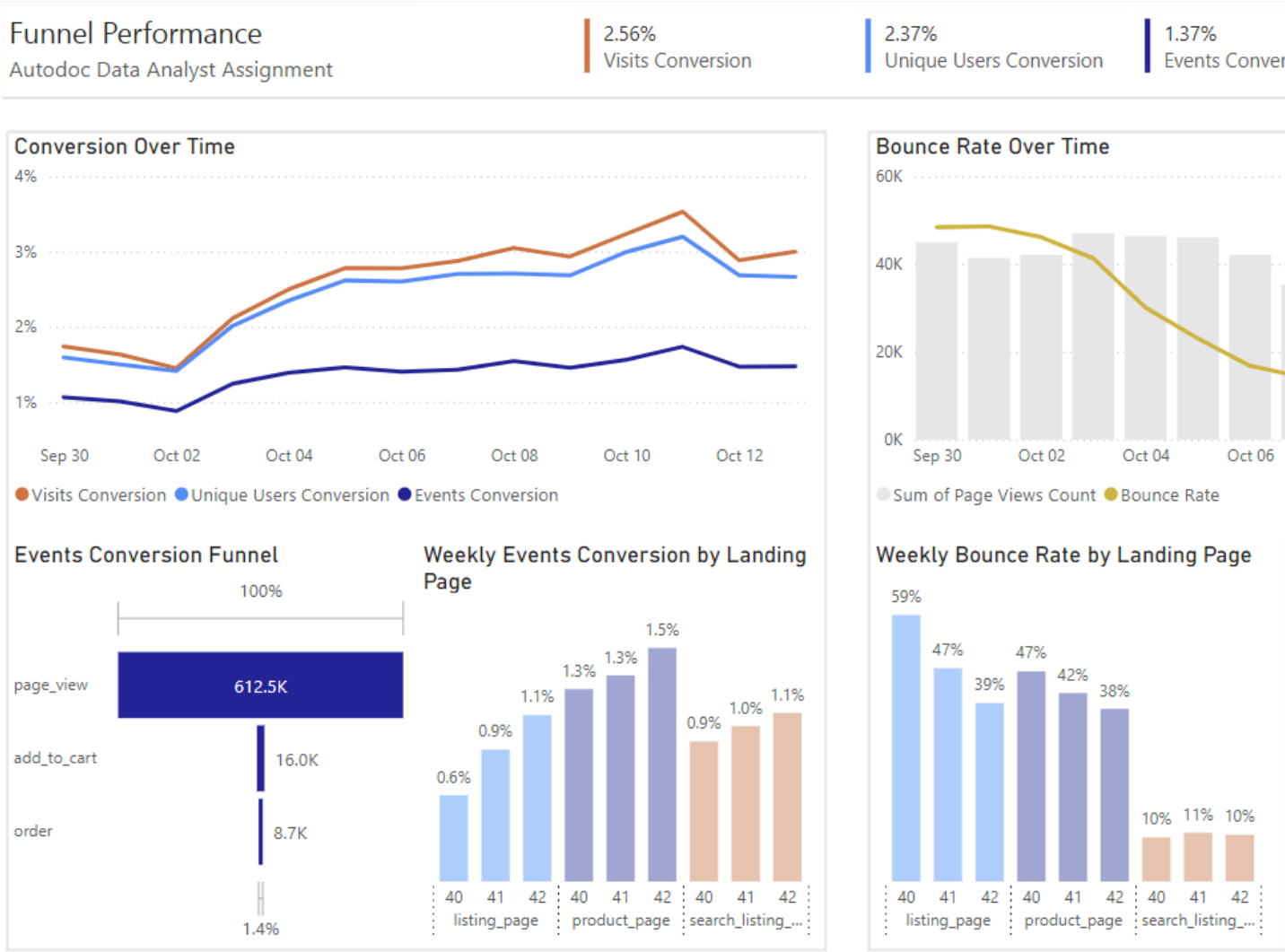
SQL Script

```
-----------------------------------------------
---- ANALYTICAL DATA MODEL ----
-----------------------------------------------
CREATE TABLE sales_funnel_fact AS
WITH bounces AS (
SELECT
	"session",
	MIN(event_date) AS min_event_date,
	COUNT(*) AS events_count
FROM
	event_source
GROUP BY
	"session"
)
, landing_page AS (
SELECT
	b."session",
	MIN(es.page_type) AS landing_page_type -- Eliminating concurrent events
FROM
	bounces AS b
LEFT JOIN
	event_source AS es
		ON es."session" = b."session"
		AND es.event_date = b.min_event_date
GROUP BY
	b."session"
)
, unique_values_count AS (
SELECT
	DATE(event_date) AS event_date,
	COUNT(DISTINCT CASE WHEN product = 0 THEN NULL ELSE product END ) AS
unique_products_count,
	COUNT(DISTINCT "user") AS unique_users_count,
	COUNT(DISTINCT "session") AS unique_sessions_count,
	COUNT(DISTINCT CASE WHEN event_type = 'order' THEN "user" ELSE NULL END ) AS
order_unique_users_count
```

```
FROM
	event_source
GROUP BY
	DATE(event_date)
)
SELECT
	ROW_NUMBER() OVER (ORDER BY
		DATE(es.event_date),
		es.page_type,
		lp.landing_page_type,
		es.event_type
	) AS sales_funnel_sk,
	DATE(es.event_date) AS event_date,
	es.page_type,
	lp.landing_page_type,
	es.event_type,
	COUNT(*) AS events_count,
	COUNT( DISTINCT es."session" ) AS session_count,
	SUM( CASE WHEN es.event_type = 'page_view' THEN 1 ELSE 0 END ) AS page_views_count,
	SUM( CASE WHEN es.product = 0 THEN 0 ELSE 1 END ) AS products_events_count,
	SUM( CASE WHEN es.event_type = 'order' THEN 1 ELSE 0 END ) AS order_events_count,
	SUM( CASE WHEN b.events_count = 1 AND es.event_type = 'page_view' THEN 1 ELSE 0 END
) AS bounces_count,
	AVG(uvc.unique_sessions_count) AS unique_sessions_count,
	AVG(uvc.unique_users_count) AS unique_users_count,
	AVG(uvc.unique_products_count) AS unique_products_count,
	SUM(
		SUM( CASE WHEN es.event_type = 'order' THEN 1 ELSE 0 END )
	) OVER ( PARTITION BY DATE(es.event_date) ) AS unique_orders_count,
	AVG(uvc.order_unique_users_count) AS order_unique_users_count
FROM
	event_source AS es
LEFT JOIN
	bounces AS b
		ON b."session" = es."session"
LEFT JOIN
	landing_page AS lp
		ON lp."session" = es."session"
LEFT JOIN
	unique_values_count AS uvc
		ON uvc.event_date = DATE(es.event_date)
GROUP BY
	DATE(es.event_date),
	es.page_type,
	lp.landing_page_type,
	es.event_type
;
```

Report

The report is the conclusion of all the previous work and it's mapping the previous surfaced points in the documentation.



SQL tasks

1. Write an SQL query that will return a number of clients by day that:
   - only viewed products in their first session;

2. Write a query that will return any abnormal (to our view) user behavior. Describe why the behavior is unusual.

Task 1

Comments

Although not required, I've created a fact table to describe better the relation between the user and their sessions. This could evolve depending on stakeholders needs in order to address more problems and business cases.

In parallel, I calculated the total events count and the total users count in the results. This wasn't required as well, but it gives a better sense on the magnitude of each metric and, therefore, the conversion to the actual requested metric.

This additional metric made explicit that between 2022-10-07 (Friday) and 2022-10-09 (Sunday) we had less activity on the platform, even if compared to the previous weekend.

In the first sight, I'd relate it to two possible factors:

1. A regional holiday on Friday that extended the weekend and decreased the activity.

2. Unavailability of the website.

SQL Script

```
-------------------------
---- SQL TASK 1 ----
-------------------------
CREATE TABLE user_session_fact AS
SELECT
	ROW_NUMBER() OVER (ORDER BY MIN(event_date), "user", "session") AS
user_session_sk,
	"user",
	"session",
	MIN(event_date) AS session_start,
	MAX(event_date) AS session_end,
	ROW_NUMBER() OVER (PARTITION BY "user" ORDER BY MIN(event_date)) AS
session_number,
	COUNT(*) AS event_count,
	MAX(
		CASE
```

```
                        WHEN page_type = 'product_page'
                        AND event_type = 'page_view'
                                THEN 1
                        ELSE 0
                END
        ) AS has_visited_product_page
FROM
        event_source
GROUP BY
        "user",
        "session"
;

WITH user_product_view_first_session AS (
SELECT
        "user",
        MAX(
                CASE
                        WHEN session_number > 1
                        AND has_visited_product_page = 1
                                THEN 0
                        WHEN session_number = 1
                        AND has_visited_product_page = 1
                                THEN 1
                        ELSE 0
                END
        ) AS has_only_product_view_in_first_session
FROM
        user_session_fact
GROUP BY
        "user"
)
SELECT
        DATE(es.event_date) AS event_date,
        COUNT(*) AS event_count,
        COUNT(DISTINCT es."user") AS user_count,
        COUNT(DISTINCT
                CASE WHEN upvfs.has_only_product_view_in_first_session = 1 THEN es."user" END
        ) AS prod_view_only_first_session_user_count
FROM
        event_source AS es
LEFT JOIN
        user_product_view_first_session AS upvfs
                ON es."user" = upvfs."user"
GROUP BY
        DATE(es.event_date)
;
```

Results

	event_date	event_count	user_count	prod_view_only_first_session_user_count
1	2022-09-30	46,323	26,566	10,670
2	2022-10-01	42,615	24,593	10,145
3	2022-10-02	43,286	24,668	9,875
4	2022-10-03	48,760	26,626	11,594
5	2022-10-04	48,235	24,856	11,844
6	2022-10-05	48,054	23,385	11,131
7	2022-10-06	43,800	20,605	10,931
8	2022-10-07	36,872	17,197	9,843
9	2022-10-08	35,763	17,023	9,269
10	2022-10-09	39,454	18,444	10,060
11	2022-10-10	52,758	23,832	13,044
12	2022-10-11	52,149	23,929	12,930
13	2022-10-12	49,759	23,737	11,580
14	2022-10-13	49,410	22,908	11,161

Task 2

Comments

It's possible to clearly identify an outlier in the query results presented below.

The user with the most events in the website also has a pattern to trigger events faster than the average.

Sessions with higher events count tend to have a lower pace to trigger events if compared with the overall average, but the session with the most events has pace even higher than the average.

Note: I used SQLite to create this solution, which doesn't have date difference functions, so I needed to use this non-traditional method of subtracting the dates (Julian days).

SQL Script

```
-------------------------
---- SQL TASK 2 ----
-------------------------
WITH session_duration AS (
SELECT
        *,
        (
                JULIANDAY(session_end) - JULIANDAY(session_start) -- Calculating the difference in days in SQLite
        ) * 24 * 60 AS session_duration_in_minutes -- Times 24 hours in a day, times 60 minutes per hour to get the results in minutes
FROM
```

```
        user_session_fact AS usf
)
SELECT
        "user",
        "session",
        event_count,
        ROUND(session_duration_in_minutes, 1) AS session_duration_in_minutes,
        ROUND(event_count / session_duration_in_minutes, 1) AS events_per_minutes,
        ROUND( AVG( event_count / session_duration_in_minutes ) OVER () , 1 ) AS
avg_events_per_minutes
FROM
        session_duration
ORDER BY
        event_count DESC
LIMIT 100;
```

Results

	user	session	event_count	session_duration_in_minutes	events_per_minutes	avg_events_per_minutes
1	11769744300065907078u	95436696421369858685	1,264	74.5	17	4.9
2	10671157392604114939u	148169635598133955935	346	334.4	1	4.9
3	14551769000936952084u	3509679068011838002s	251	344.2	0.7	4.9
4	96971359388688807 38u	716032141109111604 9s	203	351.9	0.6	4.9
5	26848678112939940 80u	64105486671764516 24s	193	198.5	1	4.9
6	97094936992280942 94u	118876763635198340 43s	187	119.5	1.6	4.9
7	67326784856730807 59u	15466953772873357 47s	175	152.1	1.2	4.9
8	97094936992280942 94u	103973033040250853 30s	170	115.7	1.5	4.9
9	14736585841709492857u	90349273562847852 94s	143	85.6	1.7	4.9
10	10671157392604114939u	43474710937063958 40s	131	236.2	0.6	4.9