Data Model & Calculation Logic

This document explains how raw search analytics events are transformed into meaningful metrics. It covers event sequences, timing calculations, and business rules with clear examples.

Table of Contents

1. Event Types & Sequence

Search Event Types

The search system generates these event types (stored in the `name` column):

Event	Description	When Fired
`SEARCH_STARTED`	User initiates a search	User types query and presses Enter
`SEARCH_COMPLETED`	Search query submitted to backend	Query sent to search service
`SEARCH_RESULT_COUNT`	Results returned to user	Search results displayed
`SEARCH_TAB_CLICK`	User clicks a General result	Click on main search tab
`SEARCH_ALL_TAB_PAGE_CLICK`	User clicks an All tab result	Click on All tab
`SEARCH_NEWS_TAB_PAGE_CLICK`	User clicks a News result	Click on News tab
`SEARCH_GOTO_TAB_PAGE_CLICK`	User clicks a GoTo result	Click on GoTo tab
`SEARCH_PEOPLE_*`	User clicks a People result	Click on People tab

Typical Event Sequence

```
User types "project budget" and presses Enter
    |
    v
[SEARCH_STARTED]  <-- timestamp: 10:30:15.123
    |
    v
[SEARCH_COMPLETED]  <-- timestamp: 10:30:15.234  (111ms later)
    |
    v
[SEARCH_RESULT_COUNT]  <-- timestamp: 10:30:15.567  (333ms after COMPLETED)
    |                                               (444ms after STARTED)
    v
User sees results, clicks one
    |
    v
[SEARCH_TAB_CLICK]  <-- timestamp: 10:30:18.890  (3.3s after results shown)
```

Example: Complete Session

```
Session: 2025-01-15_user123_session456

Event 1: SEARCH_STARTED      @ 10:30:15.123   (search term: "budget report")
Event 2: SEARCH_COMPLETED    @ 10:30:15.234
Event 3: SEARCH_RESULT_COUNT @ 10:30:15.567   (15 results found)
Event 4: SEARCH_TAB_CLICK    @ 10:30:18.890   (user clicked a result)
Event 5: SEARCH_STARTED      @ 10:30:45.000   (user searches again: "2024 budget")
Event 6: SEARCH_COMPLETED    @ 10:30:45.100
Event 7: SEARCH_RESULT_COUNT @ 10:30:45.400   (8 results found)
Event 8: SEARCH_TAB_CLICK    @ 10:30:52.500   (user clicked another result)
```

2. Processing Pipeline Overview

Data Flow

```
                 KQL Export (CSV/Excel)
                         |
                         v
              +---------------------+
              |   Raw Data Import   |
              |  (load_file_to_     |
              |   temp_table)       |
              +---------------------+
                         |
         - Parse timestamps (preserve milliseconds)
         - Normalize column names
         - Handle Excel datetime precision issues
                         |
                         v
              +---------------------+
              |  Column Enrichment  |
              |  (add_calculated_   |
              |   columns)          |
              +---------------------+
                         |
         - Normalize event names to UPPERCASE
         - Calculate session_key
         - Add timing columns (prev_event, ms_since_prev)
         - Derive flags (is_null_result, click_category)
         - Calculate search term metrics
                         |
                         v
              +---------------------+
              |   Parquet Export    |
              |  (export_parquet_   |
              |   files)            |
              +---------------------+
                    /    |    \
                   /     |     \
                  v      v      v
           +-------+ +--------+ +-------+
           | Raw   | | Daily  | | Terms |
           +-------+ +--------+ +-------+
                     +----------+
                     | Journeys |
                     +----------+
```

Key Transformations

1. Event Name Normalization

Raw event names come in mixed case from App Insights. We normalize to uppercase for consistent matching.

```
Input:  "Search_completed"
Output: "SEARCH_COMPLETED"
```

2. Session Key Generation

A unique session is identified by combining date + user + session ID:

```
session_key = session_date || '_' || user_id || '_' || session_id
-- Example: "2025-01-15_user123_abc789"
```

3. Search Term Normalization

Search terms are cleaned for consistent aggregation:

```
search_term_normalized = LOWER(TRIM(COALESCE(CP_searchQuery, searchQuery, query)))
-- Input:  "  Budget Report  "
-- Output: "budget report"
```

3. Timing Calculations

ms_search_to_result (User-Perceived Latency)

What it measures: The time from when a user initiates a search until they see results.

Event span: `SEARCH_STARTED` --> `SEARCH_RESULT_COUNT`

How it's calculated:

```
-- Step 1: Track the most recent SEARCH_STARTED timestamp
last_search_started_ts = LAST_VALUE(
    CASE WHEN name = 'SEARCH_STARTED' THEN timestamp END
    IGNORE NULLS
) OVER (PARTITION BY session_key ORDER BY timestamp)

-- Step 2: Calculate time difference when SEARCH_RESULT_COUNT occurs
ms_search_to_result = DATEDIFF('millisecond', last_search_started_ts, timestamp)
-- Only when name = 'SEARCH_RESULT_COUNT'
```

Example:

```
Event: SEARCH_STARTED      @ 10:30:15.123
Event: SEARCH_COMPLETED    @ 10:30:15.234
Event: SEARCH_RESULT_COUNT @ 10:30:15.567

ms_search_to_result = 10:30:15.567 - 10:30:15.123 = 444ms
```

ms_result_to_click (Decision Time)

What it measures: How long the user takes to click a result after seeing search results.

Event span: `SEARCH_RESULT_COUNT` --> `Click Event`

How it's calculated:

```
ms_result_to_click = ms_since_prev_event
-- Only when click_category IS NOT NULL AND prev_event = 'SEARCH_RESULT_COUNT'
```

Example:

```
Event: SEARCH_RESULT_COUNT @ 10:30:15.567
Event: SEARCH_TAB_CLICK    @ 10:30:18.890

ms_result_to_click = 10:30:18.890 - 10:30:15.567 = 3,323ms (3.3 seconds)
```

ms_since_prev_event (Inter-Event Timing)

What it measures: Time between any two consecutive events in a session.

```
ms_since_prev_event = DATEDIFF('millisecond',
    LAG(timestamp) OVER (PARTITION BY session_key ORDER BY timestamp),
    timestamp
)
```

Example:

```
Event 1: SEARCH_STARTED      @ 10:30:15.123  --> ms_since_prev = NULL (first event)
Event 2: SEARCH_COMPLETED    @ 10:30:15.234  --> ms_since_prev = 111ms
Event 3: SEARCH_RESULT_COUNT @ 10:30:15.567  --> ms_since_prev = 333ms
Event 4: SEARCH_TAB_CLICK    @ 10:30:18.890  --> ms_since_prev = 3,323ms
```

Time Buckets

Timing values are bucketed for easier visualization:

Metric	Bucket	Range

Metric	Bucket	Range
search_to_result	`< 0.5s`	0-499ms
	`0.5-1s`	500-999ms
	`1-2s`	1000-1999ms
	`2-5s`	2000-4999ms
	`> 5s`	5000ms+
	`No Result`	NULL (no SEARCH_RESULT_COUNT event)
result_to_click	`< 2s (quick)`	0-1999ms
	`2-5s`	2000-4999ms
	`5-10s`	5000-9999ms
	`10-30s`	10000-29999ms
	`30-60s`	30000-59999ms
	`> 60s (browsing)`	60000ms+
	`No Click`	NULL (user didn't click)

4. Business Rules & Classifications

is_null_result

Definition: The search returned zero results.

```
is_null_result = CASE
    WHEN name = 'SEARCH_RESULT_COUNT' AND CP_totalResultCount = 0 THEN true
    WHEN name = 'SEARCH_RESULT_COUNT' AND CP_totalResultCount > 0 THEN false
    ELSE NULL  -- Only meaningful for SEARCH_RESULT_COUNT events
END
```

Example:

```
Event: SEARCH_RESULT_COUNT with CP_totalResultCount = 0
--> is_null_result = true (user saw "No results found")

Event: SEARCH_RESULT_COUNT with CP_totalResultCount = 15
--> is_null_result = false (user saw 15 results)
```

click_category

Definition: Categorizes click events by which tab/section was clicked.

```
click_category = CASE
    WHEN name = 'SEARCH_TAB_CLICK' THEN 'General'
    WHEN name = 'SEARCH_ALL_TAB_PAGE_CLICK' THEN 'All'
    WHEN name = 'SEARCH_NEWS_TAB_PAGE_CLICK' THEN 'News'
    WHEN name = 'SEARCH_GOTO_TAB_PAGE_CLICK' THEN 'GoTo'
    WHEN name LIKE '%PEOPLE%' THEN 'People'
    ELSE NULL  -- Not a click event
END
```

journey_outcome (Session-Level)

Definition: Classifies how a search session ended.

```
journey_outcome = CASE
    WHEN click_count > 0 THEN 'Success'
    WHEN result_count > 0 AND null_result_count = result_count AND click_count = 0
        THEN 'No Results'
    WHEN result_count > 0 AND click_count = 0 THEN 'Abandoned'
    ELSE 'Unknown'
END
```

Example scenarios:

Scenario	click_count	result_count	null_result_count	Outcome
User searched, clicked a result	1	1	0	**Success**
User searched, got 0 results	0	1	1	**No Results**
User searched, saw results but didn't click	0	1	0	**Abandoned**
Incomplete session data	0	0	0	**Unknown**

session_complexity

Definition: Categorizes sessions by number of events.

```
session_complexity = CASE
    WHEN total_events = 1 THEN 'Single Event'
    WHEN total_events <= 3 THEN 'Simple'
    WHEN total_events <= 10 THEN 'Medium'
    ELSE 'Complex'
END
```

had_reformulation

Definition: Did the user refine/change their search query within the session?

```
had_reformulation = CASE
    WHEN unique_search_terms > 1 THEN true
    ELSE false
END
```

Example:

```
Session with searches: "budget", "2024 budget", "budget report Q4"
--> unique_search_terms = 3
--> had_reformulation = true (user refined their search)
```

recovered_from_null

Definition: Did the user eventually find something despite getting zero results initially?

```
recovered_from_null = CASE
    WHEN null_result_count > 0 AND click_count > 0 THEN true
    ELSE false
END
```

Example:

```
Session: Search "bugdet" (typo) --> 0 results
         Search "budget" --> 15 results --> Click
--> null_result_count = 1, click_count = 1
--> recovered_from_null = true
```

User Cohort: is_users_first_session

Definition: Is this the first time we've seen this user search?

```
user_session_number = ROW_NUMBER() OVER (
    PARTITION BY user_id
    ORDER BY session_start
)
is_users_first_session = CASE WHEN user_session_number = 1 THEN true ELSE false END
```

New vs Returning Users (Daily)

Definition: Count of users who are new vs returning on each day.

```
-- First, find when each user first appeared
first_seen_date = MIN(session_date) GROUP BY user_id

-- Then classify on each day
new_users = COUNT(DISTINCT CASE WHEN session_date = first_seen_date THEN user_id END)
returning_users = COUNT(DISTINCT CASE WHEN session_date > first_seen_date THEN user_id END)
```

5. Output Files & Column Definitions

searches_raw.parquet

Granularity: One row per event (click, search, result)

Use case: Detailed event-level analysis, debugging

Column	Type	Description	Example
`timestamp`	Timestamp	Event timestamp (microsecond precision)	2025-01-15 10:30:15.567123
`name`	String	Event type (normalized to uppercase)	SEARCH_RESULT_COUNT
`user_id`	String	Anonymous user identifier	user_abc123
`session_id`	String	Session identifier	sess_xyz789
`session_key`	String	Composite key: date_user_session	2025-01-15_user_abc123_sess_xyz789
`session_date`	Date	Date of the event	2025-01-15
`event_order`	Integer	Sequence number within session	3
`prev_event`	String	Previous event type in session	SEARCH_COMPLETED
`ms_since_prev_event`	Integer	Milliseconds since previous event	333
`search_term_normalized`	String	Cleaned search query	budget report
`is_null_result`	Boolean	True if zero results returned	false
`click_category`	String	Click type (General/All/News/GoTo/People)	General
`last_search_started_ts`	Timestamp	Most recent SEARCH_STARTED timestamp	2025-01-15 10:30:15.123

searches_journeys.parquet

Granularity: One row per search session

Use case: Session-level behavior analysis, funnel metrics

Column	Type	Description	Calculation
`session_date`	Date	Date of session	
`session_start`	Timestamp	First event timestamp	MIN(timestamp)
`session_start_str`	String	Session start as string	STRFTIME for Power BI compatibility
`total_events`	Integer	Events in session	COUNT(*)
`search_count_in_session`	Integer	SEARCH_STARTED events	COUNT(SEARCH_STARTED)
`result_count`	Integer	SEARCH_RESULT_COUNT events	COUNT(SEARCH_RESULT_COUNT)
`click_count`	Integer	Click events	COUNT(click_category IS NOT NULL)
`unique_search_terms`	Integer	Distinct queries	COUNT(DISTINCT search_term)
`null_result_count`	Integer	Zero-result events	SUM(is_null_result)
`max_total_results`	Integer	Max results shown	MAX(CP_totalResultCount)
`sec_search_to_result`	Float	Seconds: search to results	MIN(ms_search_to_result) / 1000
`sec_result_to_click`	Float	Seconds: results to click	MIN(ms_result_to_click) / 1000
`total_duration_sec`	Float	Session length in seconds	(MAX - MIN timestamp) / 1000
`first_event_hour`	Integer	Hour of first event (0-23)	MIN(event_hour)
`last_event_hour`	Integer	Hour of last event (0-23)	MAX(event_hour)
`general_clicks`	Integer	General tab clicks	COUNT(click_category='General')
`all_tab_clicks`	Integer	All tab clicks	COUNT(click_category='All')

Column	Type	Description	Calculation
news_clicks	Integer	News tab clicks	COUNT(click_category='News')
goto_clicks	Integer	GoTo tab clicks	COUNT(click_category='GoTo')
people_clicks	Integer	People tab clicks	COUNT(click_category='People')
includes_first_search_of_day	Boolean	Session has day's first search	MAX(is_first_search_of_day)
search_to_result_bucket	String	Latency category	See Time Buckets
result_to_click_bucket	String	Decision time category	See Time Buckets
session_duration_bucket	String	Session length category	< 5s, 5-30s, 30-60s, 1-3 min, etc.
journey_outcome	String	Session result	Success/No Results/Abandoned
had_reformulation	Boolean	User changed query	unique_search_terms > 1
session_complexity	String	Session size category	Based on total_events
search_to_result_sort	Integer	Sort order for latency bucket	1-6 for Power BI sorting
result_to_click_sort	Integer	Sort order for click time bucket	1-7 for Power BI sorting
session_duration_sort	Integer	Sort order for duration bucket	1-6 for Power BI sorting
journey_outcome_sort	Integer	Sort order for outcome	1=Success, 2=Abandoned, 3=No Results
session_complexity_sort	Integer	Sort order for complexity	1-4 for Power BI sorting
had_null_result	Boolean	Had zero-result search	null_result_count > 0
recovered_from_null	Boolean	Success despite null result	null_result > 0 AND click > 0
user_session_number	Integer	User's session sequence	ROW_NUMBER per user
is_users_first_session	Boolean	First time user	user_session_number = 1
distinct_click_categories	Integer	Tab types clicked	COUNT(DISTINCT click_category)
had_tab_switch	Boolean	Clicked multiple tabs	distinct_click_categories > 1

---

searches_daily.parquet

Granularity: One row per day

Use case: Daily KPIs, trend analysis

Column	Type	Description	Calculation
date	Date	The day	
total_events	Integer	All events	COUNT(*)
unique_sessions	Integer	Distinct sessions	COUNT(DISTINCT session_key)
unique_users	Integer	Distinct users	COUNT(DISTINCT user_id)
unique_search_terms	Integer	Distinct search queries	COUNT(DISTINCT search_term_normalized)
search_starts	Integer	SEARCH_STARTED events	COUNT(SEARCH_STARTED)
result_events	Integer	SEARCH_RESULT_COUNT events	COUNT(SEARCH_RESULT_COUNT)
click_events	Integer	Click events	COUNT(click_category)
null_results	Integer	Zero-result events	SUM(is_null_result)
result_events_with_results	Integer	Results with >0 hits	SUM(is_clickable_result)
sessions_with_results	Integer	Sessions that got results	From session_stats CTE
sessions_with_clicks	Integer	Sessions with clicks	From session_stats CTE
sessions_abandoned	Integer	Results but no click	sessions_with_results - sessions_with_clicks
click_rate_pct	Float	Click rate	click_events / search_starts * 100
null_rate_pct	Float	Null result rate	null_results / result_events * 100
session_success_rate_pct	Float	Session success	sessions_with_clicks / sessions_with_results * 100
session_abandonment_rate_pct	Float	Session abandonment	sessions_abandoned / sessions_with_results * 100
avg_searches_per_session	Float	Avg searches per session	search_starts / unique_sessions
avg_search_term_length	Float	Avg query char length	AVG(search_term_length)

Column	Type	Description	Calculation
`avg_search_term_words`	Float	Avg query word count	AVG(search_term_word_count)
`sum_search_term_length`	Integer	Sum of query lengths	SUM(search_term_length) - for weighted avg in DAX
`sum_search_term_words`	Integer	Sum of word counts	SUM(search_term_word_count) - for weighted avg in DAX
`search_term_count`	Integer	Count of queries	COUNT(search_term_length IS NOT NULL)
`first_searches_of_day`	Integer	First searches of day	COUNT(is_first_search_of_day)
`clicks_general`	Integer	General tab clicks	COUNT(click_category='General')
`clicks_all`	Integer	All tab clicks	COUNT(click_category='All')
`clicks_news`	Integer	News tab clicks	COUNT(click_category='News')
`clicks_goto`	Integer	GoTo tab clicks	COUNT(click_category='GoTo')
`clicks_people`	Integer	People tab clicks	COUNT(click_category='People')
`day_of_week`	String	Day name	DAYNAME(session_date)
`day_of_week_num`	Integer	ISO day number (1=Mon)	ISODOW(session_date)
`searches_morning`	Integer	Searches 6:00-12:00	Hour-based filter
`searches_afternoon`	Integer	Searches 12:00-18:00	Hour-based filter
`searches_evening`	Integer	Searches 18:00-24:00	Hour-based filter
`searches_night`	Integer	Searches 0:00-6:00	Hour-based filter
`new_users`	Integer	First-time users today	Users where first_seen = today
`returning_users`	Integer	Repeat users today	Users where first_seen < today

---

searches_terms.parquet

Granularity: One row per search term per day

Use case: Search term performance analysis, content gap identification

Column	Type	Description	Calculation
`session_date`	Date	The day	
`search_term`	String	Normalized search query	LOWER(TRIM(query))
`word_count`	Integer	Words in query	COUNT of spaces + 1
`search_count`	Integer	Times searched today	COUNT(SEARCH_STARTED)
`unique_users`	Integer	Users who searched this	COUNT(DISTINCT user_id)
`unique_sessions`	Integer	Sessions with this term	COUNT(DISTINCT session_key)
`result_events`	Integer	Result events for term	COUNT(SEARCH_RESULT_COUNT)
`null_result_count`	Integer	Zero-result count	SUM(is_null_result)
`click_count`	Integer	Clicks from this term	COUNT(click_category)
`clicks_general`	Integer	General tab clicks	COUNT(click_category='General')
`clicks_all`	Integer	All tab clicks	COUNT(click_category='All')
`clicks_news`	Integer	News tab clicks	COUNT(click_category='News')
`clicks_goto`	Integer	GoTo tab clicks	COUNT(click_category='GoTo')
`clicks_people`	Integer	People tab clicks	COUNT(click_category='People')
`avg_sec_to_click`	Float	Avg decision time	AVG(ms_result_to_click) / 1000
`clicks_with_timing`	Integer	Clicks with timing data	COUNT(click after SEARCH_RESULT_COUNT)
`sum_sec_to_click`	Float	Sum of click times	SUM(ms_result_to_click) / 1000 - for weighted avg in DAX
`searches_morning`	Integer	Searches 6:00-12:00	Hour-based filter
`searches_afternoon`	Integer	Searches 12:00-18:00	Hour-based filter
`searches_evening`	Integer	Searches 18:00-24:00	Hour-based filter
`searches_night`	Integer	Searches 0:00-6:00	Hour-based filter
`first_seen_date`	Date	First day term appeared	MIN(session_date) over all time

Column	Type	Description	Calculation
`is_new_term`	Boolean	First appearance today	session_date = first_seen_date

6. Power BI Calculated Columns

These columns are created in Power BI using DAX and are not present in the parquet files.

searches_terms Table

Query_Length_Bucket

Categorizes search queries by word count for visualization.

```
Query_Length_Bucket =
SWITCH(
    TRUE(),
    searches_terms[word_count] = 1, "1 word",
    searches_terms[word_count] = 2, "2 words",
    searches_terms[word_count] = 3, "3 words",
    searches_terms[word_count] = 4, "4 words",
    searches_terms[word_count] >= 5, "5+ words",
    "Unknown"
)
```

Query_Length_Sort

Sort order for Query_Length_Bucket. Set "Sort by column" in Power BI.

```
Query_Length_Sort =
SWITCH(
    TRUE(),
    searches_terms[word_count] = 1, 1,
    searches_terms[word_count] = 2, 2,
    searches_terms[word_count] = 3, 3,
    searches_terms[word_count] = 4, 4,
    searches_terms[word_count] >= 5, 5,
    99
)
```

Term_Outcome

Classifies search term performance into actionable categories.

```
Term_Outcome =
VAR nullRate = DIVIDE([null_result_count], [result_events], 0)
VAR ctr = DIVIDE([click_count], [search_count], 0)
RETURN
SWITCH(
    TRUE(),
    nullRate = 1, "Zero Results",
    nullRate > 0.5, "Mostly No Results",
    ctr = 0, "No Clicks",
    ctr < 0.2, "Low CTR",
    "Success"
)
```

Category	Meaning	Action
Zero Results	100% null rate	Content gap - add content
Mostly No Results	>50% null rate	Partial gap - improve coverage
No Clicks	Has results but 0 clicks	Poor relevance - tune ranking
Low CTR	<20% click rate	Suboptimal - review content
Success	Good performance	Monitor

searches_journeys Table

Journey_Type

Combines outcome and behavior flags for segmentation.

```
Journey_Type =
searches_journeys[journey_outcome] &
IF(searches_journeys[had_reformulation], " (Refined)", "") &
IF(searches_journeys[recovered_from_null], " (Recovered)", "")
```

---

7. Power BI Measures

These measures are created in Power BI for aggregated calculations.

Search Effectiveness Score

Combined metric considering both CTR and null rate. Higher is better.

```
Search Effectiveness Score =
VAR ctr = DIVIDE(SUM(searches_terms[click_count]), SUM(searches_terms[search_count]), 0)
VAR nullRate = DIVIDE(SUM(searches_terms[null_result_count]), SUM(searches_terms[result_events]), 0)
RETURN
(ctr * 100) - (nullRate * 50)
```

Score interpretation:

- Positive scores: Good performance (CTR outweighs null rate penalty)
- Near zero: Balanced but could improve
- Negative scores: High null rates hurting performance

Term CTR %

Click-through rate for search terms.

```
Term CTR % =
DIVIDE(
    SUM(searches_terms[click_count]),
    SUM(searches_terms[search_count]),
    0
) * 100
```

Term Null Rate %

Percentage of searches returning zero results.

```
Term Null Rate % =
DIVIDE(
    SUM(searches_terms[null_result_count]),
    SUM(searches_terms[result_events]),
    0
) * 100
```

Weighted Avg Search Term Length

Correctly weighted average across days (use instead of AVERAGE on avg_search_term_length).

```
Weighted Avg Search Term Length =
DIVIDE(
    SUM(searches_daily[sum_search_term_length]),
    SUM(searches_daily[search_term_count]),
    0
)
```

Weighted Avg Search Term Words

Correctly weighted average across days.

```
Weighted Avg Search Term Words =
DIVIDE(
    SUM(searches_daily[sum_search_term_words]),
    SUM(searches_daily[search_term_count]),
    0
)
```

Weighted Avg Sec to Click

Correctly weighted average click time (for terms aggregation).

```
Weighted Avg Sec to Click =
DIVIDE(
    SUM(searches_terms[sum_sec_to_click]),
    SUM(searches_terms[clicks_with_timing]),
    0
)
```

Example: Full Data Flow

Raw Input (from App Insights)

```
timestamp,name,user_Id,session_Id,CP_searchQuery,CP_totalResultCount
2025-01-15 10:30:15.123456,Search_Started,user123,sess456,budget report,
2025-01-15 10:30:15.234567,Search_Completed,user123,sess456,budget report,
2025-01-15 10:30:15.567890,Search_Result_Count,user123,sess456,,15
2025-01-15 10:30:18.890123,Search_Tab_Click,user123,sess456,,
```

After Processing (searches_raw.parquet)

timestamp	name	session_key	prev_event	ms_since_prev	search_term	is_null_result	click_categ
10:30:15.123	SEARCH_STARTED	2025-01-15_user123_sess456	NULL	NULL	budget report	NULL	NULL
10:30:15.234	SEARCH_COMPLETED	2025-01-15_user123_sess456	SEARCH_STARTED	111	NULL	NULL	NULL
10:30:15.567	SEARCH_RESULT_COUNT	2025-01-15_user123_sess456	SEARCH_COMPLETED	333	NULL	false	NULL
10:30:18.890	SEARCH_TAB_CLICK	2025-01-15_user123_sess456	SEARCH_RESULT_COUNT	3323	NULL	NULL	General

Aggregated (searches_journeys.parquet)

session_date	total_events	search_count	click_count	sec_search_to_result	sec_result_to_click	journey_outcome
2025-01-15	4	1	1	0.44	3.32	Success

Calculation breakdown:

- `sec_search_to_result`: 10:30:15.567 - 10:30:15.123 = 444ms = 0.44s
- `sec_result_to_click`: 10:30:18.890 - 10:30:15.567 = 3323ms = 3.32s
- `journey_outcome`: click_count > 0 --> "Success"

Version History

Version	Date	Changes
1.0	2025-01-15	Initial documentation
1.1	2025-01-16	Added missing parquet columns (click breakdowns, sort columns, timing aggregates), Power BI calculated columns section, Power BI measures section